

Quantum Computing and Quantum Neural Networks:

their foundation, optimisation, and application

Jessica Pointing

Department of Physics
University of Oxford
St Peter's College

A thesis submitted for the degree of
Doctor of Philosophy
31 August 2024

Dominus Illuminatio Mea
The Lord Is My Light

The University of Oxford's motto from the opening words of Psalm 27

Copyright © 2024 Jessica Pointing

Chapter cover images created with (classical, not quantum) neural networks. These images are owned by Jessica Pointing.



Acknowledgements

Thanks be to God for *establishing the laws of physics.*

"Do you know the laws of the heavens? Can you set up God's dominion over the earth?" (Job 38:33).

Thanks be to God for *creating the heavens and the earth.*

"When I consider your heavens, the work of your fingers, the moon and the stars, which you have set in place, what is mankind that you are mindful of them, human beings that you care for them?" (Psalm 8:3-4).

Thanks be to God for who He is, for He is *Good*, He is *Love*, He is *Truth*.

"Give thanks to the Lord, for he is good. His love endures forever" (Psalm 136:1). "We love because he first loved us" (1 John 4:19). "Jesus answered, 'I am the way and the truth and the life. No one comes to the Father except through me'" (John 14:6).

Thanks be to God for His *gift of salvation* through his Son Jesus Christ.

"For God so loved the world that he gave his one and only Son, that whoever believes in him shall not perish but have eternal life. For God did not send his Son into the world to condemn the world, but to save the world through him. Whoever believes in him is not condemned, but whoever does not believe stands condemned already because they have not believed in the name of God's one and only Son" (John 3:16-18).

Thanks be to God for this hope of *eternal life*.

"So will it be with the resurrection of the dead. The body that is sown is perishable, it is raised imperishable; it is sown in dishonour, it is raised in

glory; it is sown in weakness, it is raised in power; it is sown a natural body, it is raised a spiritual body. If there is a natural body, there is also a spiritual body" (1 Corinthians 15:42-44).

Thanks be to God for this *gospel (the good news).*

"By this gospel you are saved, if you hold firmly to the word I preached to you. Otherwise, you have believed in vain. For what I received I passed on to you as of first importance: that Christ died for our sins according to the Scriptures, that he was buried, that he was raised on the third day according to the Scriptures, and that he appeared to Cephas, and then to the Twelve. After that, he appeared to more than five hundred of the brothers and sisters at the same time, most of whom are still living, though some have fallen asleep. Then he appeared to James, then to all the apostles, and last of all he appeared to me also, as to one abnormally born" (1 Corinthians 15:2-7).

Thanks be to God for His *grace.*

"For all have sinned and fall short of the glory of God, and all are justified freely by his grace through the redemption that came by Christ Jesus" (Romans 3:23-24).

Thanks be to God that we can have a *relationship* with Him.

"Here I am! I stand at the door and knock. If anyone hears my voice and opens the door, I will come in and eat with that person, and they with me" (Revelation 3:20).

Thanks be to God for empowering us through the *Holy Spirit.*

"May the God of hope fill you with all joy and peace as you trust in him, so that you may overflow with hope by the power of the Holy Spirit" (Romans 15:13).

Thanks be to God for *prayer.*

"Do not be anxious about anything, but in every situation, by prayer and petition, with thanksgiving, present your requests to God" (Philippians 4:6).

Thanks be to God for the *Word of God.*

"Jesus answered, 'It is written: Man shall not live on bread alone, but on every word that comes from the mouth of God'" (Matthew 4:4).

Thanks be to God for the people in the *body of Christ.*

"And let us consider how we may spur one another on toward love and good deeds, not giving up meeting together, as some are in the habit of doing, but encouraging one another—and all the more as you see the Day approaching" (Hebrews 10:24-25).

I am grateful to God for all of the above, which provided a foundation during this PhD (**P**raise **H**im **D**aily) journey and life. I give thanks to God the Father, the Son Jesus Christ, and the Holy Spirit for all of the gifts He richly blesses us with, including the gift to do this research, write, and publish this work.

My fascination for physics at a younger age and belief in God were a response to the awe and wonder I experienced when I considered the universe and life. They were also a response to my pursuit of truth. "The heavens declare the glory of God; the skies proclaim the work of his hands" (Psalm 19:1). I thank God that I can continue that pursuit and have the opportunity to write this PhD thesis in physics and discover more about His creation through my research.

Acknowledgements

I thank God for his strength to persevere during this journey and equipping me with everything I needed to complete this thesis. I thank God for my family, friends, and others who have been there along this journey.

I thank God for these bible verses that helped me to endure when the PhD journey was difficult:

- "And we know that in all things God works for the good of those who love him, who have been called according to his purpose" (Romans 8:28).
- "Endure hardship as discipline; God is treating you as his children. For what children are not disciplined by their father?" (Hebrews 12:7) . . . "No discipline seems pleasant at the time, but painful. Later on, however, it produces a harvest of righteousness and peace for those who have been trained by it" (Hebrews 12:11).
- "Consider it pure joy, my brothers and sisters, whenever you face trials of many kinds, because you know that the testing of your faith produces perseverance. Let perseverance finish its work so that you may be mature and complete, not lacking anything" (James 1:2-4).

Personal Acknowledgements

To people and communities that have directly or indirectly helped me during my PhD

Family

Thank you to my family for their unwavering love and support. Thank you to my mother, who is the most prayerful person I know. I thank her for her endless and countless prayers, which God has stored in golden bowls (Revelation 5:8). I thank her for all of her wisdom and support. I thank my father for all of his prayers, wisdom, and support too. I thank him for being there to listen and to offer advice. I thank him for passing on his interest for science and technology to me and for inspiring me with his cool DIY projects. I thank my brother for all of his prayers, wisdom, and support too. I thank him for our insightful conversations, for his infectious enthusiasm for the Word of God, for helping open up my eyes to the beauty of Scripture, and for teaching me a bit of Hebrew!

Thank you to my extended family for their prayers and support throughout this journey. Thank you to my mum's life group of strong women who have showed me their love through their consistent prayers for me. I am so grateful to have sisters in Christ who have lifted prayers to God on my behalf and to have seen God's amazing answers to their prayers. Thank you so much Aggi and Emily.

Church

Thank you to the body of Christ, in particular my local church St Aldate's Church. I thank them for providing a community and home in Oxford. I thank the lovely people there that have encouraged and blessed me. I thank the postgraduate community there, in particular Owen and Laura who have led it so kindly. I also thank the sisters and brothers in Christ in the 'The Outer Room' group for being there to listen, to pray, and to laugh together. I am grateful to have met Anna there, who has cared for people in the church. I am also grateful to Alissa, for her authenticity, her support, and encouragement. I thank the Break Ground group for helping me grow in my discipleship in Jesus and I thank Emily and Jack in particular for their hospitality and care for us.

Christian Communities

Thank you to the Oxford Centre for Christian Apologetics (OCCA) Doctoral Fellowship, who have been a constant presence from the very beginning to the end of my PhD journey at Oxford. I thank them for helping me connect my faith and my work together and for helping me to see how God works in a broad range of disciplines and for enlightening me on philosophy and theology in particular. I thank Max for leading the group throughout these years. I thank all of the other OCCA Doctoral Fellows for our encouraging and enlightening times together.

Thank you to the Oxford Pastorate, in particular the Christian Leadership and Service group, which has hosted wonderful events on meaningful questions in life. Thank you to Kay for leading that group and for also being a supportive mentor for me. Thank you to Bethan for leading the Christians in Academia group and for organising these events, including the Good Life Reading Group. Thank you to the Oxford Pastorate for giving me the opportunity to combine my work and faith during the Oxford Connections Macedonia trip, which was a wonderful Christian academic missions trip. Thank you to Dr Jonathan Brant

for leading the trip and thank you to Dr Kosta and Nada Milkov for welcoming us and for inviting us into their lives. It was great to see the admirable work they are doing in Macedonia and their love for their country. Thank you to everyone else on that trip for providing such a meaningful time to work together in Christ.

Friends

Thank you to all of my friends for being there throughout this journey. Thank you for all of the moments of joy and sorrow we have shared together. I thank God for bringing Nadja and I together through an unlikely place in a gymnastics class and for growing our friendship throughout my PhD journey. I thank Nadja for her support, her care, her curiosity and all of the beautiful memories we have together at Oxford. I thank Athena and Tia for their support and sister-like love throughout this time. I thank Adaku for her fun energy, inspiring me with her many endeavours, and for sticking together in the Oxford Blockchain Society. I thank Nam and Jordan for their infectious energy, their humour, and for starting and leading a bible study group for helping us put into practice Hebrews 10:25, to not "neglect meeting together, as some have the

habit of doing. Rather, let us encourage each other, and all the more as you see the Day approaching." I thank Sebastian for encouraging me and helping me keep accountable with some of my work, including one of the papers in this thesis. I am grateful to Ning for our friendship and meaningful conversations. I am grateful to have met Sam (and to discover the similarities we have!). I'm inspired by her growth in the Lord and her friendliness to those around her. I am also grateful to have met Maria, for inspiring me with her walk in the Lord and for shining joy to those around her.

Societies and Communities

Thank you to all of the lovely people I have met at St Peter's College, in particular those in Paradise, who provided such a friendly community and good times during the COVID lockdown. Thank you to the porters for being a smiling face and helping me with many logistical matters and thank you to the scouts at Paradise.

Thank you to the gymnastics society for helping me be treasurer and for the fun of doing gymnastics together. Thank you to Buzz Gym and the instructors

there who provided fun workouts that I really enjoyed to help my body and mind.

Thank you to the Oxford Blockchain Society for providing the Chainshot boot-camp to help me learn Solidity and blockchain development and for providing sponsorship for hackathons and conferences. Thank you to Keivan for his authenticity and working together for the start of the Oxford DAO. Thank you to Brandon for the fun energy he brings. Thank you to Brandon and Jan for being fellow quantum-blockchainers at Oxford. Thank you to the Retrox team, including Kyle, Jan, Orlando, Konrad, and Fernando for fun hackathons, creating dApps, insightful discussions, and enlightening me on the connection between academia and blockchain through DeSci.

Thank you to the Esbjerg Hovedbibliotek, the well-equipped library in Denmark where I am writing these words and where I have spent the last few weeks of writing this thesis.

Organisations

Thank you to the World Economic Forum for a wonderful opportunity to be a part of their Global Future Council on Quantum Computing and for providing valuable experience on the business and political aspects of quantum computing. I thank Mr. Grigory Shutko for his leadership and support.

Thank you to the Knight-Hennessy Scholarship program, in particular Professor John Hennessy, for such a wonderful opportunity at the start of my PhD journey. I am grateful for the kind community, valuable events on leadership, beautiful space in Denning House, memorable trips, and financial support. Thank you to all of the brilliant people there who made it a meaningful experience. Thank you to the Knight-Hennessy Christians group for their love and encouragement. I thank the Stanford Quantum Computing Association team for all of their work into making a great society.

Thank you to HomeDAO for providing a home and community for me in Oxford. I thank Josh for selecting me to have that opportunity that helped support me during my PhD journey. I thank Josh for his leadership and friendship,

especially through a couple of difficult times. I thank Simon for his care to help others grow, and for inviting me to be a part of DirectEd to share my experiences.

I thank ETHGlobal and the Ethereum Foundation for all of the amazing events they've hosted that I have enjoyed and for their support. I thank Certora and all of the helpful people there who helped me to grow as a blockchain security researcher and for providing wonderful opportunities that supported me throughout my PhD journey. Thank you to Neo for a fantastic community, amazing retreats, and thoughtful gifts. Thank you to the Delta Fellowship for the opportunity to learn and for their support. Thank you to the Society of Exploration Geophysicists (SEG) for their support.

Professional Acknowledgements

To people and organisations in academia

Thank you to Professor Dieter Jaksch for providing this opportunity to do a PhD in Physics at Oxford and for all of his advice on research. I thank all

of the wonderful researchers in his group, who have been supportive. I thank Dr. Ben Jaderberg who has been very helpful, providing valuable guidance, and encouraging me by successfully completing his PhD in a field similar to this thesis topic. I thank Dr. Lewis Anderson who helped provide a piece for a proof and who was helpful in discussions on the research. I thank Dr. Nikita Gourianov for a variety of interesting discussions, for his advice, and for a consistent presence in the office. I thank Mr. Christopher Willby for our fruitful discussions and our shared experiences whilst writing a paper. I thank Mr. Oliver Chapman for his courage and engaging discussions. I thank Mr. Tom O'Leary for our discussions as well. I thank Dr. Martin Kiffner for our research meetings and his feedback on my research.

Thank you to Professor Ard Louis for responding with interest after I suggested to him one of the research projects in this thesis. I thank him and Mr. Chris Mingard for helping me understand more deeply their previous work on simplicity bias in deep neural networks. I thank them for various discussions on the project and helping me make the narrative of the project clearer.

Thank you to Professor Alex Aiken for contacting me to work on the quantum

compiler project. I thank him, Dr. Oded Padon, Dr. Zhihao Jia, Mr. Henry Ma, Mr. Auguste Hirth, and Professor Jens Palsberg for their involvement on the project. Thank you to Mr. Joe Latone and Dr. Mario Motta for providing the opportunity to connect with IBM, learn and grow in research, and work together on a quantum application. I thank Mr. Yousef Hindy, Dr. Emine Meltem Tolunay and Mr. Sreeram Venkatarao for their involvement on that project. I thank Mr. Jay Borenstein, who runs the CS210 course at Stanford, for providing an enjoyable environment and opportunity to work together with other students and professionals on the quantum application project. I thank Mr. Jonathan Candelaria for all of the work he did on helping quantum computing move forward at Stanford and beyond; I thank him for his effort and enthusiasm that made a lot of things possible, including the quantum application project with IBM. Thank you to Professor Aviad Rubinstein for his kindness and support in the earlier part of my PhD journey.

Thank you to the Department of Physics, including Professor Peter Norreys, Ms. Erin Wheeler, Professor Simon Hooker, and Dr. Hannah Lingard for providing support for the submission of this thesis. Thank you to Professor Andrew Daley for kindly taking the time to read my thesis and provide feedback

on it. Thank you to Dr. Mustafa Bakr, my college advisor, who has been very supportive. I thank him for taking the time to meet with me, to read my papers, and to give me advice. Thank you to Professor Sonia Antoranz Contera, who has also been very supportive and has given her time to help me. I thank her for walking alongside me in the latter part of my PhD journey and for giving me hope both in her words and action. I thank her for all of the work she does behind-the-scenes to help people, including me, to finish their journeys and learn life lessons.

Thank you to Professor Seth Lloyd, who provided me with a research opportunity when I was exploring research in physics at MIT. I am also grateful for his support that enabled me to start this PhD. Thank you to Professor Alan Aspuru-Guzik, who also provided me with an opportunity to do research early on at Harvard and introduced me to variational quantum algorithms, one of the topics in this thesis. Thank you to Dr. Yudong Cao for providing research guidance before this PhD and helping me learn about variational quantum algorithms, which inspired me to explore these algorithms during my PhD. Thank you to Dr. Isaac Kim for also providing research support early in my PhD journey.

I thank the Engineering and Physical Sciences Research Council (EPSRC) Doctoral Training Partnership (DTP) Scholarship and the Department of Physics for funding.

In Memory Acknowledgements

To people who were part of my PhD journey but who are tragically no longer here

Thank you to Peter Wittek who kindly invited me to Toronto in Canada to be a graduate student researcher there and connected me with the Creative Destruction Lab and Xanadu. I thank him for contributing to my knowledge on quantum machine learning, for advising me on a project, and for the inspiring ways he did things. I am so sad about your tragic trip. May you rest in peace Peter.

Thank you to Alex who helped start the wonderful HomeDAO community, which provided a home for me in Oxford, who beamed with energy, and made an impact on me. Your short life is heartbreaking. May you rest in peace Alex.

Acknowledgements Closing

I am grateful for all of these people whom God has blessed me with. I want to acknowledge that these acknowledgements naturally have focused on the positive and good parts of the PhD journey. I also want to acknowledge that there were difficult times.

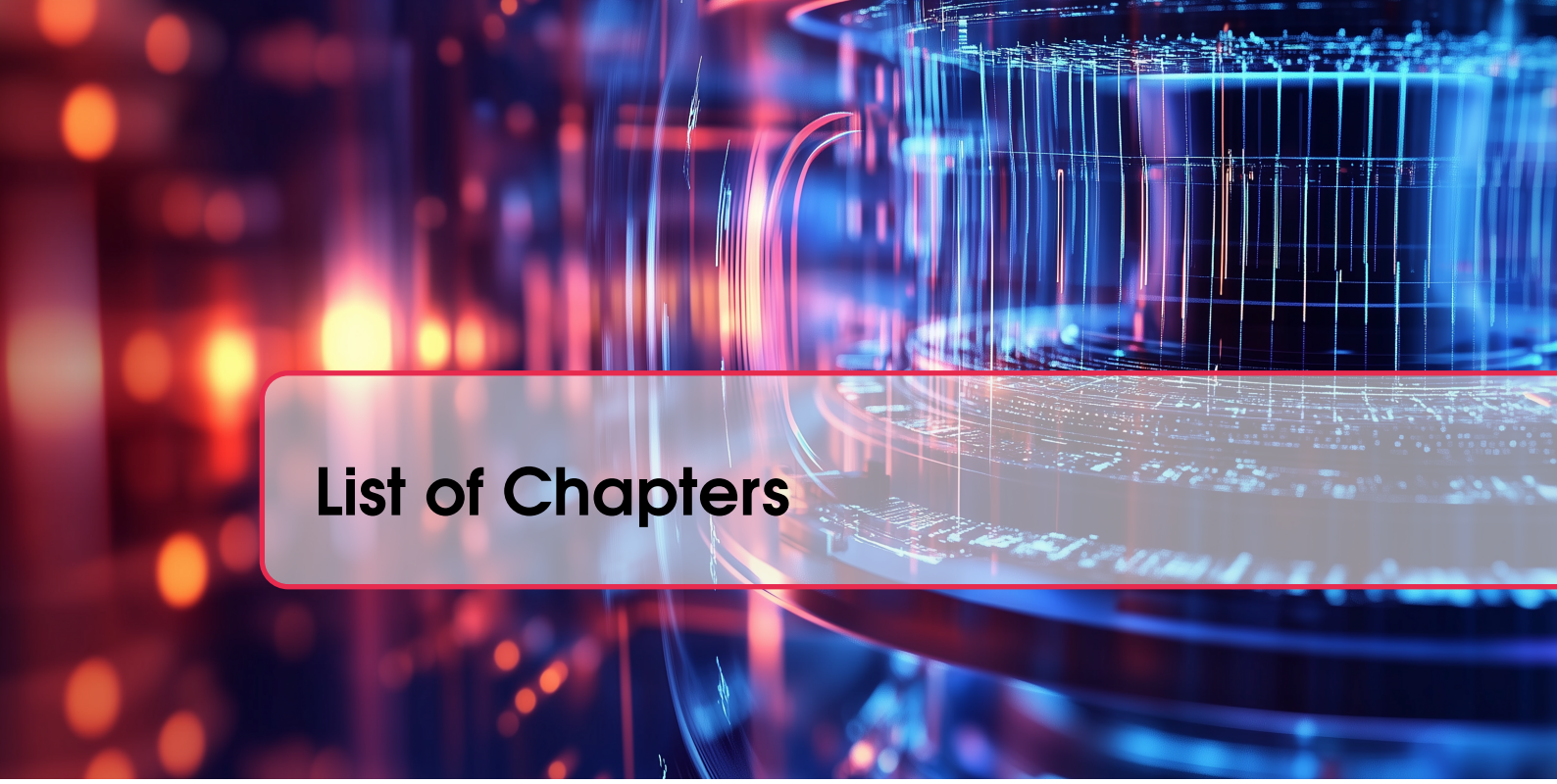
There are many other people, communities, and organisations that I have not mentioned that have been a part of this journey and I am very grateful to them. To those I've acknowledged and others, thank you! Your involvement has made an impact on me and has made this thesis possible and I want to honour all of you in these acknowledgements. To God be the glory, forever and ever, Amen!



Abstract

Quantum computing is a fascinating discipline, combining the laws of quantum physics with the practicalities of computing. Certain problems could potentially be solved faster on quantum computers, and researchers hope to use them to solve difficult problems in industries such as healthcare and finance. Quantum machine learning and quantum neural networks (QNNs) aim to solve some of these problems. Open questions remain for their utility, and this thesis aims to contribute to answers by exploring the foundation, optimisation, and application of quantum algorithms. We explore the foundation of QNNs by studying their bias and expressivity. We investigate whether a QNN can have a classical deep neural network's (DNN) bias, known as simplicity bias, which is hypothesised to contribute to the success of DNNs. We run numerical experiments and derive

proofs to demonstrate how the type of QNN affects its bias and expressivity. Our results suggest a bias-expressivity tradeoff, highlighting the need for alternative frameworks to make QNNs effective general-purpose learning tools for classical data. We explore the optimisation of quantum circuits by creating a new quantum compiler with a novel technique: we automatically generate quantum circuits that are equivalent and design a tiling method to optimise larger circuits according to a customisable cost function. We generate novel equivalent circuits, compile custom quantum gates unlike previous compilers, and outperform existing compilers on certain circuits. We explore the application of quantum computers by constructing a quantum algorithm to solve a real-world problem in the mining industry. We provide a mapping for this problem onto a quantum circuit and run simulator and hardware experiments to find solutions to small-scale instances of the problem, which is the first attempt to apply such quantum algorithms to this mining problem. This thesis also presents a framework to identify definitions, key questions, and goals for quantum advantage.



List of Chapters

1 Introduction 80

2 Quantum Computing 120

3 Quantum Neural Networks 155

4 Classical Neural Networks 229

5 Categories of Quantum Advantage and Algorithms 265

6 Foundation of Quantum Neural Networks 294

7 Optimisation of Quantum Circuits 384

8 Conclusion 451

A Application of Quantum Computers 479



Table of Contents

Acknowledgements 3

Abstract 22

List of Chapters 24

Table of Contents 25

List of Figures 45

List of Tables 72

1 Introduction 80

1.1 Context 81

1.2 Framework for Quantum Algorithms 88

1.3 Thesis Structure 91

1.3.1 Foundation of Quantum Neural Networks 92

1.3.2 Optimisation of Quantum Circuits 93

1.3.3 Application of Quantum Computers 94

1.4 Contributions 94

1.4.1 Objective 94

1.4.2 Foundation of Quantum Neural Networks 95

1.4.2.1 Context 95

1.4.2.2 Summary of Work 97

1.4.2.3 Summary of Results 98

1.4.2.4 Contributions to the Field 99

1.4.3 Optimisation of Quantum Circuits 102

1.4.3.1 Context 102

1.4.3.2 Summary of Work 104

1.4.3.3 Summary of Results 105

1.4.3.4 Contributions to the Field 105

1.4.4 Application of Quantum Computers 107

1.4.4.1 Context 107

1.4.4.2 Summary of Work 109

1.4.4.3 Summary of Results 110

1.4.4.4 Contributions to the Field 110

1.4.5 Categories of Quantum Advantage and Algorithms 111

1.5 Thesis Chapters 112

1.5.1 Chapter 1: Introduction 112

1.5.2 Chapter 2: Quantum Computing 112

1.5.3 Chapter 3: Quantum Neural Networks 113

1.5.4 Chapter 4: Classical Neural Networks 114

1.5.5 Chapter 5: Categories of Quantum Advantage and Algorithms 115

1.5.6 Chapter 6: Foundation of Quantum Neural Networks 115

1.5.7 Chapter 7: Optimisation of Quantum Circuits 115

1.5.8 Chapter 8: Conclusion 116

1.5.9 Appendix A: Application of Quantum Computers 116

1.6 Publications 117

1.7 Computational Resources 119

2 Quantum Computing 120

2.1 Preview 121

2.2 Computing 122

2.3 Quantum Physics 124

2.4 Quantum Computing 125

2.5 Quantum Algorithms 125

2.6 Quantum Hardware 126

2.6.1 Brief History 126

2.6.2 Categories of Quantum Hardware 127

2.6.3 Levels of Quantum Hardware 128

2.6.4 Outlook for Quantum Hardware 131

2.7 Quantum Cloud Platforms 133

2.8 Quantum Software 135

2.9 Quantum Tools 136

2.10 Quantum Applications 137

2.11 Quantum Advantage 138

2.12 NISQ and Fault-Tolerant Devices 139

2.13 Variational Quantum Algorithms 141

2.14 Quantum Machine Learning 146

2.14.1 First Wave of Quantum Machine Learning 146

2.14.1.1 Grover-Inspired Algorithms 149

2.14.2 Adiabatic Computing 149

2.14.3 Second Wave of Quantum Machine Learning 151

2.15 Future Directions 152

2.16 Review 153

3 Quantum Neural Networks 155

3.1 Preview 156

3.2 Quantum Neural Networks Overview 156

3.3 History 157

3.4 How Quantum Neural Networks Work 164

3.4.1 Encoding Methods 166

3.4.1.1 Basis Encoding 166

3.4.1.2 Amplitude Encoding 167

3.4.1.3 ZZ Feature Map 167

3.4.1.4 Random Relu Transform 169

3.4.1.5 Other Encoding Methods 169

3.4.2 Variational Circuit 170

3.4.2.1 Expressivity of Variational Circuits 171

3.4.2.2 Fully Boolean Expressive QNNs 172

3.4.2.3 Non-Fully Expressive QNNs 177

3.4.3 Measurement Operators 179

3.4.4 Learning 180

3.4.5 Analytic Gradients 181

3.5 Quantum Kernels 183

3.5.1 How to Use a Quantum Kernel 189

3.5.1.1 Define a Quantum Feature Map 190

3.5.1.2 Construct the Kernel Matrix 190

3.5.1.3 Basis Encoding Kernel Matrix 192

3.5.1.4 ZZ Feature Map Kernel Matrix 193

3.5.1.5 Random Relu Transform Kernel Matrix 195

3.5.1.6 Amplitude Encoding Kernel Matrix 198

3.6 Characteristics of QNNs from the Literature 200

3.6.1 Barren Plateaus 200

3.6.2 Model Capacity 203

3.6.3 Role of Data 205

3.6.3.1 Classical Models Can Be Elevated through Data 206

3.6.3.2 Error Bounds 207

3.6.3.3 Projected Kernels 209

3.6.3.4 Constructed Datasets 210

3.6.4 Inductive Bias in Quantum Kernels 211

3.7 Quantum Advantage 213

3.7.0.1 Quantum Speedup Cases 216

3.7.0.2 Quantum Speedup in Supervised Learning 218

3.7.0.3 Challenges for Demonstrating Quantum Advantage in Machine Learning 222

3.8 State of the Field 226

3.9 Review 227

4 Classical Neural Networks 229

4.1 Preview 230

4.2 Neural Networks Overview 230

4.3 History 231

4.4 Understanding Neural Networks 235

4.5 Inductive Bias 242

4.5.1 Defining and Quantifying Bias 242

4.5.2 Simplicity Bias in Deep Neural Networks 244

4.5.3 Simplicity Bias and Generalisation 245

4.5.4 No Free Lunch Theorem 246

4.6 Algorithmic Information Theory 248

4.6.1 Simplicity Bias 248

4.6.2 Complexity of a Function 253

4.7 How Neural Networks Work 254

4.7.1 Neurons 254

4.7.2 Multi-Layer Neural Networks 256

4.7.3 Training a Neural Network 257

4.7.4 Generalisation Error 259

4.7.5 Expressivity 259

4.8 Dataset for Inductive Bias 261

4.8.1 The Boolean System 261

4.8.2 Boolean System with Deep Neural Networks 262

4.9 Review 263

5 Categories of Quantum Advantage and Algorithms 265

5.1 Preview 265

5.2 Quantum Algorithms 266

5.3 Quantum Advantage 271

5.3.1 Purpose of Quantum Advantage 271

5.3.2 Measures for Quantum Advantage 273

5.3.3 Categories of Quantum Advantage 276

5.3.3.1 Flowchart Symbols 276

5.3.3.2 Flowchart Explanation 276

5.3.3.3 Goals 290

5.4 Review 292

6 Foundation of Quantum Neural Networks 294

6.1 Preview 295

6.2 Summary 296

6.3 Introduction 297

6.4 Results 300

6.4.1 Summary of the Bias and Expressivity of Different Encoding Methods 303

6.4.2 Summary of Results 306

6.4.3 Inductive Bias of QNNs 308

6.4.4 Generalisation Error of QNNs 312

6.4.5 Expressivity of QNNs 316

6.5 Methods 317

6.5.1 Finding Inductive Bias in QNNs 317

6.5.2 Finding Inductive Bias in Quantum Kernels 320

6.5.3 Comparison Between the Methods 322

6.5.4 Training QNNs 324

6.6 Discussion 324

6.6.1 Limitations and Future Work 327

6.7 Proofs 330

6.7.1 Proof of Expressivity of Amplitude Encoding 330

6.7.1.1 Example 331

6.7.2 Proof That Fully-Expressive Random QNNs with Basis Encoding Have No Bias 337

6.8 Additional Experiments and Results 348

6.8.1 Inductive Bias in QNNs 348

6.8.1.1 Probability of Complexity vs Complexity 348

6.8.1.2 Entropy vs Complexity 349

6.8.1.3 Probability of Boolean Function vs Rank 352

6.8.2 Inductive Bias in Quantum Kernels 354

6.8.2.1 The Kernel Matrix 354

6.8.2.2 Kernel Matrix Plots 355

6.8.2.3 Kernel Eigenvalues 355

6.8.2.4 Probability of Boolean Function vs Complexity 357

6.8.2.5 Probability of Complexity vs Complexity 358

6.8.2.6 Probability of Boolean Function vs Rank 359

6.8.2.7 Entropy vs Complexity Plots 360

6.8.3 Training and Test Datasets for Boolean Data 360

6.8.4 Generalisation Error of QNNs 362

6.8.4.1 Target Functions 362

6.8.4.2 Generalisation Error vs Complexity, Entropy, and Probability of Target Function 363

6.8.4.3 Generalisation Error vs Complexity of Output Function 367

6.8.5 Quantum Generalisation Performance on Parity Function 369

6.8.6 Training QNNs 371

6.8.7 Inductive Bias of Quantum Convolutional Neural Networks (QCNNS) 377

6.8.7.1 Definition of QCNNs 377

6.8.7.2 Experiment Details 379

6.8.7.3 Probability of Boolean Function vs Complexity 380

6.9 Review 381

7 Optimisation of Quantum Circuits 384

7.1 Preview 385

7.2 Summary 386

7.3 Introduction 387

7.4 Results 397

7.4.1 Generating Existing Circuit Identities 397

7.4.1.1 Example of Known Circuit Identities 403

7.4.2 Generating Novel Circuit Identities 405

7.4.3 Optimising Quantum Circuits 406

7.4.4 Benefits of Optimisation 407

7.4.4.1 Execution Times 407

7.4.4.2 Noise Reduction 409

7.5 Discussion 414

7.5.1 Related Work 414

7.5.1.1 Automatically Generating Identities 414

7.5.1.2 Superoptimisation 414

7.5.1.3 Quantum Optimisation 415

7.5.2 Limitations and Future Work 416

7.5.2.1 Scalability 416

7.5.2.2 Missed Substitutions and Larger Circuits 416

7.5.2.3 Parametrised Gates 417

7.5.2.4 Noise Effects 417

7.5.2.5 Customised cost functions 417

7.5.3 Conclusion 418

7.6 Methods 419

7.6.1 Circuit Identity Generator 419

7.6.1.1 Generation Algorithm 419

7.6.2 Scaling 428

7.6.3 Optimiser 432

7.6.3.1 Algorithm 432

7.6.3.2 Reordering Qubits 440

7.6.4 Runtime 441

7.6.5 Implementation 443

7.6.5.1 Circuit Module 443

7.6.5.2 Subs Module 444

7.6.5.3 Optimise Module 444

7.7 Review 444

8 Conclusion 451

8.1 Overview 451

8.2 Thesis Summary 452

8.2.1 Thesis Introduction 452

8.2.2 Quantum Computing Chapter 453

8.2.3 Quantum Neural Networks Chapter 455

8.2.4 Classical Neural Networks Chapter 456

8.2.5 Categories of Quantum Advantage and Algorithms Chapter 458

8.2.6 Foundation of Quantum Neural Networks Chapter 459

8.2.7 Optimisation of Quantum Circuits Chapter 460

8.2.8 Application of Quantum Computers Chapter 462

8.3 Research Summary and Future Research 463

8.3.1 Foundation 463

8.3.1.1 Summary of Results and Contributions 463

8.3.1.2 Future Research 466

8.3.2 Optimisation 468

8.3.2.1 Summary of Results and Contributions 468

8.3.2.2 Future Research 470

8.3.3 Application 472

8.3.3.1 Summary of Results and Contributions 472

8.3.3.2 Future Research 474

8.4 Concluding Remarks 477

A Application of Quantum Computers 479

A.1 Preview 480

A.2 Summary 482

A.3 Introduction 482

A.4 Methods 488

A.4.1 Hamiltonian Formulation 488

A.4.2 Domain Decomposition 495

A.5 Results 497

A.5.1 Simulations on Classical Computers 501

A.5.2 Domain Decomposition 504

A.5.3 Hardware Experiments 505

A.6 Conclusions 509

A.7 Comparison Between Optimisers 512

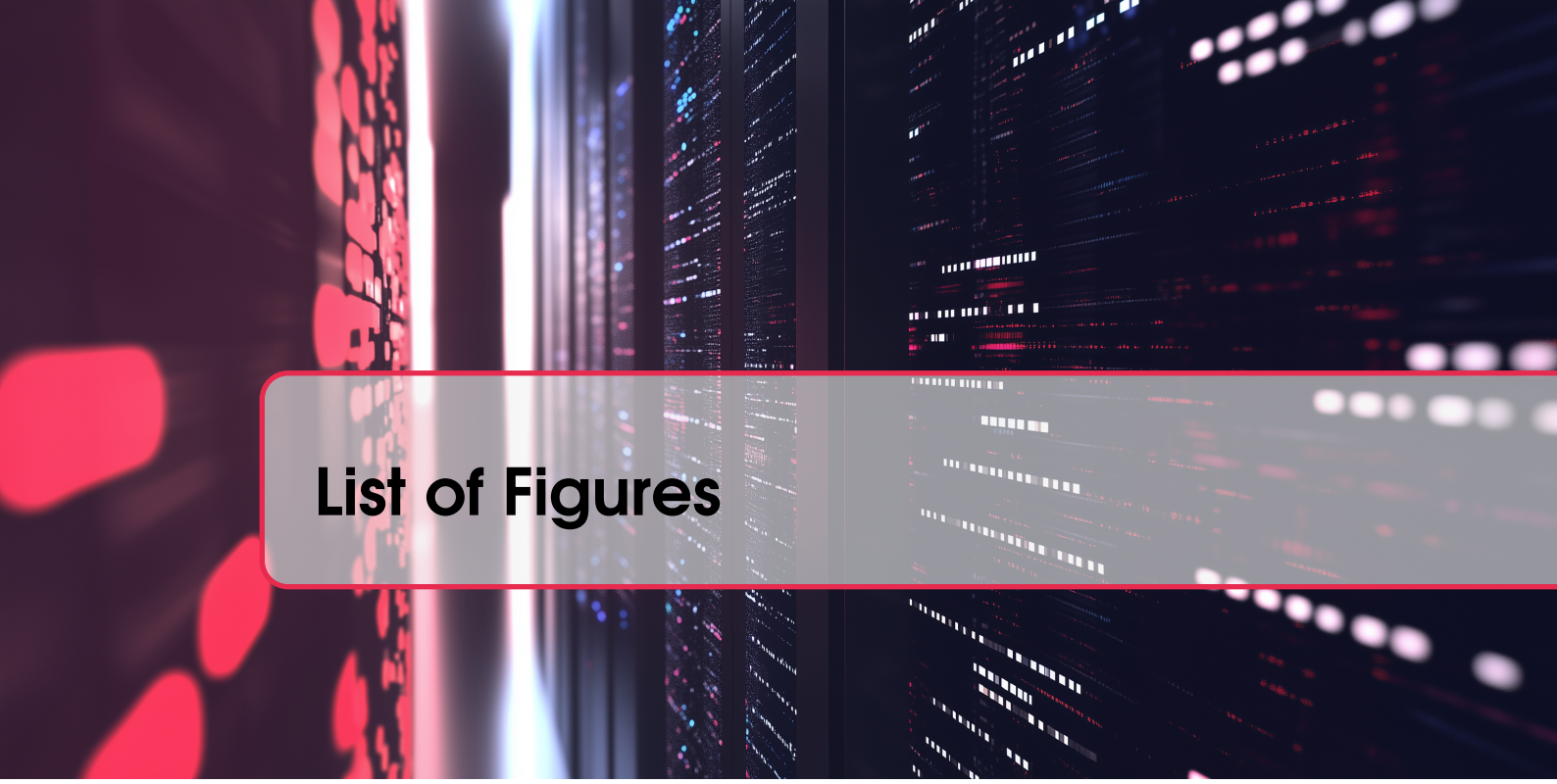
A.8 Fragmentation Details 513

A.8.1 Comparison Between Fragmentation Strategies 517

A.9 Data Availability 520

A.10 Review 521

References 523



List of Figures

1.1 **Conceptual layers of quantum algorithms explored in this thesis.** From the foundation, to optimisation, to application of quantum algorithms. . 89

2.1 **Variational quantum algorithms process.** A quantum ansatz (in this case, a parametrised quantum circuit with a particular structure) has an output of $|\psi(\theta)\rangle$ which is then used for calculating a cost function on the classical computer. The optimiser uses this cost function to find a new set of parameters θ which are then used in the quantum ansatz. The process repeats until an optimal set of parameters is found. 143

3.1 **Quantum neural network with three distinct parts.** The quantum neural network consists of the (1) Encoder Circuit, which encodes the data into the quantum neural network, the (2) Variational Circuit, which is parametrised and has its parameters optimised, and the (3) Measurement Operators, which retrieve classical information from the quantum neural network. Figure inspired by Figure 2 in Reference (125). 165

3.2 **Learning process for a quantum neural network.** The classical information retrieved from the measurement operators are fed into a classical optimiser to compute the loss, gradients, and update the parameters. These new parameters are updated in the parametrised quantum gates in the variational circuit and the process is repeated until the output of the quantum neural network converges or the process satisfies a stopping condition. 166

3.3 **ZZ feature map for two qubits.** This feature map is used for encoding data into a QNN. The feature map for two qubits consists of Hadamard gates, followed by U_1 gates where the first data point x_0 is encoded in the parameter for the U_1 gate acting upon the first qubit q_0 and the second data point x_1 is encoded in the parameter for the U_1 gate acting upon the second qubit q_1. A CX gate follows, then another U_1 gate on the second qubit q_1, which encodes both data points x_0 and x_1 in the parameter for the U_1 gate. Another CX gate follows. Figure inspired by Reference (128). 168

3.4 **Data re-uploading.** The classical input data $\vec{x}$ is re-uploaded into different layers in the quantum circuit, alternated with the parametrised (trainable) quantum gates with parameters $\vec{\theta}$. Figure inspired by Reference (131). 170

3.5 **Quantum circuit for expressing the Boolean function 00101001.** The Boolean function in Table 3.1 can be expressed as this quantum circuit. The circuit uses multi-controlled X gates to express the behaviour of the Boolean function. Quantum circuit inspired by Figure 2 in Reference (132). 174

3.6 **Boolean QNN for three inputs.** This QNN can express any Boolean function with length three. Quantum circuit inspired by Figure 3 in Reference (132). 175

3.7 **Boolean QNN for three inputs.** This QNN can express any Boolean function of length three and is overparametrised as it contains more parameters needed to express any Boolean function of length three. 176

3.8 **Variational quantum circuit.** This circuit includes the XX gate, which is the $e^{i\theta X \otimes X}$ two-qubit gate. Figure inspired by Reference (133). 178

3.9 **Linear decision boundary with a feature map.** This figure shows an example of how a feature map can enable a linear decision boundary for the data. The left plot shows the original data points in 1-dimension; in 1-dimension the classes for these data points are not linearly separable. One can apply the feature map $\Phi(x) = |x|$ to the data points, which takes the absolute value of the data points and maps them to 2-dimensions - shown in the right plot. The classes for these data points are now linearly separable in 2-dimensions. These figures were inspired by Reference (137). 185

3.10 **Kernel matrices for $\mathbf{n = 2}$.** The kernel matrices calculated using $K_{ij} = \left|\langle\phi^{\dagger}(\vec{x}_j)|\phi(\vec{x}_i)\rangle\right|^2$, where $\vec{x_i}$ consist of all the combinations of binary strings of length two. 191

3.11 **Quantum circuits for basis encoding.** These quantum circuits encode the data using basis encoding. (a) This circuit which consists of no quantum gates (equivalent to Identity gates) encodes the data $x_0 = 00$ as the qubits q_0 and q_1 are initially in the state $|0\rangle\,|0\rangle$. (b) This circuit consists of no gate on the q_0 qubit and applies the X gate to the qubit q_1 as it flips the state of this qubit to $|1\rangle$ and therefore encodes the data $x_1 = 01$. (c) This circuit applies the X gate to the qubit q_0 as it flips the qubit to the state $|1\rangle$ and does nothing to the qubit q_1 and therefore encodes $x_2 = 10$. (d) This circuit applies the X gate to both of the qubits q_0 and q_1 so that it encodes the data $x_3 = 11$. 192

3.12 **ZZ Feature Map for two qubits that encodes 01.** This figure shows the quantum circuit that encodes the data $x_1 = 01$: the 0th element of x_1 is $x_{1_0} = 0$ and the 1st element of x_1 is $x_{1_1} = 1$. In Figure 3.12a, the quantum circuit consists of two H (Hadamard) gates, where $H = \frac{1}{\sqrt{2}}\left(\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right)$ followed by two U_1 gates applied to both qubits, where $U_1(\theta) = \left(\begin{smallmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{smallmatrix}\right)$. The angle θ of the U_1 gate applied to the first qubit q_0 is $2x_{1_0}$ which is equal to 0 in this example. Therefore this matrix is equivalent to $U_1(0) = \left(\begin{smallmatrix} 1 & 0 \\ 0 & e^{i0} \end{smallmatrix}\right) = \left(\begin{smallmatrix} 1 & 0 \\ 0 & i \end{smallmatrix}\right) = S$, where S is the phase matrix. The angle θ of the U_1 gate applied to the second qubit q_1 is $2x_{1_1}$ which is equal to $2 \cdot 1 = 2$ in this example. Therefore this matrix is equivalent to $U_1(2) = \left(\begin{smallmatrix} 1 & 0 \\ 0 & e^{2i} \end{smallmatrix}\right)$. These U_1 gates are followed by a CX gate and then another U_1 gate applied to q_1, where the angle of this gate is equivalent to $2(\pi - x_{1_0})(\pi - x_{1_1})$. This gate is equivalent to the matrix $\left(\begin{smallmatrix} 1 & 0 \\ 0 & e^{i(2(\pi - x_{1_0})(\pi - x_{1_1}))} \end{smallmatrix}\right) = \left(\begin{smallmatrix} 1 & 0 \\ 0 & e^{i(2\pi(\pi-1))} \end{smallmatrix}\right)$. Figures 3.12b and 3.12c show the steps of replacing the angles with the data points and simplifying the form of the quantum circuit. 194

3.13 **Quantum circuits for the ZZ feature map.** These quantum circuits encode the data using the ZZ feature map. 194

3.14 **ReLU function.** The ReLU function returns the input if it is positive, otherwise it returns zero. 195

3.15 **Quantum circuits for random relu transform.** These quantum circuits encode the data using the random relu transform. The first step of the random relu transform is encoding the boolean data using basis encoding and then applying a random unitary matrix to the qubits. After the statevector of the quantum circuits are obtained, the ReLU function is applied to the statevector. 197

3.16 **Quantum circuits for amplitude encoding**. These quantum circuits prepare the statevectors in which the amplitudes are equivalent to the input data. (a) Encode the data $x_1 = 01$, so the statevector needs to be $0 \cdot |0\rangle + 1 \cdot |1\rangle = |1\rangle$. To get this statevector, we apply the $R_y(\pi)$ gate to the qubit, where $R_y(\theta) = \begin{pmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{pmatrix}$. Therefore, $R_y(\pi) = \begin{pmatrix} \cos(\pi/2) & -\sin(\pi/2) \\ \sin(\pi/2) & \cos(\pi/2) \end{pmatrix} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$. When we apply $R_y(\pi)$ to the qubit, we obtain $\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle$. Therefore the final statevector is $|1\rangle$ which is equivalent to encoding the data $x_1 = 01$ into the state. (b) Encode the data $x_2 = 10$, so the statevector needs to be $1 \cdot |0\rangle + 0 \cdot |1\rangle = |0\rangle$. Given that the qubits are initialised to $|0\rangle$, we do not need to apply any quantum gates to obtain this statevector. (c) The data is $x_3 = 11$, which we normalise to $x_3 = \frac{1}{\sqrt{2}} \frac{1}{\sqrt{2}}$ so the statevector needs to be $\frac{1}{\sqrt{2}} \cdot |0\rangle + \frac{1}{\sqrt{2}} \cdot |1\rangle$. To get this statevector we apply the $R_y(\frac{\pi}{2})$ gate to the qubit, where $R_y(\frac{\pi}{2}) = \begin{pmatrix} \cos(\pi/4) & -\sin(\pi/4) \\ \sin(\pi/4) & \cos(\pi/4) \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}$. When we apply $R_y(\frac{\pi}{2})$ to the qubit, we obtain $\begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} = \frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle$. Therefore this encodes $x_3 = 11$ into the state. 199

4.1 **Fitting data points with a model.** The black points are data points and the red line is a model. (a) A model that fits the data points exactly and has high variance. (b) A model that fits the data points poorly and has high bias. (c) A model that does not fit the data points exactly but fits the data points according to its pattern and has low bias and low variance. 236

4.2 **Bias-Variance U-shaped curve.** This curve shows how the test error changes as the complexity of the deep neural network changes. Figure inspired by Figure 1 in Reference (174). 240

4.3 **Double descent curve.** This curve shows how the test error changes as the complexity of the deep neural network changes. This double descent curve includes the U-shaped curve of the bias-variance tradeoff and includes deep neural networks with higher complexity, which leads to the double descent. Figure inspired by Figure 1 in Reference (174). 240

4.4 **Perceptron: a single-layer neural network.** The inputs x_i combined with the weights w_i are fed into a neuron, which calculates a linear function $z = \sum \vec{w} \cdot \vec{x}$ and then an activation function. The output of the neuron is y which is the result of the activation function calculation. Figure inspired by Reference (194). 255

4.5 **Neural network: a multi-layer perceptron.** The inputs x_i are fed into multiple neurons which calculate the linear and activation functions. The output of these calculations are then fed as inputs into the neurons in the next layer. The last layer of the neural network is the output layer. 257

5.1 **Quantum algorithms framework.** This Venn diagram distinguishes classes of quantum algorithms, including hybrid-quantum classical algorithms, variational quantum algorithms, and quantum neural networks. 267

5.2 **Types of quantum advantage.** Classifying types of quantum advantage into performance and resources, in which performance includes speed and accuracy, and resources includes cost and energy. 274

5.3 **Quantum advantage flowchart simplified.** Classifying types of quantum advantage by asking key questions and identifying the goals of certain classes of quantum advantages (a simplified version of Figure 5.4). The purple boxes are definitions, the pink diamonds are questions, and the black arrows are goals to move towards the orange star, which is a definition that embodies the ideal ultimate goal of quantum advantage. 285

5.4 **Quantum advantage flowchart.** Classifying types of quantum advantage by asking key questions and identifying the goals of certain classes of quantum advantages. The purple boxes are definitions, the pink diamonds are questions, and the black arrows are goals to move towards the orange star, which is a definition that embodies the ideal ultimate goal of quantum advantage. Figures 5.5, 5.6, and 5.7 display larger versions of parts of this flowchart as shown in the marked boxes in this figure. 286

5.5 **Quantum advantage flowchart starting from the problem.** 287

5.6 **Quantum advantage flowchart starting from Practical Quantum Advantage.** 288

5.7 **Quantum advantage flowchart starting from Nonpractical Quantum Advantage.** 289

6.1 **Probability of a boolean function $(\mathbf{f})$ versus its complexity for five data qubits and $\mathbf{10^5}$ samples.** $P(f)$ versus Lempel-Ziv complexity, K for a fully expressive QNN with (a) basis encoding (red), (b) ZZ feature map (blue), (c) random relu transform (pink), (d) amplitude encoding (orange), (e) a DNN (green), and (f) a non-fully expressive QNN with basis encoding (grey). $P(f)$ is calculated by generating 10^5 samples of functions from the QNN by using random samples of parameters Θ over a uniform distribution. 10^7 samples were generated from the DNN and 10^5 of the samples were randomly selected to match the scale of the QNN. 310

6.2 **Generalisation error versus Lempel-Ziv (LZ) complexity (top figures) and entropy (bottom figures) for five data qubits for a fully expressive QNN with different encoding methods.** The QNN is trained to zero training error on a training set S of size $m = 16$ and the generalisation error is calculated on the remaining 16 functions. The QNNs are trained by randomly sampling parameters 10^5 times and finding the generalisation error on the set of parameters that obtain zero training error. Error bars are one standard deviation. The data points show the error on the test dataset for different target functions with different LZ complexties and entropies. The maximum entropy is 16, as the boolean functions have length $2^5 = 32$ and entropy is the minimum number of 0s or 1s in the function. The green datpoint is the parity function. . . . 314

6.3 $\mathbf{u_1}$ **gate.** 340

6.4 $\mathbf{u_2}$ **gate.** The circuit for a general two-qubit rotation proposed in Reference (205). 342

6.5 $\mathbf{U_N}$ **gate.** Inspired by Reference (206). 345

6.6 **Probability to obtain a function of Lempel-Ziv complexity $\mathbf{K}$ versus its complexity for five data qubits and $\mathbf{10^5}$ samples.** $P(f)$ versus Lempel-Ziv complexity, K for a QNN with (a) basis encoding (red), (b) ZZ feature map (blue), (c) random relu transform (pink), (d) amplitude encoding (orange), (e) a DNN (green), and (f) a random learner (purple). $P(K)$ is calculated by generating 10^5 samples of functions from the QNN by using random samples of parameters Θ over a uniform distribution. 10^7 samples were generated from the DNN and 10^5 of the samples were randomly selected to match the scale of the QNN. 349

6.7 **Entropy of a boolean function (f) versus its complexity for five data qubits and $\mathbf{10^5}$ samples.** The colour of the data point corresponds to the maximum count of a function that has that complexity and entropy. Therefore, a data point with a colour that has a higher count indicates that the QNN has a stronger bias towards functions with that entropy and complexity. The boolean functions produced by the QNN were grouped and for each entropy-complexity pair, there were multiple functions, so the maximum count of a function is displayed. 350

6.8 **Probability of a boolean function versus its rank for five data qubits and 10^5 samples.** $P(f)$ ranked by probability of individual functions, generated from 10^5 samples of parameters Θ over a uniform distribution. The dotted line is Zipf's law $P(f) = \frac{1}{(32ln2)\text{Rank}(f)}$. 10^7 samples were generated from the DNN and 10^5 of the samples were randomly selected to match the scale of the QNN. 353

6.9 **Kernel matrices for different encodings for seven qubits ($n = 7$) resulting in 128×128 dimension matrices.** 356

6.10 **Kernel eigenvalues for different encoding methods for five qubits ($n = 5$) where 'x' is the index of the eigenvalue.** 356

6.11 **Probability of a boolean function (f) versus its complexity for five data qubits and 10^7 samples.** $P(f)$ versus Lempel-Ziv complexity, K for a quantum kernel with (a) basis encoding (red), (b) ZZ feature map (blue), (c) random relu transform (pink), and (d) amplitude encoding (orange). $P(f)$ is calculated by generating 10^7 samples of functions from the quantum kernel by using multivariate normal in numpy. 357

6.12 **Probability to obtain a function of Lempel-Ziv complexity $\mathbf{K}$ versus its complexity for five data qubits and $\mathbf{10^7}$ samples.** $P(f)$ versus Lempel-Ziv complexity, K for a quantum kernel with (a) basis encoding (red), (b) ZZ feature map (blue), (c) random relu transform (pink), (d) amplitude encoding (orange), (e) a DNN (green), and (f) a random learner (purple). $P(f)$ is calculated by generating 10^7 samples of functions from the quantum kernel by using multivariate normal in numpy. 358

6.13 **Probability of a boolean function versus its rank for five data qubits and $\mathbf{10^7}$ samples.** $P(f)$ ranked by probability of individual functions, generated from 10^7 samples from the kernel. The dotted line is Zipf's law $P(f) = \frac{1}{(32ln2)\text{Rank}(f)}$. 359

6.14 **Entropy of a boolean function (f) versus its complexity for five data qubits and $\mathbf{10^6}$ samples.** The colour of the data point corresponds to the maximum count of a function that has that complexity and entropy. The boolean functions produced by the quantum kernel were grouped and for each entropy-complexity pair, there were multiple functions, so the maximum count of a function is displayed. 361

6.15 **Generalisation error of a fully expressive QNN with different encoding methods for five data qubits. (a)** Generalisation error versus Lempel-Ziv complexity of the target function for a QNN with different encoding methods. The QNN is trained to zero training error on a training set S of size $m = 16$ and the generalisation error is calculated on the remaining 16 functions. The QNNs are trained by randomly sampling parameters 10^5 times and finding the generalisation error on the set of parameters that obtain zero training error. Error bars are one standard deviation. **(b)** Generalisation error versus entropy of the target function with the same specifications listed in (a). **(c)** Generalisation error vs probability of a target function for a QNN with different encoding methods. 10^5 samples of functions are generated from the QNN by using random samples of parameters. The probability of each function is calculated and its generalisation error is calculated. 365

6.16 **Generalisation error versus output function LZ complexity scatterplots for $\mathbf{n = 5}$ and for 20 random initialisations for three target functions.** The output LZ complexity is the complexity of the target function that the neural network produces after it has been trained to zero training error. A neural network that produces an output target function that is equivalent to the specified target function will have zero generalisation error and the same output LZ complexity as the specified target function. Each data point corresponds to one iteration. The dashed vertical line denotes the target function complexity. 368

6.17 **Generalisation error versus Lempel-Ziv complexity of the QNN with ZZ Feature Map vs DNN for $\mathbf{n = 5}$ for different target functions $\mathbf{T}$ training to zero error on a training set $\mathbf{S}$ of size $\mathbf{m = 16}$, with the error calculated on the remaining $\mathbf{|T| = 16}$ functions.** The target functions correspond to those in Table 6.6 and Table 6.7. An additional target function was added in Table 6.8. The networks were trained with the COBYLA optimiser on MSE loss for 20 random initialisations. Error bars are one standard deviation. The stars are the data points for the QNN with the ZZ feature map and the diamonds are the data points for the DNN. The legend indicates the LZ complexity (C) of the target function and the probability (P) of obtaining the target function in 10^6 samples. 370

6.18 **MSE loss and training error vs iterations for different target functions and different encoding methods for $\mathbf{n = 5}$.** The target function numbers correspond to those in Table 6.6 and Table 6.7. 373

6.19 **Bloch sphere for the $\mathbf{U_3}$ gate applied to one qubit with parameters sampled from different distributions.** Figures generated with Pennylane software (78) and inspired by Reference (206). 375

6.20 **General circuit diagram for a QCNN with four qubits.** Figure inspired by Figure 2 in Reference (210). 378

6.21 **Circuit for general two-qubit rotation.** Proposed in Reference (205). 378

6.22 **Restricted two-qubit rotation.** 379

6.23 **The prior probability $\mathrm{P(f)}$ versus LZ complexity K for QCNNs with different encoding methods and expressivity for $\mathrm{n} = 4$ and 10^4 samples.** . . . 381

7.1 **Overview of Quanto.** 390

7.2 **Example of a quantum circuit that can be optimised with Quanto.** 390

7.3 **The optimal circuit of the quantum circuit in Figure 7.2 for the given set.** 394

7.4 **Splitting the circuit into tiles of width 3 and height 2.** 395

7.5 **Steps to optimise the circuit when the circuit is split into tiles.** The boxes shaded in blue are the tiles. The boxes shaded in orange are the substitutions. 396

7.6 **Comparing Quanto to other compilers for optimising quantum circuits with known optimisations.** Some of these circuits are chosen from Reference (212). IBM and Tket backend is *ibmq essex*. 398

7.7 **Comparing Quanto to other compilers for optimising quantum circuits with known optimisations.** IBM and Tket backend is *Aerbackend*. 399

7.8 **Comparing Quanto to other compilers for optimising quantum circuits with known optimisations.** IBM and Tket backend is *Aerbackend*. 400

7.9 **Comparing Quanto to other compilers for optimising quantum circuits with known optimisations.** Some of these circuits are chosen from Reference (212). IBM backend is *ibmq lima* and Tket backend is *Aerbackend*. . . . 401

7.10 **Comparing Quanto to other compilers for optimising quantum circuits for unique gate sets.** Some of these circuits are chosen from Reference (215). 402

7.11 **Known circuit identity from Reference (216) where A and B are unitary if and only if V is of the form V = $\mathrm{R_z}(\alpha) \cdot \mathrm{R_y}(\theta) \cdot \mathrm{R_z}(\alpha) \cdot \mathrm{X}$.** 404

7.12 **Comparing Quanto to other compilers for optimising quantum circuits with unknown optimisations.** 406

7.13 **Comparing the compilation and execution time of the circuit compiled by Quanto versus by IBM.** 409

7.14 **Comparing the noise of the circuit compiled by Quanto versus compiled by IBM, using a noisy simulation of the IBM FakeVigo device.** 410

7.15 **(a) Quantum circuit which can be represented as (b) a matrix, which can be represented computationally as (c).** 421

7.16 (a) A possible circuit generated from a gate set. (b) The gate representation of the circuit. (c) The matrix calculation of the circuit to obtain the unitary matrix. 426

7.17 A quantum circuit can be represented by a unitary matrix. The hash of this unitary matrix is its fingerprint. 427

7.18 Permutations of single-qubit and two-qubit gates in $\mathrm{n} = 4$ circuit. 430

7.19 Dimensions of a tile compared to the dimensions of a quantum circuit. 433

7.20 All the possible tiles (shaded in blue) generated for a tile width and height = 2 for a quantum circuit with width and height = 3. 434

7.21 An example of an invalid tile (shaded in red), because the CX gate is cut off and is not first or last on its wire. 435

7.22 An example of a valid tile (shaded in blue), because even though the CX gate is cut off, it is first or last on its wire. 435

7.23 (a) The cut-off CX gate in the tile is 'replaced' with an Identity gate to make the tile searchable in the hash table of fingerprints (the CX is replaced back at a later stage). (b) The matrix representation of the tile is searched for in the hash table of fingerprints, which returns its fingerprint. (c) The fingerprint is used to find the circuit identities for the tile in the hash table of circuit identities. (d) The circuit identities for the tile. 437

7.24 Applying the substitution. 440

7.25 Example of a quantum circuit with a long-range two-qubit gate. 441

7.26 Splitting the circuit into tiles of width 5 and height 2. 442

7.27 Example of a quantum circuit with qubits reordered. 442

7.28 **Steps to optimise the circuit when the circuit is split into tiles.** The boxes shaded in blue are the tiles. The boxes shaded in orange are the substitutions. 443

A.1 **Pit profiles studied in the present work.** On each block of ore (square cells) the profit w_i is shown. Green (brown) blocks correspond to excavated (unexcavated) blocks in an optimal solution of the ultimate pit problem, and maximum profits are 5, 9, 113 and 21 respectively. The pit profiles are represented with systems of 4, 9, 12, and 12 qubits respectively. 490

A.2 **Evolution of the VQE negative cost function in the simulation of profiles in Figure A.1 (top to bottom), using different initial wavefunctions $|\Psi_0\rangle = |0\rangle^{\otimes n}$, $|1\rangle^{\otimes n}$ and $|+\rangle^{\otimes n}$ (respectively denoted unexcavated/excavated/superposition and shown in green/blue/red).** Black dashed lines denote maximum profit in optimal solutions of the problem. 502

A.3 **Evolution of the mining profile, i.e. the probability $p(z_i = 1)$ for each site i, at three different moments (marked a,b,c) of the optimisation.** Green (brown) denote excavated (unexcavated) sites. 504

A.4 **Evolution of the negative cost function of the *horizontal* fragments depicted in Figure A.5 using domain decomposition.** 505

A.5 ***Horizontal* cuts considered in fragmentation simulations.** 505

A.6 **Quantum circuit used to study the 4-site pit profile shown in Figure A.1a.** 506

A.7 **Evolution of the negative cost function during the variational optimisation.** Red, blue symbols denote raw and readout-error-mitigated results respectively, and the dashed horizontal black line denotes the optimal negative cost function value. Coloured bands denote a 67% confidence interval for the negative cost function samples after convergence of the optimisation. 507

A.8 **Probability distribution for the measurement of the four qubits of the register in Figure A.6 over the converged VQE wavefunction using a noiseless classical simulator of a quantum computer (green) and on quantum hardware with (blue) and without (red) readout error mitigation.** In all cases, the optimal open-pit configuration $(1, 1, 1, 1)$ is the most likely outcome of the measurement. 508

A.9 **Evolution of the negative cost function for the pit profile in Figure A.1c run on a selection of different optimisers with respect to evaluation number.** 513

A.10 **Fragmented pit profile.** 516

A.11 **Evolution of the negative cost function of the *other* fragments depicted in Figure A.12 using the domain decomposition procedure.** 518

A.12 ***Other* cuts considered in fragmentation simulations.** 518

List of Tables

2.1 **Levels of development for quantum hardware.** The levels of different quantum hardware platforms. Levels 1-4 examples are from Table 4 in Reference (21) by the *World Economic Forum, Global Future Council on Quantum Computing*. Level 0 is an addition for this work. 130

2.2 **Examples of organisations building different quantum hardware approaches**. 132

3.1 **Example of a Boolean function.** x consists of all the possible combinations of a binary string with length three. Each input x_i is mapped to an output $f(x_i)$, which is the Boolean function 00101001. 173

3.2 **ReLU function.** The ReLU function returns the input if it is positive, otherwise it returns zero. When applied to a complex number, we apply ReLU to the imaginary and real parts of the complex number. 196

4.1 **Example of a Boolean function.** x consists of all the possible combinations of a binary string with length three. Each input x_i is mapped to an output $f(x_i)$, which is the boolean function 00101001. 262

5.1 **Variational quantum algorithms vs quantum neural networks.** The difference between the problem and goal for these two types of quantum algorithms. The encoding method is the way to convert the classical input data into a quantum state (see Chapter 3 Section 3.4.1 for further details). 269

6.1 **Summary of inductive bias and expressivity produced by different encoding methods.** A low-entropy bias means that the neural network produces functions with a low entropy with a greater probability than functions with a high entropy. Similarly, a low-complexity bias means that the neural network produces functions with a low Lempel-Ziv (LZ) complexity with a greater probability than functions with a high LZ complexity. *Full expressivity* means that the QNN with that encoding method can express all of the possible Boolean functions. See Section 3.4.2.1 for more information about the expressivity of QNNs. For the $n = 3$ system, there are $2^{2^3} = 256$ possible Boolean functions so *Full expressivity* means that all 256 of these functions can be expressed. The expressivity of basis and amplitude encoding have been proven (see Section 3.4.2.2 and 6.7.1 respectively). The expressivity of the ZZ feature map and random relu transform were verified with 10^7 and 10^4 samples respectively from the QNN. 304

6.2 **Comparison of the methods for calculating the inductive bias in QNNs vs quantum kernels.** This table shows the difference between the input data encoding, output calculation, and function sampling tasks for QNNs and quantum kernels. 323

6.3 **Encoding the $\mathbf{n = 3}$ boolean dataset into two qubits using amplitude encoding.** If $\vec{x}$ is the dataset, the first column shows the x_i data elements. $x_0 = 000$ can not be encoded using amplitude encoding so its following columns are Not Applicable (N/A). The second column shows the process for encoding the data into the two qubits. The data is encoded into the two-qubit state as follows: $x_{i_0}|00\rangle + x_{i_1}|01\rangle + x_{i_2}|10\rangle + 0|11\rangle$, where x_{i_j} is the jth index in the x_i data element. The $|11\rangle$ state is encoded with 0 as it is not needed for the data encoding. The third column shows the simplified, normalised, encoded state. The fourth column is showing the statevector representation of the third column. The fifth column adds the readout qubit q_r to the statevector, which is achieved by applying the tensor product of the statevector of the data qubits $|\phi_i\rangle$ with the readout qubit $|q_r\rangle = |0\rangle$ so that the encoded statevector of the QNN with the readout qubit is $|\psi_i\rangle = |\phi_i\rangle \otimes |q_r\rangle$. 332

6.4 **The formulas and constraints used for the satisfiability tests.** The first column shows the Boolean dataset. The second column shows the simplified formula for the expectation value $\langle \sigma_{z_i} \rangle$ for the Boolean data x_i. The third column shows the parity function f and the fourth column shows the constraint that needs to be made on the formula in the second column in order to meet the value of the function in the third column. The $x_0 = 000$ data can not be encoded using amplitude encoding and therefore its columns are Not Applicable (N/A). 336

6.5 **Boolean input data for $\mathbf{n = 3}$.** 362

6.6 **Target functions for five qubits**. The first column is the number of the target function for identifying it. The second column is the symbol used for the target function. The third column is the target function itself and the fourth column is a description of the target function. 363

6.7 **Target functions for five qubits.** We show the LZ complexity (K) and the entropy (E) of the target function in the third and fourth columns respectively. We categorise the complexity and entropy in the fifth and sixth columns. For the following columns, we show the probability of obtaining the target function f from the QNN with the different encoding methods. 364

6.8 **Additional target function for five qubits.** This target function with LZ complexity = 35 is used in Figure 6.17. 371

7.1 **Gate error rates for the IBM Vigo device.** 411

7.2 **Inputs of the generation algorithm.** 420

7.3 **Possible circuits for single-qubit gates given gate set $[\mathbf{I}, \mathbf{X}, \mathbf{H}]$.** . . . 422

7.4 **Possible circuits for two-qubit gates given gate set $[\mathbf{I}, \mathbf{CX}]$.** 424

7.5 **Possible circuits for multiple layers.** 425

7.6 **Example of a hash table with one entry demonstrating the quantum circuit representation as the key and the fingerprint as the value.** Note that the fingerprint in this table is only for illustrative purposes (as the actual fingerprint is too long to be displayed). 427

7.7 **Hash table with one entry demonstrating the fingerprint as the key and the array of substitutions as the value.** 428

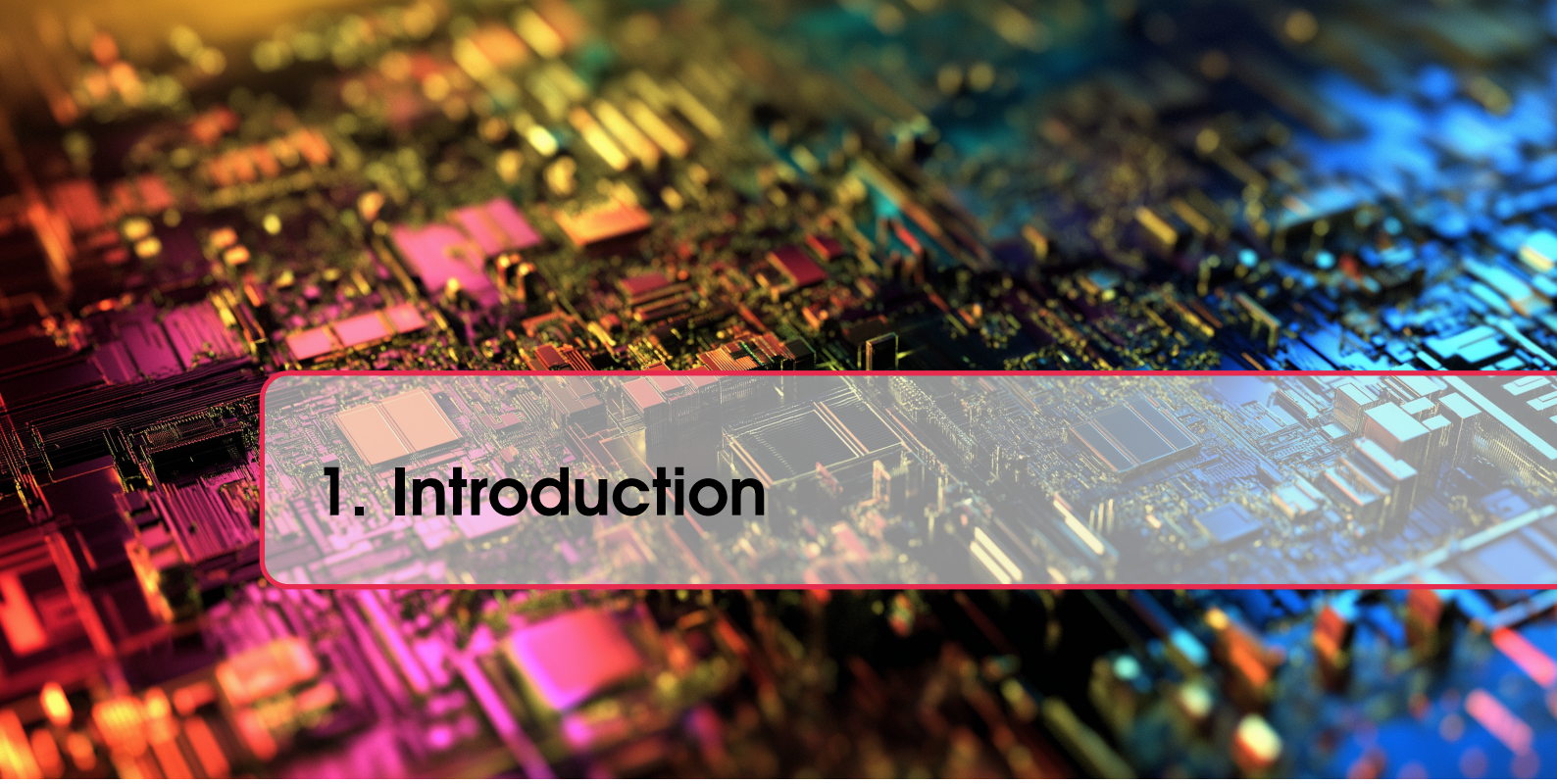
7.8 **Symbols for scaling formula.** 429

7.9 **Inputs to the optimiser algorithm.** 432

7.10 **Circuit identities and their cost.** 438

A.1 **Penalty parameter γ and probability of detecting an optimal solution of the pit profile problem $\mathbf{p}_{\text{opt}}$ for the calculations reported in Figure A.2.** Penalty parameters were initialised to $\gamma_0 = \max_i(w_i - \sum_{j \in P_i} w_j)/3$ and then varied manually by integer multiples of 1 until maximisation of p_{opt} was achieved. 503

A.2 **Penalty parameter γ and probability of detecting an optimal solution $\mathbf{p}_{\mathrm{opt}}$, for the calculations in Figure A.4 and A.11.** "h": *horizontal* cut, "o": *other* cut. 520



1. Introduction

1.1 Context 81

1.2 Framework for Quantum Algorithms 88

1.3 Thesis Structure 91

1.4 Contributions 94

1.5 Thesis Chapters 112

1.6 Publications 117

1.7 Computational Resources 119

1.1 Context

Quantum computing is an exciting field that uses the fascinating laws of quantum physics in computing devices to potentially solve real-world problems, which are challenges that affect people, communities, organisations, society, or the environment, and can make an impact on their everyday experiences. The concept of quantum computing emerged around the 1980s by merging the principles of quantum physics with the principles of computing, which had both developed independently decades earlier. Significant interest in the field of quantum computing sparked in the 1990s, after researchers such as Shor [1] and Grover [2] created novel quantum algorithms, which are a sequence of instructions to solve a problem on a quantum computer, in 1994 and 1996 respectively. These quantum algorithms gained attention because they have the potential to solve real-world problems faster than comparable classical algorithms. The most notable application of Shor's quantum algorithm to a real-world problem is breaking certain cryptography schemes, which are responsible for secure communication on the internet today. Grover's quantum algorithm could be applied to search problems in cryptography and optimisation problems, which occur in a variety of fields. The invention of such quantum

algorithms accelerated interest in the field as people realised that quantum computers could impact many industries, including cryptography, finance, machine learning, and chemistry [3] [4] [5].

At that time, quantum computing seemed to be just a theoretical concept. Yet, research groups endeavoured to build the building block of a quantum computer: the quantum bit (known as the qubit). Since 1988, several proposals for physically building a qubit were made [6, 7, 8, 9], but it wasn't until 1998 when these theoretical concepts became reality as the first quantum computer to run a quantum algorithm (Grover's algorithm) was built with two qubits [10]. Early quantum computers were just accessible by researchers in labs until 2016 when IBM made their five-qubit quantum computer publicly available to access through the internet [11]. Since then, larger quantum computers have been built and accessed by researchers worldwide. In 2024, quantum computers and simulators, which simulate the behaviour of the quantum computers on classical computers, are now accessible via major cloud platforms, including Microsoft Azure, Amazon Web Services, Google Cloud Platform, and IBM Cloud.

As quantum computing hardware matured, tools, software, and infrastructure emerged so that users could interface with the hardware effectively. Most notably, software, such as IBM Qiskit [12], were developed so that users could run quantum algorithms on the quantum hardware or simulate them on classical computers. In order to implement a user's quantum algorithm on quantum hardware, the hardware providers created quantum compilers which compiled the user's abstract quantum instructions to the hardware's specific quantum operations and connectivity. Several hardware-agnostic compilers emerged, most notably Quantinuum's Tket compiler [13], which allowed a user to run a quantum algorithm on multiple types of quantum hardware offered by various companies.

Though the ecosystem and infrastructure for quantum computing has expanded rapidly, the use of such quantum computers are still limited as fundamental theoretical and practical problems remain. The quantum computers that exist today have been coined *NISQ* (Noisy Intermediate-Scale Quantum) devices by John Preskill [14] because these devices are limited in terms of the number of qubits and the quality of the qubits and quantum gates, which are the basic operations of a quantum computer. Thus, the traditional quantum

algorithms that have been developed can not be run effectively on these NISQ devices. Since 2013, a number of *hybrid quantum-classical algorithms* have been invented [15] [16]; these algorithms dedicate parts of the algorithm to be solved on a quantum computer and other parts on a classical computer so that the quantum and classical resources work together to solve the problem. A significant subset of these algorithms are *variational quantum algorithms*, which use a classical optimiser to optimise a quantum algorithm to solve a problem. These algorithms were designed to be run on NISQ devices as they require fewer quantum resources (i.e. number of qubits and quantum gates) compared to the traditional algorithms and are designed to be robust to noise, which are disturbances in the system that lead to errors in the computation.

With the accessibility of NISQ devices, quantum software, and a framework for running quantum algorithms (i.e. variational quantum algorithms) on NISQ hardware, researchers could now try to find applications of quantum computers by solving a practical problem with a variational quantum algorithm and then simulating it on a classical computer and/or running it on NISQ devices. The first applications were finding ground states in quantum chemistry [15] and combinatorial optimisation problems [16], in which the aim is to find the optimal

solution from a finite set of possible solutions. Applications then expanded to compilation [17], linear algebra [18], machine learning [19], and other areas as described in Reference [20]. These technical problems could be applied to solve real-world problems in a variety of industries including materials, finance, logistics and transportation as outlined in Reference [21].

One notable application of variational quantum algorithms is *quantum machine learning*, a subfield of quantum computing which aims to solve machine learning problems with quantum computers and was developed years earlier based on the traditional quantum algorithms [22]. One of the main tools of quantum machine learning is *quantum neural networks*, which are quantum algorithms that aim to mimic classical neural networks, which are models inspired by the human brain, where we define a model as a program that has been trained to find patterns in data and make predictions based on the data. As early as 1995 [23], there have been proposals for quantum neural networks, but the development of variational quantum algorithms led to the standard framework of quantum neural networks we use today. In 2018, Farhi and Neven showed that these quantum neural networks could be used to find patterns in real-world data, consisting of handwritten digits, known as the MNIST dataset

[19]. Popular classical machine learning software libriaries, such as Tensorflow, included quantum neural networks in 2020, known as Tensorflow Quantum [24].

Researchers began to study the properties of these quantum neural networks in an attempt to understand them and find a *quantum speedup*, which occurs when a quantum algorithm can solve a problem faster than classical algorithms. Several researchers found limitations of these quantum neural networks, including most notably in 2018 the concept of *barren plateaus*, which means that certain calculations go to zero [25], which makes training them difficult. In 2019, Schuld and Killoran explored the theoretical link between quantum neural networks and kernel methods [26], which are another class of machine learning algorithms, and proposed strategies inspired by kernel methods to find patterns in data more efficiently with quantum computers. In 2020, Liu et al. used these kernel methods to make the first rigorous claim of a quantum speedup for *supervised quantum machine learning* [27], a category of quantum machine learning that uses labeled datasets to train the algorithms to find patterns and make predictions on the data. Since then, there have been other claims of quantum speedups using kernel methods for specific problems [28] and empirical claims of quantum speedups for small-scale problems using quantum

neural networks [29], but the latter was later shown to be incorrect [30].

Although there are a couple of non-practical specific problems that can be solved with a quantum speedup with quantum supervised machine learning algorithms [27, 28], we still don't know where quantum neural networks will provide a *practical quantum advantage*, which occurs when a quantum algorithm is more advantageous for solving a *practical* problem compared to classical algorithms. One of the reasons for this lack of knowledge is because finding a quantum advantage in machine learning is challenging as quantum machine learning algorithms can only operate on a small-scale, whereas classical machine learning algorithms currently operate on a very large-scale, thereby making it unattainable to make any substantial empirical claims of quantum advantage. Another challenge is that the theory of classical machine learning is lacking in some places, which means that claims of successful algorithms are evaluated empirically, whereas quantum algorithms have historically been evaluated theoretically; as some parts of classical and quantum machine learning lack this theoretical basis, it is challenging to make theoretical claims of quantum advantage.

Questions still remain on whether we can use hybrid quantum-classical algorithms, including quantum neural networks, to find useful applications of quantum computers with NISQ devices. The goal of this thesis is to contribute to answering this question by investigating certain aspects of these algorithms: their foundation, optimisation, and application.

1.2 Framework for Quantum Algorithms

This thesis covers multiple topics within *quantum computing* and *quantum machine learning*, with a particular focus on *quantum neural networks*. To solve problems with quantum computing, one uses a *quantum algorithm*, which is a sequence of instructions to obtain the solution to a problem. I have created a framework to categorise the main areas of quantum algorithms research. This framework is a structure of how to work with quantum algorithms and consists of three conceptual layers, inspired by the technical structure of quantum neural networks, which consists of layers (this is described in the *Quantum Neural Networks* chapter - Chapter 3 in Section 3.4). These conceptual layers are different areas of research into quantum algorithms and build on top of each other, as depicted in Figure 1.1. The layers are as follows:

1. **Foundation Layer**: aims to understand the foundation of quantum algorithms, which includes their theory and properties.
2. **Optimisation Layer**: aims to optimise the performance of quantum algorithms, which includes the development of tools, techniques, and software.
3. **Application Layer**: aims to find applications of quantum algorithms.

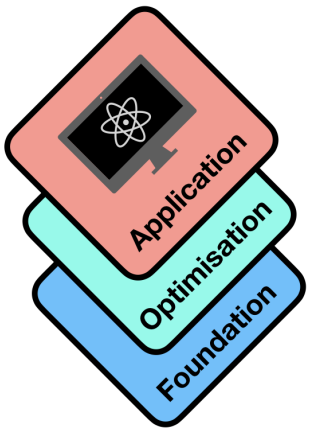


Figure 1.1: **Conceptual layers of quantum algorithms explored in this thesis.** From the foundation, to optimisation, to application of quantum algorithms.

As depicted in Figure 1.1, the bottom layer is the foundation layer. It is possible to build the other two layers (the optimisation and application layers) without this foundation layer, but a stronger foundation layer provides a stronger foundation for the other two layers. For example, a stronger understanding

of the theory and properties of certain quantum algorithms could help us create techniques to optimise their performance and find useful applications of quantum algorithms with quantum advantage.

The middle layer is the optimisation layer. It is possible to build the top layer (the application layer) without this optimisation layer, but a stronger optimisation layer provides a stronger foundation for the top layer. For example, a better understanding of optimisation could help us find more applications and improve the performance of existing applications.

The top layer is the application layer, which is the most visible layer, both literally (as it is on the top) and metaphorically, as it is the layer that the public would see, as this layer moves beyond research and moves into the realm of real-world problems, affecting individuals and society.

It should also be noted that it is possible for higher layers to affect and improve lower layers. For example, if we find new applications of quantum algorithms empirically, it may help us update the foundation layer of theory. In fact, this is currently happening in deep learning, a subset of classical machine learning

using deep neural networks, where its empirical performance and success has led to new theories as the previous theories were not matching their empirical behaviour (this is described further in the *Classical Neural Networks* chapter - Chapter 4).

I believe that most of the research in quantum algorithms can be categorised into these three conceptual layers and this framework is helpful for understanding the research going on in the field and how these areas fit into each other.

1.3 Thesis Structure

My thesis aims to cover the three conceptual layers of quantum algorithms covered in the *Framework* section (Section 1.2), by exploring particular problems in each layer:

1. **Thesis Foundation Layer**: aims to understand two fundamental properties, the inductive bias and expressivity, of a quantum neural network.
2. **Thesis Optimisation Layer**: aims to optimise a quantum circuit by developing a quantum compiler using new techniques.

3. **Thesis Application Layer**: aims to find a new real-world application of a quantum computer by constructing a quantum algorithm to solve an optimisation problem in the mining industry.

At the end of each chapter that presents the research for each layer, I show how the layers fit together and how the intersection of the research in these chapters leads to new interesting research questions that could be explored in future work.

1.3.1 Foundation of Quantum Neural Networks

To explore the ***foundation*** of ***quantum algorithms***, this thesis focuses on two foundational properties of a particular type of quantum algorithm, called a ***quantum neural network***. The two properties we investigate are the inductive bias (the preferences of the algorithm) and the expressivity (how many functions it can express). Having an inductive bias is essential for any learning algorithm, for that is what enables it to be useful to make predictions for unseen data. Therefore, understanding the inductive bias of quantum neural networks can help us understand how quantum neural networks learn and where

they might have an advantage or disadvantage over classical neural networks, especially given that classical neural networks are widely used and work well for a variety of problems.

1.3.2 Optimisation of Quantum Circuits

To explore the ***optimisation*** of ***quantum algorithms***, this thesis focuses on optimising ***quantum circuits***, which consist of a sequence of quantum gates to implement quantum algorithms. One major stumbling block to quantum algorithms is their performance on quantum hardware, as the deeper a quantum circuit is (i.e. the number of steps from the beginning to the end), the more errors are introduced and the worse it performs. Therefore, in order to practically use quantum algorithms, we need to understand how to optimise them to give the optimal results on NISQ devices. This optimisation technique is also a first step for optimising the performance of quantum neural networks, as they are also implemented with quantum circuits.

1.3.3 Application of Quantum Computers

To explore the ***application*** of ***quantum algorithms***, this thesis uses a particular type of quantum algorithm, called a *variational quantum algorithm*, to solve a real-world optimisation problem on a ***quantum computer***. The problem is called the *ultimate pit problem*, which aims to find an optimal design for open-pit mines. Because quantum computers work fundamentally differently from classical computers, it is not straightforward to map a classical problem onto a quantum computer. This research project provides an approach to map this classical problem onto the quantum algorithm. We ran the algorithm on NISQ devices and classical simulators.

1.4 Contributions

1.4.1 Objective

In this section, I show how this thesis contributes to the field of quantum computing. In order to do this, I provide additional background context and demonstrate how this thesis fits into the wider context provided in the *Context* section (Section 1.1). I summarise the research my coauthors and I completed

in order to address the research questions arising from the context and then I summarise the results of the research. I then explain how this thesis contributes to the field, in light of the background context, the research performed, and the research results. This thesis is split into three areas: foundation, optimisation, and application. In this section, I provide the context, summary of work, summary of results, and contributions to the field for each area.

1.4.2 Foundation of Quantum Neural Networks

1.4.2.1 Context

Towards the end of the *Context* section (Section 1.1), we saw that specific cases of quantum speedups using kernel methods emerged in 2020 [27, 28], yet we still don't know where quantum neural networks will provide a quantum advantage. Little is understood on how quantum neural networks perform generally in comparison to classical neural networks as highlighted by Schuld and Killoran in Reference [31]. They note that "good performance on selected hand-crafted examples therefore does not tell us anything about quantum computers as general learning tools." To understand how quantum computers perform as general learning tools, we can review the understanding of how

classical computers perform as general learning tools.

We know that classical computers have been highly successful as general learning tools; in particular, deep neural networks, which are neural networks with many layers, became more successful in practice and became widely used in applications, such as ChatGPT, which has gained hundreds of millions of users [32]. Despite its success, there were questions about the theoretical performance of deep neural networks, in particular because their performance was not predicted by the standard theory of these learning tools. A popular concept, known as the *bias-variance tradeoff*, which describes the relationship between a model's complexity and error, suggested that the performance of deep neural networks would be poor as the model became more complex. As a result, a suggestion to rethink the theory of these models was made most notably by Zhang et al. in 2016 [33]. Since then, several hypotheses have emerged as to why deep neural networks perform well despite their higher complexity. One hypothesis comes from Valle-Perez et al. [34], who argue that deep neural networks have a bias (i.e. a preference) towards functions, which map the input to the output of the neural network, that are simple and this enables them to predict well on real-world data, as such data is structured (i.e.

simple). Their argument originally came from an adaptation of *algorithmic information theory*, where it was shown that many input-output maps have this bias, called *simplicity bias* [35].

Therefore, one hypothesis for why deep neural networks perform well as general learning tools is because of their simplicity bias. After I read these papers on deep neural networks and simplicity bias, I thought that looking at the bias of quantum neural networks could give us insight into their performance as general learning tools, thereby contributing to a richer understanding of this unanswered question in the field. The question I wanted to answer was "Do quantum neural networks have simplicity bias?" [36]. An answer to this question would give us a clue into their performance as general learning tools.

1.4.2.2 Summary of Work

To answer this research question, we investigated the bias of quantum neural networks. We learned that the expressivity of the quantum neural network (i.e. how many functions it can express) is an important factor in answering this research question, so we investigated the expressivity of the quantum neural networks as well. One of the main components of a quantum neural

network that affects the bias and expressivity is the encoding method, which is the way the data is put into the quantum neural network. Therefore, we constructed various quantum neural networks with different encoding methods and different levels of expressivity. We adapted the method of finding the bias of deep neural networks to quantum neural networks. We ran experiments to understand how the bias changes when the encoding method and expressivity of the quantum neural network change. We also came up with proofs to verify that a certain encoding method for random quantum neural networks has no bias and a certain encoding method limits the expressivity of the quantum neural network. We performed other experiments to show how the encoding method and bias affects the performance of the quantum neural network. We also looked at the bias of quantum kernels, another class of quantum machine learning algorithms, and the bias of quantum convolutional neural networks, a type of quantum neural network.

1.4.2.3 Summary of Results

Our results show that it is possible to have a simplicity bias by using a certain encoding method, as the encoding method affects the bias of the quantum

neural networks. We show that this encoding method reduces the expressivity of the quantum neural network. We show that other encoding methods we studied either have no bias or a trivial bias. Our results also show that the performance of the quantum neural network corresponds with the strength of the bias. We also demonstrate that an artificial bias can be produced by intentionally restricting the expressivity of the quantum neural network. We show that quantum kernels have a similar bias to the quantum neural networks we studied. Our results suggest a tradeoff between the expressivity and the bias of the encoding method.

1.4.2.4 Contributions to the Field

Classical deep neural networks have both a good bias and high expressivity, which are two key properties hypothesised to be behind their success. Our results suggest that it would be difficult to design quantum neural networks, using the standard framework, that would be at least as good as deep neural networks because of our observed bias-expressivity tradeoff. As it has been suggested that the reason deep neural networks perform well on real-world data is because of their simplicity bias, the lack of simplicity bias and high

expressivity in quantum neural networks suggest that they may not be good general-purpose learning tools on real-world data.

Moreover, our work highlights that quantum neural networks need knowledge about the properties of the data to perform well, also suggested by other papers in the literature [37], which is a limitation compared to deep neural networks that have an inherent bias towards simple functions and don't need special knowledge about the properties of the data. Our work contributes clearer reasons posed to the field as to why alternative frameworks for quantum neural networks should be explored, as also suggested in Reference [31].

Our work also helps researchers know what they need if they want to design a general-purpose quantum neural network; in particular, researchers could try to discover new encoding methods with the properties we have suggested (a simplicity bias and high expressivity) and they can use our methods to evaluate their quantum neural network designs.

Schuld and Killoran in Reference [31] noted that "good performance on selected hand-crafted examples therefore does not tell us anything about quantum

computers as general learning tools." Our work contributes to a better understanding of quantum computers as general learning tools, in particular that they are not good general learning tools compared to deep neural networks because of our observed bias-expressivity tradeoff.

Schuld also highlighted other unaddressed questions for quantum neural networks in Reference [38] including "what makes quantum models, supervised and unsupervised, be flexible enough to obtain a low training error, but simple enough to not overfit?" To unpack this question, supervised models are ones that use labeled datasets and unsupervised models are ones that use unlabeled datasets. The training error portrays how well the quantum model can fit the training data and overfitting implies that the quantum model fits all of the fluctuations and particularities of the training data, which would make it not perform well on other unseen data. Our work contributes to a better understanding of this question, as we show that it is the bias that can make quantum models flexible enough to obtain a low training error but simple enough to not overfit, based on the hypothesis in deep neural networks.

1.4.3 Optimisation of Quantum Circuits

1.4.3.1 Context

As mentioned in the *Context* section (Section 1.1), tools, software, and infrastructure developed as quantum computing hardware matured. One vital tool is a quantum compiler that compiles a user's abstract set of instructions to a physical device. The first quantum compilers that appeared were developed by hardware providers, such as IBM and Google, as they needed to create quantum compilers to allow users to interact with their hardware. Later on, independent organisations developed quantum compilers that were agnostic to the hardware, so the compiler could compile a user's instructions to multiple types of hardware.

The main goal of the compiler was to perform this algorithm-to-device mapping, but some of them also tried to optimise the quantum circuit, which is the sequence of quantum gates to implement a quantum algorithm, by relying on a specific set of rules to replace quantum gates that are equivalent. There are several measures to determine whether a quantum circuit has been optimised well, but one effective measure is the depth of the quantum circuit, which is

the number of time steps from the beginning to the end. If the depth of the quantum circuit is smaller, the time steps are fewer, and generally the less time the computation will take and the fewer errors that are introduced.

Thereby, a quantum compiler could try to reduce the quantum circuit depth so that the quantum circuit could run more efficiently on the hardware. Such a quantum compiler is especially useful for NISQ devices, where its resources are limited, and reducing the resources needed (e.g. the number of quantum gates) to run an algorithm could help make solutions more attainable with these NISQ devices. In addition, the noise is high on NISQ devices and reducing the quantum circuit depth could reduce the noise, making the solutions more accurate.

My coauthors and I wanted to create a better quantum compiler, inspired by successful techniques in classical compilers [39]. We noticed that more complex *quantum circuit identities* (quantum circuits that are equivalent to each other and therefore could replace each other) were not being used in the previous quantum compilers because only a set of simple rules were being used for quantum circuit identities and therefore the quantum circuit opimisations

were sub-optimal. We published a new quantum compiler that automatically generates circuit identities and optimises quantum circuits with them [40].

1.4.3.2 Summary of Work

To create this new type of quantum compiler, we created a program that generates circuit identities that takes as inputs a set of gates, number of qubits, and depth of the circuit identities, and generates as output a database consisting of all circuit identities for the inputs specified. We then created a quantum circuit optimiser that takes as input the user's quantum circuit to be optimised and produces an optimised circuit. To produce this optimised circuit, we developed a tiling method to break down a quantum circuit into tiles if the quantum circuit is larger than the circuit identities generated; we then optimise the tiles within the quantum circuit using the circuit identities generated, leading to the entire quantum circuit being optimised. We derived a scaling formula for our algorithm and ran experiments to compare our optimiser's performance to other quantum compilers. We also ran experiments to observe the reduction in noise and execution time as a result of the optimisation.

1.4.3.3 Summary of Results

Our results showed that our quantum compiler could lead to better performance (i.e. better optimisation by reducing the quantum circuit depth) on certain quantum circuits compared to existing compilers. In addition, our results showed that we could generate novel circuit identities and circuit identities for custom gates. Our results also showed that better optimisation means that quantum circuits could run faster and with fewer errors.

1.4.3.4 Contributions to the Field

Our work opened up a new method of quantum compilation: instead of just seeing compilers as a way to map quantum circuits to quantum hardware, quantum compilers could just focus on optimising the abstract quantum circuits. Before our work, circuit identities were found manually, but we provided a way to generate circuit identities automatically for multiple-qubit gates, which also led to a database of novel circuit identities; these novel circuit identities can be used as rules for other quantum compilation systems. Furthermore, as previous compilers were based on a set of rules for a specific set of gates, unique gates could not be compiled by their compilers, which was restricting the scope of

gates that could be optimised. Our compiler allows for custom gates to be optimised, meaning that it can be applied for a broader range of quantum circuits and quantum devices. In addition, our compiler allows for customised cost functions, meaning that the compiler can optimise a quantum circuit by taking into account a variety of metrics. In this work, we use the depth of the quantum circuit as the cost, but one could also use, for example, the number of quantum gates, the number of two-qubit gates, the quantum gate errors, or a combination of them. By using quantum gate errors as the cost, our compiler could also serve as an explicit error mitigation technique. This is in contrast with compilers before our work, which rely on a fixed cost function, as they use manual circuit identities (which are made according to some predetermined cost function) for the optimisation. Therefore, our work contributes a new way of optimising quantum circuits by leveraging custom cost functions. Cost functions that combine a variety of metrics (such as the number of quantum gates and their errors) could lead to better optimisations. Furthermore, as our optimisation results in quantum circuits that could run faster and with fewer errors, our compiler helps make the possibility of quantum applications more likely, as the speed and accuracy of the solutions are some of the bottlenecks

for quantum applications.

1.4.4 Application of Quantum Computers

1.4.4.1 Context

As mentioned in the *Context* section (Section 1.1), researchers started to use variational quantum algorithms to discover new applications of quantum computers [20]. My coauthors and I wanted to solve a real-world problem with these algorithms that hasn't been done before. A couple of my coauthors were working at IBM and found the 'ultimate pit problem', which is a combinatorial optimisation problem in the mining industry to find the optimal design of an open-pit mine, which is an excavation at the surface of the ground for the purpose of extracting valuable materials from the earth. The optimal design of the open-pit mine involves choosing which blocks or sites of the mine should be dug or not.

Mining is the extraction of minerals or materials from the earth. The ore, which is the natural rock or sediment, extracted from the earth may contain valuable materials, such as gemstones or metals, including gold, iron, coal,

copper, diamonds, granite, marble or silver. Open-pit mining is a surface mining technique that extracts such materials from an open pit in the ground.

The process, however, can be expensive and involves large risks. Before taking on such a project, it is important to know what ore can be extracted and how much of the ore should be extracted in order to make the entire project profitable. Before the digging of the mine, it is possible to estimate the amount of valuable materials and their location by the geological interpretation of information obtained from drill cores. The purpose of open-pit design is to determine the amount of ore to be mined. Given the information about the concentration of ores and impurities at each point, the problem is to decide which sites of the pit to dig - in other words the 'design' or 'contour' of the pit. The goal is to find the optimised contour that would minimise the total cost of mining. This leads to lower costs, higher profits, and increased safety and protection for the environment and mine workers.

In 1965, the (classical) Lerchs-Grossman (LG) algorithm [41] was developed to solve this problem and was commercialised 20 years later as Whittle by Dassault Systèmes, which provides this solution for its customers to plan their

mining process. Ten years later, it became a mainstream approach for open-pit design optimisation [42]. The Pseudoflow algorithm was later developed as a more computationally efficient version of the LG algorithm and is also commercialised by several companies, including Deswik [43]. My coauthors and I were interested in using a quantum computer to solve a small-scale version of this real-world problem [44].

1.4.4.2 Summary of Work

To solve this combinatorial optimisation problem, we constructed a Hamiltonian, equivalent to the total energy of the system, so that the lowest energy state of the Hamiltonian (i.e. its ground state), would correspond to the optimal open-pit mine design and thus the solution to the ultimate pit problem. We also constructed a quantum ansatz, which is a design of a parametrised quantum circuit; this is a quantum circuit that has gate parameters that can be changed. Our quantum ansatz mimics the ultimate pit problem, using the principles of superposition to provide a unique approach to the problem; this means the pit blocks are in a superposition of 'dug' and 'undug' in the quantum algorithm. My coauthors also created a technique to decompose a larger-scale version of

the problem so that it could be solved on small-scale hardware. After that, my coauthors and I simulated our quantum algorithm on classical computers to find the solution to the ultimate pit problem for different open-pit mines and ran it on quantum hardware using four qubits. We used error mitigation techniques to reduce the hardware errors. We compared the quantum algorithm's solutions to the classical Pseudoflow solution.

1.4.4.3 Summary of Results

Our results showed that our quantum algorithm could find the optimal solutions to the problem instances we ran, even on the quantum hardware and with the decomposition technique.

1.4.4.4 Contributions to the Field

Our work was the first attempt, to the best of our knowledge, to solve the ultimate pit problem, a valuable problem in the mining industry, on a quantum computer. We also believe it was the first attempt to apply quantum computing to the field of computational mining engineering. Therefore, our work represents the beginnings of understanding and establishing the impact of quantum

computing to the field of mining engineering, a useful endeavour for both communities. Our research has opened up a new application of quantum computers, which could be the first step to using quantum computers in the mining industry. The decomposition technique could also be adapted to other problems which require a number of qubits larger than the number of qubits available in quantum hardware today. Furthermore, our work provided an approach to map such a problem to a quantum computer and this approach could be adapted to map other real-world problems to a quantum algorithm.

1.4.5 Categories of Quantum Advantage and Algorithms

In addition to the research chapters, Chapter 5 also makes contributions to the field with frameworks. One of these frameworks clarifies the categorisations of quantum algorithms used in this thesis. I have also constructed a framework, through the form of a flowchart, to categorise different types of quantum advantage.

1.5 Thesis Chapters

1.5.1 Chapter 1: Introduction

Chapter 1 provides an overview of the context for this thesis; more in-depth context is provided in Chapters 2, 3, and 4. This chapter introduces a framework for the areas of quantum algorithms research and outlines how the structure of the thesis fits into this framework. This chapter also provides a summary of additional context for Chapters 6, 7, and Appendix A, and provides a summary of the work done, the results, and contributions for the thesis. This chapter outlines the chapters in this thesis, lists the publications during my DPhil, and states the computational resources used.

1.5.2 Chapter 2: Quantum Computing

Chapter 2 provides a brief history on computing, quantum physics, quantum computing and key developments in the field, including quantum algorithms, quantum hardware, quantum cloud platforms, quantum software, quantum tools, quantum applications, quantum advantage, NISQ and fault-tolerant devices, variational quantum algorithms, and quantum machine learning. These key

developments in the broader field of quantum computing have influenced the direction of quantum machine learning and quantum neural networks. This chapter provides relevant context for why the research in this thesis has been done (e.g. because of the unanswered questions we uncover in quantum machine learning) and also why the research in this thesis can be done in this period of time (e.g. because of the quantum software that enables efficient simulations of quantum circuits).

1.5.3 Chapter 3: Quantum Neural Networks

Chapter 3 lays the groundwork on quantum neural networks, including a brief history. This chapter presents the technicalities of how quantum neural networks work, including explanations of encoding methods, variational quantum circuits, expressivity, measurements, and the learning process. This chapter provides a history of the intersection of kernel methods and quantum neural networks and describes the technicalities of quantum kernels. This chapter includes an overview of the research on quantum neural networks, including some of the seminal papers in the field. This chapter also describes quantum advantage and how it can be obtained with quantum machine learning algorithms.

1.5.4 Chapter 4: Classical Neural Networks

Chapter 4 lays the groundwork on classical neural networks. This chapter provides a brief history and the shift of understanding in the field, based on the empirical performance of deep neural networks that defy the standard theory of errors in models; this chapter explains the bias-variance tradeoff. This chapter introduces the concept of inductive bias and simplicity bias in deep neural networks. This chapter outlines algorithmic information theory and its connection to simplicity bias. This chapter describes the technicalities of how classical neural networks work, including the expressivity of neural networks and generalisation performance. This chapter gives a brief explanation of the boolean system which can enable us to study inductive bias. As quantum neural networks mimic the structure and principles of classical neural networks, an understanding of the inner workings of classical neural networks is helpful.

1.5.5 Chapter 5: Categories of Quantum Advantage and Algorithms

Chapter 5 clarifies the different categories of quantum algorithms explored and used in the subsequent research chapters in this thesis. This chapter also presents a framework for categorising different types of quantum advantage.

1.5.6 Chapter 6: Foundation of Quantum Neural Networks

Chapter 6 presents the research we did to address unanswered questions in the field that arise from the context given in the *Context* section (Section 1.1). We investigated the inductive bias and expressivity of different classes of quantum neural networks and how that affects their generalisation performance, which is how accurately an algorithm is able to predict outcome values for previously unseen data.

1.5.7 Chapter 7: Optimisation of Quantum Circuits

Chapter 7 presents the research we did on quantum optimisation. At the end of this chapter, we connect the research done in this chapter to the previous

chapters, which leads to future research questions. We developed a method to optimise quantum circuits according to a custom metric, such as the depth of the quantum circuit; reducing the depth of the quantum circuit can reduce errors in the final solution and thereby enable more accurate results. The results showed that we are able to reduce the depth of certain quantum circuits more than existing quantum compilers because of our novel technique. This chapter demonstrates one of the methods that can be used to improve variational quantum algorithms and quantum neural networks.

1.5.8 Chapter 8: Conclusion

Chapter 8 presents a summary of the thesis and a future outlook on the research presented in this thesis, including new research questions that could be explored in follow-up work.

1.5.9 Appendix A: Application of Quantum Computers

Appendix A presents the research we did to construct a quantum algorithm for a mining optimisation problem. We demonstrated how to map this classical problem to a quantum circuit and how to solve the problem using variational

quantum algorithms. The results showed that we were able to find the optimal solution for the inputs we used.

1.6 Publications

My work has led to the publication of research papers and articles, which are listed below. Chapter 6 of this thesis is based on Paper 1, Chapter 7 is based on Paper 2, and Appendix A is based on Paper 3.

1. Pointing, Jessica. "Do Quantum Neural Networks have Simplicity Bias?" arXiv preprint (2024). https://doi.org/10.48550/arXiv.2407.03266.
2. Pointing, Jessica, Oded Padon, Zhihao Jia, Henry Ma, Auguste Hirth, Jens Palsberg, and Alex Aiken. "Quanto: Optimizing quantum circuits with automatic generation of circuit identities." Quantum Science and Technology (2024). https://doi.org/10.1088/2058-9565/ad5b16.
3. ($*$ = co-first authors, listed alphabetically) Hindy, Yousef*, Jessica Pointing*, Meltem Tolunay*, Sreeram Venkatarao*, Mario Motta, and Joseph A. Latone. "Application of the Variational Quantum Eigensolver to the

Ultimate Pit Problem" IEEE International Conference on Quantum Computing and Engineering (2023) https://doi.org/10.1109/QCE57702.2023.00083.

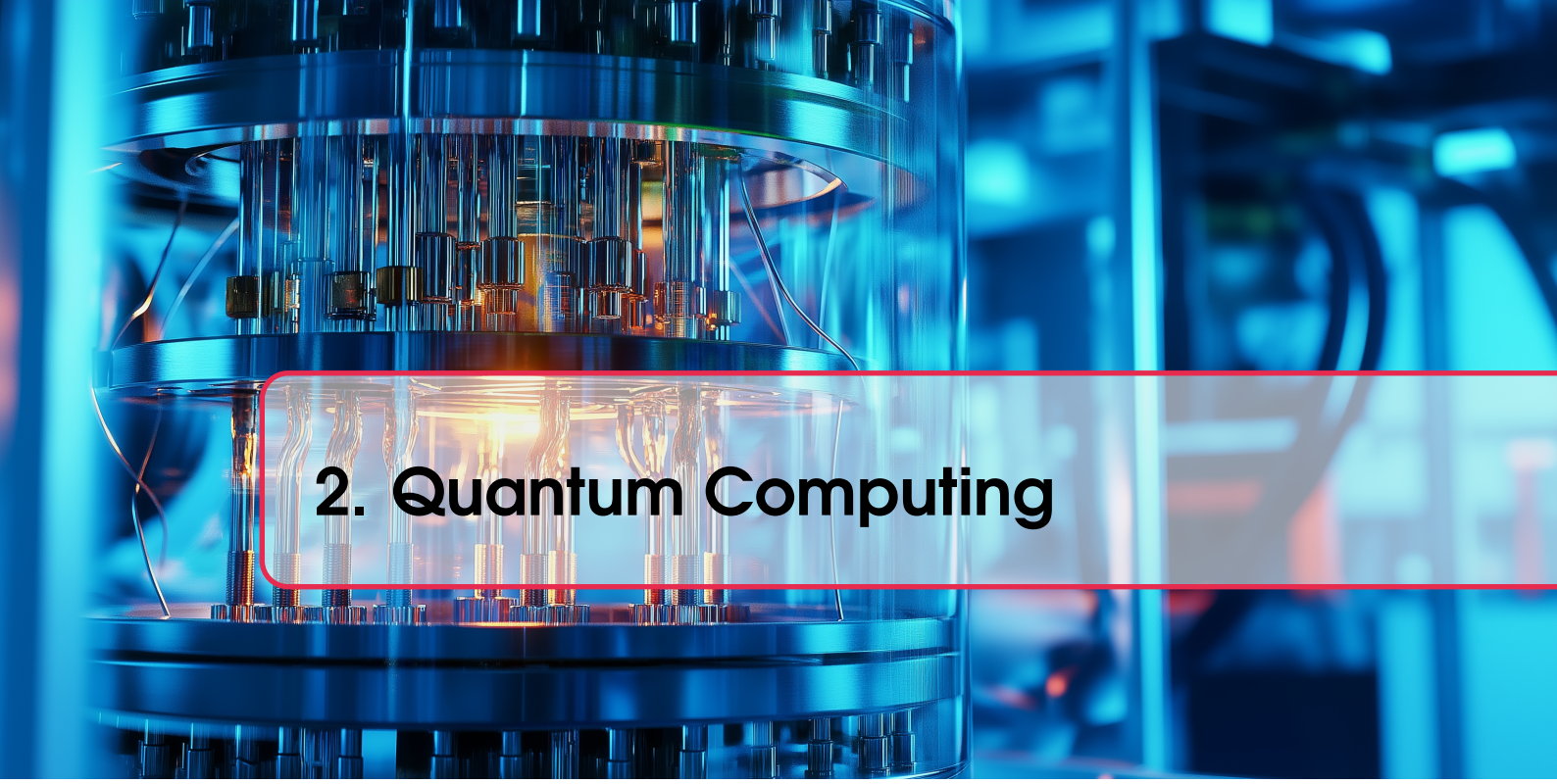
4. (∗ = lead authors, listed alphabetically) Chow, Jerry*, Eliška Greplová*, Freeke Heijman*, Carlos Kuchkovsky*, Derek O'Halloran*, Jessica Pointing*, Grigory Shutko*, and Carl J. Williams*. "State of Quantum Computing: Building a Quantum Economy." World Economic Forum, 2022. https://www.weforum.org/publications/state-of-quantum-computing-building-a-quantum-economy/.

5. Xu, Mingkuan, Zikun Li, Oded Padon, Sina Lin, Jessica Pointing, Auguste Hirth, Henry Ma et al. "Quartz: superoptimization of quantum circuits." In Proceedings of the 43rd ACM SIGPLAN International Conference on Programming Language Design and Implementation, pp. 625-640 (2022) https://doi.org/10.1145/3519939.3523433.

6. Mingard, Chris, Jessica Pointing, Charlie London, Yoonsoo Nam and Ard Louis. "Exploiting the equivalence between quantum neural networks and perceptrons" arXiv preprint (2024). https://doi.org/10.48550/arXiv.2407.04371.

1.7 Computational Resources

Numerical experiments in this thesis involved the simulation of quantum circuits. I wrote code to simulate the operations of a quantum computer (i.e. a software quantum simulator) to run quantum circuits on a classical computer. I also used the Qiskit simulators [45], including the *Qiskit statevector simulator*, which simulates the vector and matrix operations in the quantum circuit and produces the final statevector, and the *QASM simulator*, which draws samples from the distribution generated by the quantum circuit, which more realistically mimics the process of sampling from the quantum hardware. Large simulations were run on the Hydra computing cluster at the University of Oxford [46]. Experiments involving quantum hardware were accessed through the IBM Quantum Computing Platform [47], which allows public open access to their quantum computers.



2. Quantum Computing

2.1 Preview 121

2.2 Computing 122

2.3 Quantum Physics 124

2.4 Quantum Computing 125

2.5 Quantum Algorithms 125

2.6 Quantum Hardware 126

2.7 Quantum Cloud Platforms 133

2.8 Quantum Software 135

2.9 Quantum Tools 136

2.10 Quantum Applications 137
2.11 Quantum Advantage 138
2.12 NISQ and Fault-Tolerant Devices 139
2.13 Variational Quantum Algorithms 141
2.14 Quantum Machine Learning 146
2.15 Future Directions 152
2.16 Review 153

2.1 Preview

In this chapter, I provide background information on the broader field of quantum computing, starting from its roots in computing and quantum physics, progressing through key themes and developments in the field, and pointing out future directions based on the current state of the field. This chapter provides fundamental information for understanding the contributions of this thesis and for understanding why the research done in this thesis is even made possible due to the vital developments that have occured in the field.

2.2 Computing

Whilst the first mechanical calculating devices could be traced back thousands of years ago with the invention of the abacus, a hand-operated calculating tool, the first mechanical computer is thought to be the Difference Engine invented in 1822 by Charles Babbage, known as the father of the computer. He later invented the Analytical Engine in 1837. Ada Lovelace was the first to recognise that the machine had applications beyond pure calculation and programmed the machine. Throughout the late 19th century and the early 20th century, computers with new capabilities were invented including the Tabulating Machine in 1890 and the Differential Analyzer in 1930. In 1936, Alan Turing, known as the father of computer science, presented the Turing Machine, a model capable of implementing any computer algorithm [48].

After that, the first generation of computers, built using vacuum tubes, came forth, including in 1943 the Electronic Numerical Integrator and Computer (ENIAC), the first general purpose digital computer. Then, in 1947, the transistor was invented at Bell Labs. This led to the second generation of computers, built using transistors. In 1953 the International Business Machine

(IBM) computers were available for the public. Around the same time, Grace Hopper developed the first computer language. In 1954, IBM released the FORTRAN programming language. In 1958, Jack Kilby and Robert Noyce invented the integrated circuit, known as the computer chip. This led to the third generation of computers, built on integrated circuits.

During the 1970s - 1980s, the fourth generation computers were built using Very Large Scale Integration (VLSI) circuits. During this period, personal computers were introduced, including the founding of Microsoft in 1975 and Apple in 1976.

Since the 1980s, we have been in the fifth generation of computers, built on Ultra Large Scale Integration (ULSI) circuits. In 1989, Tim Berners-Lee proposed the World Wide Web. In this generation, the focus is on parallel-processing hardware and artificial intelligence. As computing hardware capabilities progressed, classical neural networks could be run with deeper layers, more parameters, and larger datasets, resulting in their successful performance today.

2.3 Quantum Physics

In the meantime, quantum theory was being developed by physicists. In 1900, Planck postulated that energy is made up of quanta (discrete units). Five years later, in 1905, Albert Einstein explained the photoelectric effect by suggesting that light consists of quanta as well [49]. The phrase *quantum mechanics* appeared for the first time in a paper by Max Born in 1924 [50]. In the same year, Louis de Broglie proposed that electrons exhibit particle or wave behaviour according to the experimental circumstances [51], which became known as the principle of wave-particle duality. The Schrödinger equation was developed by Erwin Schrödinger in 1926. One year later, in 1927, one of the first interpretations of quantum mechanics, the Copenhagen Interpretation, was developed. In 1927, Werner Heisenberg claimed that for certain pairs of physical properties, such as position and momentum, the more accurately one property is measured, the less accurately the other property can be known [52]. This became known as the uncertainty principle.

2.4 Quantum Computing

The foundation of quantum computing is quantum mechanics and computing, which are two fields that emerged independently in a similar era and have now merged together in our generation. One of the first attempts at constructing a quantum information theory was by Roman Stanisław Ingarden in 1976 [53]. In 1980, Paul Benioff published a paper describing a quantum mechanical model of a Turing machine [54] and in 1985, David Deutsch at the University of Oxford developed this idea and described the first universal quantum computer [55]. In 1981, Richard Feynman gave a keynote speech at the California Institute of Technology on the topic *Simulating physics with computers* where he uttered his famous words stating that nature isn't classical and "if you want to make a simulation of nature, you'd better make it quantum mechanical, and by golly it's a wonderful problem, because it doesn't look so easy" [56].

2.5 Quantum Algorithms

In 1992, the Deutsch-Jozsa algorithm was invented and was one of the first quantum algorithms shown to be exponentially faster than any possible deter-

ministic classical algorithm [57]. In 1993, one of the first papers on quantum teleportation was published [58] and Simon's algorithm was invented [59], which would be one of the building blocks for Shor's algorithm, used for factoring integers, which was developed in 1994 at Bell Labs [1]. Shor's algorithm drove significant interest in quantum computing as the impact of implementing the algorithm on a real quantum device would have dire implications for cryptography and security. A couple of years later, in 1996, also at Bell Labs, Lov Grover invented Grover's algorithm, a search algorithm [2].

2.6 Quantum Hardware

2.6.1 Brief History

Whilst the quantum computing theory was developing, several proposals for implementing a quantum computer were developing, including a proposal of trapped ions, in which the qubits are ions trapped by electric or magnetic fields and their energy states and interactions between them are changed using laser or microwave pulses which perform the quantum operations [60, 61]. The trapped-ion quantum computer was proposed by Ignacio Cirac and Peter Zoller

in 1994 known as the Cirac-Zoller proposal [7, 62]. In 1998, the first quantum computation was performed, with two qubits, by Isaac Chuang, Neil Gershenfeld and Mark Kubinec by implementing Grover's algorithm using Nuclear Magnetic Resonance (NMR) techniques [10]. In 1999, Y. Nakamura, Yu. A. Pashkin, and J.S. Tsai created the first functional superconducting qubit [8], which was considered a breakthrough as it resulted in the superconducting quantum computers we have today [63]. In the same year, D-Wave Systems, a company working on quantum annealers (special-purpose devices for optimisation problems), was founded and is considered to be the first company to have sold computers using quantum effects, commercialising their D-Wave One in 2011. In the year 2000, David P. DiVincenzo proposed the DiVincenzo criteria, which are necessary conditions for the construction of a quantum computer [64]. After that, more implementations of quantum computers emerged.

2.6.2 Categories of Quantum Hardware

A universal quantum computer can perform any arbitrary computation. There are two main categories of models of quantum computation, one being general-purpose (i.e. universal), and the other being special-purpose (i.e. non-universal).

The former category, which includes the universal quantum computing model or gate-based model or quantum circuit model, leads to quantum hardware that can implement a universal gate set, which means that they can implement quantum gates that can be used to run any arbitrary quantum circuit. The latter category leads to quantum hardware that can not implement a universal gate set, but can solve particular types of problems; this category includes quantum hardware approaches such as quantum annealers and Gaussian Boson Sampling (GBS) devices, in which GBS is a model of photonic quantum computation that produces an output distribution that can not be simulated in polynomial time with quantum computers [65, 66].

2.6.3 Levels of Quantum Hardware

The World Economic Forum Global Future Council on Quantum Computing, which I was a part of as the Council Fellow, defined four levels (Levels 1 - 4 below) of advancement for quantum hardware [21]. I have since added Level 0 to the list below to showcase additional approaches:

1. **Level 0: theoretical approaches that can not yet create a qubit**.

These approaches have a theoretical basis for building a qubit but have not physically implemented a qubit.

2. **Level 1: platforms demonstrating coherence properties**. These platforms can create qubits. All approaches not based on the quantum circuit model are considered Level 1.
3. **Level 2: platforms demonstrating a universal gate set**. These platforms can control qubits with quantum operations and would be able to run an arbitrary quantum algorithm.
4. **Level 3: platforms demonstrating quantum error correction and/or error mitigation**. These platforms can mitigate or correct errors in their device.
5. **Level 4: platforms demonstrating previous levels at scale**. These platforms would consist of high-quality qubits, so that including additional error-correcting qubits in the system would correct more errors than errors generated with additional qubits. Thus scaling to a larger number of qubits would reduce the error rates (in contrast to other levels, where scaling to a larger number of qubits may increase the error rates).

Table 2.1 provides examples of quantum hardware approaches for each level.

Level 0	Level 1	Level 2	Level 3	Level 4
• Topological qubits	• T-cells	• Neutral atoms • Photonic simulators • Spin systems • Nitrogen-vacancy (NV) centres	• Superconducting qubits • Trapped ions	• None yet

Table 2.1: **Levels of development for quantum hardware.** The levels of different quantum hardware platforms. Levels 1-4 examples are from Table 4 in Reference [21] by the *World Economic Forum, Global Future Council on Quantum Computing*. Level 0 is an addition for this work.

The development of quantum hardware somewhat resembles that of classical hardware, as in the early days of classical computers several implementations were developed, including vacuum tubes. The invention of transistors, however, overshadowed previous implementations, as transistors were smaller, more efficient, and more durable compared to vacuum tubes. In this current period of quantum hardware, however, each approach has its own advantages and disadvantages and there isn't a clear consensus for an approach that overshadows all other ones. Despite that, there are some approaches that are more advanced

than others, as outlined in Table 2.1. We can see that superconducting qubits and trapped ions are the most advanced approaches (at Level 3), which corresponds with the number of organisations working to build these approaches as shown in Table 2.2. On the other end of the spectrum, at Level 0, there are theoretical approaches for qubits that have not been experimentally realised yet. One of the most well-known examples is topological qubits, an approach proposed by Alexei Kitaev in 1997 [67]. Microsoft is attempting to build this approach [68] and in 2022, they announced that they were able to demonstrate experimentally the underlying physics to build a topological qubit [69, 70, 71]. Their next milestone is to build a topological qubit.

2.6.4 Outlook for Quantum Hardware

To evaluate quantum hardware, the World Economic Forum Council on Quantum Computing suggests three evaluation metrics [21]:

1. **Scale:** the number of connected qubits with universal control.
2. **Speed:** the number of quantum circuits executed in a given time.
3. **Quality:** the two-qubit gate error rates.

Organisation	Approach
D-wave	Quantum annealing
IBM	Superconducting qubit
Google	Superconducting qubit
Rigetti	Superconducting qubit
IonQ	Trapped ions
Quantinuum	Trapped ions
Microsoft	Topological qubit
Xanadu	Photonic simulators

Table 2.2: **Examples of organisations building different quantum hardware approaches**.

Therefore, the future of quantum hardware development includes scaling up the number of connected qubits and implementing error correction to reach Level 4. In addition to transforming existing approaches, the future may involve the invention of new approaches and techniques to try and find a qubit that can be controlled with the properties we desire, including scalability, quick operations, and low error rates, with the ultimate goal of producing scalable, reliable, and fast quantum computers.

2.7 Quantum Cloud Platforms

Universities and companies built quantum hardware in their labs and only their researchers could access these devices, similar to how the early classical computers were only accessible by specialists who physically interacted with the computers. With the development of personal classical computers, access to them significantly increased as people interacted with these devices in their home. Interestingly, personal quantum computers do not really exist today, yet access to quantum computers have increased, due to cloud computing, which is the ability to access and use computing resources through the internet and thus without the need to physically interact with the devices.

One major development in quantum computing is the accessibility of quantum hardware which increased dramatically after IBM made their quantum device accessible to anyone via the cloud in 2016 initially known as the IBM Quantum Experience [72]. Therefore, today there are many people, usually researchers and hobbyists, who have never physically interacted with a quantum computer and have probably never seen one in-person but have used quantum computing resources.

Over the years, several other companies started to allow restricted and open access to their quantum devices, including Rigetti [73] and Google [74]. More recently, companies are going a step further and are launching quantum-computing-as-a-service platforms, which allow users to access multiple quantum devices via one platform. One of the first quantum computing services was Amazon Braket in 2020 [75] and Microsoft Azure Quantum, whose platform became public in 2021 [76].

Quantum cloud platforms have greatly impacted the field of quantum computing as they have made it possible to run quantum algorithms on classical simulators and quantum hardware with less specialised quantum device knowledge and with greater ease of access. One of the reasons why hybrid quantum-classical algorithms boosted in popularity is because researchers could implement and test them on classical simulators and quantum devices offered by companies via the cloud.

2.8 Quantum Software

After the release of the IBM Quantum Experience in 2016, in 2017 IBM released Qiskit [77], an open-source software development kit. Since then, various other open-source software frameworks have been released including those specific for quantum machine learning such as PennyLane [78] by the Canadian quantum computing startup Xanadu and TensorFlow Quantum [24] by Google.

These software kits allow researchers to more efficiently simulate quantum computing algorithms. Researchers no longer need to implement their own code to perform the vector and matrix calculations of quantum circuits for simulating quantum algorithms, which adds time and resources to quantum computing research. Instead, they can use the software kits to simulate quantum algorithms more efficiently, which has helped the field grow more rapidly. Furthermore, these software kits implement more efficient ways to simulate quantum circuits, thereby decreasing the runtime of these simulations and speeding up research. In addition to quantum simulations, researchers can also use these software kits to interact with the quantum hardware, thereby allowing them to run quantum algorithms on the hardware.

As well as that, the software kits specific to machine learning also save a lot of time for researchers as they can even treat some parts of the quantum machine learning algorithms as black boxes and build on top of its components, extending the capabilities of quantum machine learning. These software kits also allow the quantum machine learning field to benefit from the performance and usability of classical machine learning software. For example, Tensorflow Quantum, integrated quantum circuits into the existing Tensorflow software, which is popular for implementing classical machine learning algorithms.

2.9 Quantum Tools

As quantum hardware and software developed, tools and infrastructure for quantum computing matured. One primary tool is a quantum compiler. Quantum compilers enable the interface between software and hardware as the compilers allow the quantum circuit instructions to be implemented on a specific quantum device. In addition, quantum compilers can help reduce the number of gate operations in a quantum circuit, allowing algorithms to be run more efficiently. Quantum compilers were mainly built by the quantum hardware providers, but independent quantum compilers also emerged, such as

Cambridge Quantum Computing Tket [13]. Quantum compilers are important for quantum algorithms as they can help them run more efficiently, both for simulators and hardware devices. Furthermore, quantum compilers can help reduce the errors associated with running the algorithms on the hardware. Another advancement in quantum tools is error mitigation. Error mitigation involves various techniques to reduce the errors that occur in NISQ devices [79]. This is mostly important for running quantum algorithms on actual quantum hardware.

2.10 Quantum Applications

With the invention of hybrid quantum-classical algorithms, the accessibility of quantum hardware, and the development of quantum software kits, the possibility for using quantum machine learning to solve real-world problems became more promising. Now there was a framework for solving optimisation problems, there was the ability to simulate the quantum algorithms and test whether they work, and there was the option to run it on quantum hardware to see whether it is truly feasible. As a result, researchers began to use quantum machine learning (through the use of hybrid quantum-classical algorithms) to

find solutions to real-world optimisation problems, such as portfolio optimisation in finance [80], as well as problems in a variety of fields including chemistry, materials science, transport, medicine, energy, and logistics [21]. This has undoubtedly vastly increased interest in quantum machine learning beyond the scientific community, as companies have realised its potential impact on their industry. Now, major companies in a variety of industries, such as Goldman Sachs in the financial industry [81], have dedicated quantum computing teams that mostly try to use quantum machine learning to solve problems in their industry.

2.11 Quantum Advantage

In 2012, John Preskill proposed the term *quantum supremacy* [82] "to describe the point where quantum computers can do things that classical computers can't, regardless of whether those tasks are useful" [83]. In 2018, a paper was published by Google to characterise what quantum supremacy would be in a quantum device [84] and in October 2019, Google announced they had achieved quantum supremacy in their quantum device called the Sycamore [85]. They stated that their quantum device could solve a task in about 200

seconds, whereas they assessed it would take approximately 10,000 years for a classical supercomputer. This achievement attracted significant attention as it was the first time that an actual quantum device could run a computation faster than a classical device. Shortly after, there was some push back, including from IBM who claimed that it would take a classical supercomputer 2.5 days instead of 10,000 years [86, 87]. Nonetheless, the news led to new enthusiasm for the field. The quantum supremacy task is thought to have little use cases, but the experiment generated hope to find quantum advantage, which is the point where quantum computers can do something better than classical computers for a useful task. At the time of this writing, we are still hoping for this quantum advantage. Google's quantum supremacy experiment has therefore motivated researchers to employ quantum machine learning to find such quantum advantage.

2.12 NISQ and Fault-Tolerant Devices

When quantum computing first developed, the algorithms assumed the resources of a fault-tolerant quantum computer, which is a quantum computer that can perform reliable computations with low error rates. As quantum hardware was

being built, however, the challenges with building them meant that they were far away from this ideal quantum computer. In 2018, John Preskill coined the quantum devices that existed as Noisy Intermediate-Scale Quantum (NISQ) devices. These devices were noisy, therefore having high error rates, and were intermediate-scale, meaning they had around 50-100 qubits, which seem enough to do something meaningful but not enough to drastically change computation.

Hybrid quantum-classical algorithms were built for these types of devices and that is why these algorithms are being used mostly today. There has been a lot of focus on these hybrid quantum-classical algorithms, because they make the most use of the quantum devices today and they carry with them the hope of achieving quantum advantage as soon as possible. It is still unknown today, however, whether these algorithms will be useful or not. Some researchers are therefore still fixing their eyes on building fault-tolerant quantum computers and even fault-tolerant quantum algorithms.

It is widely believed that the way to achieve fault-tolerance is through quantum error correction, which protects information from errors by encoding the information across multiple physical qubits (i.e. qubits in the hardware) to create a

logical qubit (i.e. a qubit in the abstract algorithm). A logical qubit consists of multiple physical qubits to act as one qubit in the algorithm. In February 2023, Google announced they created a logical qubit from 49 physical qubits and it was able to outperform one they made from 17 qubits [88, 89], which was the first demonstration of scaling a qubit. That is because more qubits equates to more error sources, so a logical qubit with more qubits must reduce the errors at a rate greater than the errors introduced by adding more qubits. Therefore, at the time of this writing, we are at the advent of quantum devices being built with logical qubits, bringing us steps closer to fault-tolerant quantum computers. This may change the landscape of quantum machine learning.

2.13 Variational Quantum Algorithms

As researchers realised that NISQ devices had certain characteristics, such as high levels of noise and a small number of qubits, that would make the implementation of traditional quantum algorithms difficult, they started to design new types of algorithms with NISQ devices in mind. These new algorithms became known as variational quantum algorithms, a subset of hybrid quantum-classical algorithms, and were first proposed as the Variational Quantum Eigensolver

(VQE) [15] and Quantum Approximate Optimization Algorithm (QAOA) [16] algorithms. One subclass of variational quantum algorithms is quantum neural networks. The goal of the variational algorithm is to minimise or maximise a cost function. In particular, we describe this cost function via the Hamiltonian H. The variational algorithm consists of an ansatz, which is the structure of the quantum circuit. This quantum circuit consists of single and two-qubit gates which are parametrised. The unitary matrix of the ansatz can be described as $U(\theta)$ and the output of the quantum circuit can be described as $|\psi(\theta)\rangle = U(\theta)\,|0\rangle$. Therefore, the goal of the variational quantum algorithm is to find $\min_{\theta} \langle\psi(\theta)|\, H\, |\psi(\theta)\rangle$.

This quantum circuit is run and its output is fed into a classical optimiser, which determines the values of the parameters for the next iteration to be run on the quantum circuit. This is done iteratively until the solution is optimised. This process is shown in Figure 2.1.

One of the reasons why we run these type of algorithms on NISQ devices is because the depth of the quantum circuits is shallow. Having a shallow depth allows us to obtain reasonable solutions, because noise accumulates with the

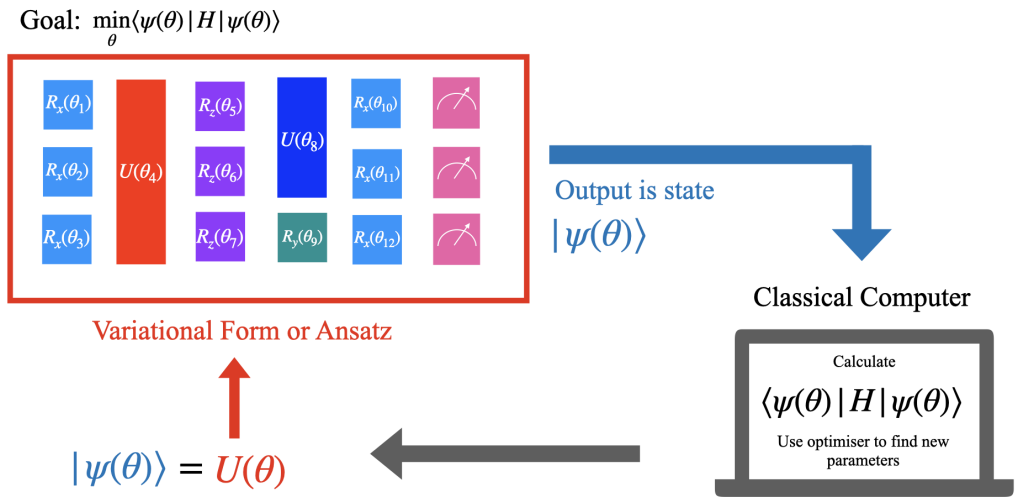


Figure 2.1: **Variational quantum algorithms process.** A quantum ansatz (in this case, a parametrised quantum circuit with a particular structure) has an output of $|\psi(\theta)\rangle$ which is then used for calculating a cost function on the classical computer. The optimiser uses this cost function to find a new set of parameters θ which are then used in the quantum ansatz. The process repeats until an optimal set of parameters is found.

depth of the circuit in NISQ devices. If too much noise accumulates with a large-depth circuit, then the solution would be too noisy and therefore not useful.

Noise in quantum computing refers to any disturbance that interferes with the computation and may weaken the accuracy of the computational results. There are two main sources of quantum noise [90] including:

1. **Decoherence:** this occurs when the quantum system interacts with the environment and this interaction causes a loss of the quantum behaviour of the system. Environmental factors, such as disturbances in the earth's magnetic field and radiation from Wi-Fi or mobile phones can affect the decoherence of the system. Another cause of decoherence can be crosstalk, which occurs when neighbouring qubits and measurements performed on them can influence the behaviour of other qubits, leading to a deviation of the ideal state of the quantum system.
2. **Operation errors:** this occurs when a quantum operation (implemented with a quantum gate or a measurement operation for example) is performed incorrectly. For example, the ideal quantum gate should be an X gate, which flips a qubit, but instead it performs a different operation. A particular type of error affecting noise is readout errors, which occur when the measurement operations are performed incorrectly and result in an incorrect measurement of the system.

Decoherence and gate errors (a type of operation error resulting from errors during the implementation of quantum gates) can compound as the circuit gets deeper. In the case of decoherence, each quantum operation takes a certain

amount of time and if there are more quantum operations, the greater the interaction of the quantum system with the environment and thus the greater the decoherence. How long a qubit can maintain its quantum state is its coherence time. In the case of gate errors, the deeper the circuit, the larger the number of gates and therefore more opportunity for gate errors to affect the final quantum state.

The limitation of how deep the circuit can be imposes a limit on the computational power of the quantum computer [14]. Despite that, variational quantum algorithms have some use as such algorithms have some resilience to noise. This is partly because they use shallow quantum circuits which reduce the effect of decoherence and gate errors. In addition, the variational nature of the circuit means that it is responsive to intermediate measurements, meaning that new parameters are suggested to minimise the impact of noise. This has been shown in Reference [91] where they "observed that variational algorithms mitigate the effects of noise by adapting the optimized parameters". They also found that the "inclusion of redundant parameterized gates makes the quantum circuits more resilient to noise." Furthermore, variational algorithms calculate the average over many measurements, which can reduce noise.

2.14 Quantum Machine Learning

2.14.1 First Wave of Quantum Machine Learning

The earliest works on quantum machine learning were not too long after Shor's algorithm was published. These early works in 1995 were focused on quantum models of neural networks [23, 92]. Their motivation, though, was to understand how the brain works through quantum theory (a topic still debated today), as mentioned in Reference [93]. Around the late 1990s and early 2000s, there were some sporadic papers discussing quantum computing and learning theory [94] and papers on various topics in quantum machine learning, including Quantum Associative Memory [95], Quantum Support Vector Machines [96], and Quantum Neural Networks [97, 98]. In 2002, there was a workshop on *Quantum Computation and Learning* [99], and we refer to a paper titled *Survey of Quantum Learning*, which reviewed some of these early papers [99].

Despite these early works, the field was fragmented and not well established. It is thought that the advent of quantum machine learning began with the publication of the HHL algorithm, which can solve a system of linear equations

(with some caveats), invented by Aram Harrow, Avinatan Hassidim, and Seth Lloyd in 2008 [22]. This led to the first wave of quantum machine learning [100], which consisted of quantum algorithms that speed up linear algebra operations. The reasoning for the dawn of the first wave of quantum machine learning was that many classical machine learning algorithms require linear algebra computations of high-dimensional vectors and a quantum algorithm, which works in the realm of linear algebra computations, could speed up some of the linear algebra subroutines in a machine learning algorithm and therefore provide a quantum speedup for various machine learning algorithms.

One of the main reasons for the speedup is that these quantum algorithms can use amplitude encoding. Amplitude encoding allows 2^n-dimensional vectors to be encoded into the 2^n amplitudes of n qubits. These quantum algorithms then use resources that grow polynomially in the number of qubits n, which leads to an exponential speedup with respect to the 2^n-dimensional input.

Therefore, quantum versions of machine learning algorithms were built upon the HHL algorithm and claimed an exponential speedup because of this efficient encoding of data. A lot of these quantum machine learning algorithms were

proposed from 2013 onwards, including principal component analysis [101] and support vector machine in 2013 [102], and recommendation systems in 2016 [103]. In 2014, Peter Wittek published a book titled *Quantum Machine Learning - What quantum computing means to data mining* which reviewed some of these early papers in quantum machine learning [104].

One major challenge of these algorithms is encoding the vector into the amplitudes. There have been several proposals, including most notably in 2008, a quantum device called *quantum random access memory* (qRAM) for loading data sets into quantum states [105]. It is still unclear, however, how to do this efficiently and practically [106].

Another caveat with these quantum machine learning algorithms is that if similar assumptions that are made for the quantum algorithms are also made for classical algorithms, the latter can have similar time complexity as the former, which was famously shown for the recommendation system algorithm by Tang in 2019 [107] and principal component analysis in 2021 [108].

2.14.1.1 Grover-Inspired Algorithms

In addition to the quantum machine learning algorithms relying on the HHL algorithm to gain an exponential speedup, there was also another group of quantum machine learning algorithms, relying on Grover's search, to gain a quadratic speedup. These algorithms could be translated into an unstructured search task. These algorithms mostly consisted of unsupervised machine learning algorithms, including K-medians [109], K-means [110], and K-nearest neighbours algorithm [111].

2.14.2 Adiabatic Computing

So far, we have focused on the gate-based model of quantum computation, in which computations are performed by applying a sequence of logical quantum gates to qubits; this model is universal, meaning that it can implement any arbitrary quantum algorithm, and is the most popular model [112]. Yet, there is another model of quantum computation known as adiabatic quantum computation [113], which is also universal (i.e. it can perform any computation). We want to note in this subsection that in the field of quantum machine learning, there have been several algorithms based on the adiabatic model of computation.

In adiabatic quantum computation, to solve an optimisation problem, the solution to a problem is found in the ground state of a Hamiltonian H_F. The system is initialised to the ground state of a simpler Hamiltonian H_I and it is adiabatically evolved to the ground state of the Hamiltonian H_F by the adiabatic theorem, which essentially states that a quantum system will remain in a ground-state after starting in a ground-state if the state is evolved slowly. Therefore, this time-varying Hamiltonian $H(t)$ starts off as H_I and ends up to our solution H_F [112].

Quantum annealing is similar to adiabatic quantum computation, but uses an arbitrary initial state and a transverse Hamiltonian H_D that converges to the final Hamiltonian H_F that encodes the solution. The evolution of the system is as follows: $H(t) = H_F + \Gamma(t)H_D$, where $\Gamma(t)$ is the transverse field coefficient, which converges to zero during the evolution [112]. Quantum annealing is not universal.

D-Wave Systems, a Canadian company founded in 1999, claims to be the first company to sell a computer explicitly exploiting quantum effects. Their devices are based on quantum annealing, which performs optimisation; therefore one

application of these devices is machine learning, which relies on optimisation. Therefore, some early and ongoing works in quantum machine learning have involved quantum annealing and have been implemented on devices such as D-Wave devices. For example in 2009, the *QBoost* algorithm was proposed and implemented on a D-Wave quantum device [114].

2.14.3 Second Wave of Quantum Machine Learning

The first wave of quantum machine learning relied on the need of fault-tolerant quantum devices for implementation, as these algorithms required a large number of quantum resources, including high-quality qubits and therefore error correction. In the meantime, from the first implementation of a quantum computer in 1998 to the present time, quantum devices have been built with a small number of low-quality qubits. As a result, variational quantum algorithms were invented around 2013. These algorithms could be implemented on these limited quantum devices. As these new framework of algorithms developed, they increasingly became used for quantum machine learning tasks, in particular in the form of quantum neural networks. This has been coined as the second wave of quantum machine learning and is the dominant current context of quantum

machine learning at the time of this writing. Quantum neural networks are explored further in Chapter 3.

2.15 Future Directions

It is possible that a quantum advantage will be found with the hybrid quantum-classical algorithms on NISQ devices. If that would be the case, that would generate more interest in these algorithms and their possibility. It is also possible that fault-tolerant quantum computing advances before any advantage in NISQ devices is found, and therefore this second wave of quantum machine learning will be only a wave and dissipate and fade as fault-tolerant quantum machine learning algorithms dominate. It is also possible that there will continue to be a mixture of use cases for both hybrid quantum-classical algorithms and fault-tolerant algorithms. Furthermore, if fault-tolerant quantum algorithms advance and become applicable for speeding up kernel methods, a class of machine learning algorithms, then kernel methods could dominate quantum machine learning and quantum neural networks may not be as useful. Nonetheless, it is still possible that more advances are made in quantum neural networks, demonstrating their practicality and usefulness for quantum machine learning

tasks.

2.16 Review

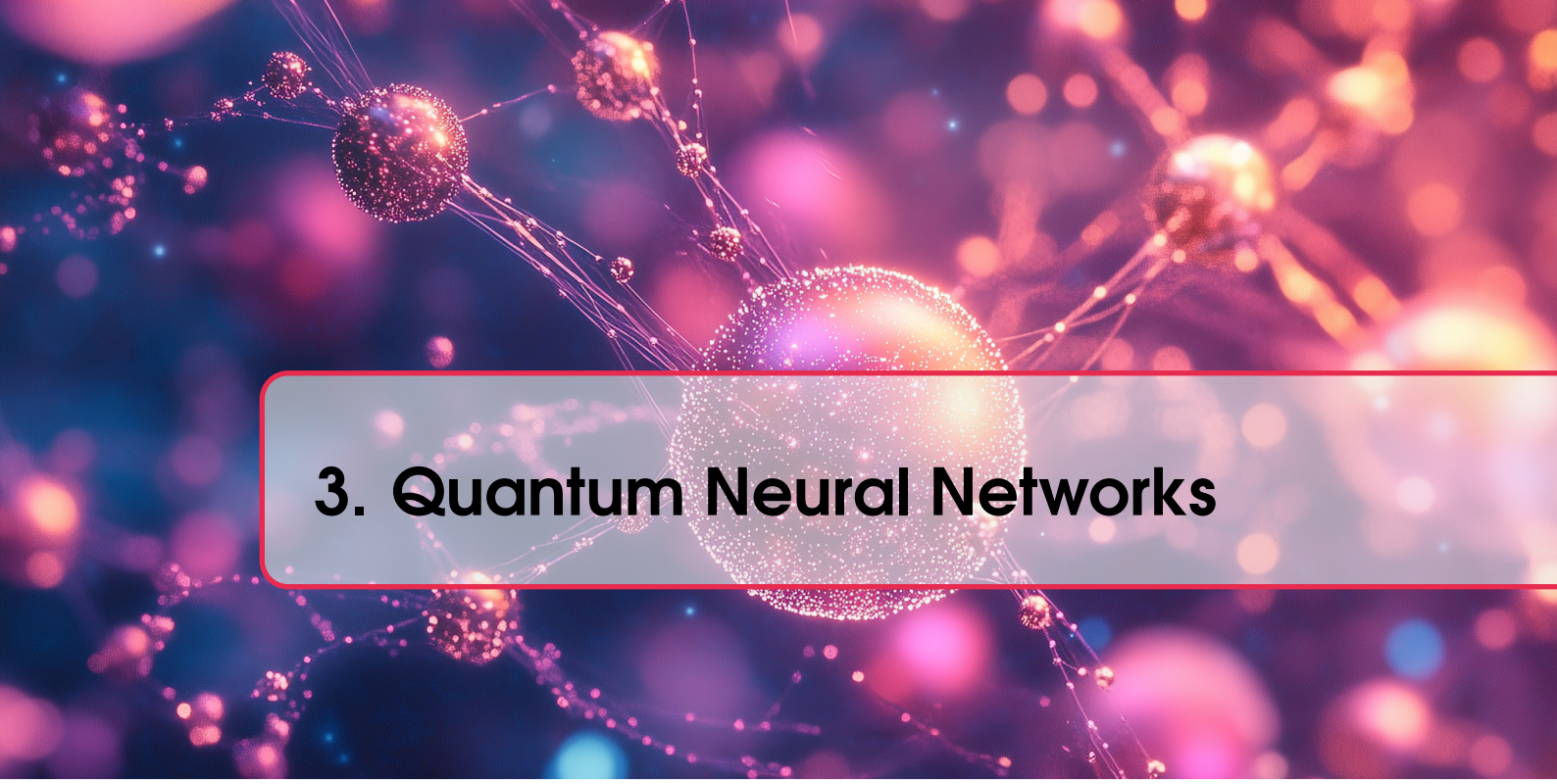
This thesis sits at the intersection of all of these trends and contexts. In Chapter 6, we give insight into the applicability of quantum neural networks, which are types of variational quantum algorithms, for quantum advantage and their potential usefulness for solving classical problems. We do this by using Qiskit to perform the experiments to find the inductive bias and expressivity in different quantum neural networks.

In Chapter 7, we contribute to the development of a quantum tool, a quantum compiler, by constructing a new method for optimising quantum circuits. Our work helps make certain quantum circuits to be run more efficiently on classical simulators and quantum hardware. We did this by programming our own quantum software to make the compiler and also adding some compatibility with other software kits, such as Qiskit.

In Appendix A, we demonstrate how all of the developments in quantum

computing can come together to solve a real-world problem on a quantum device. We use quantum hardware, in particular IBM's superconducting quantum computer which is a NISQ device, to run a variational quantum algorithm to solve an optimisation problem. We used IBM's cloud platform to run the algorithm on their classical simulator and quantum hardware. We used IBM's open-source software kit, Qiskit, to write the code to run our experiments by running our version of the variational quantum algorithm on different inputs. We implicitly used IBM's quantum compiler, which is seamlessly integrated in the software, to compile our quantum circuit instructions to the quantum hardware and we explicitly used IBM's error mitigation techniques to mitigate the errors when running on the quantum hardware.

In the next chapter, Chapter 3, I concentrate on quantum neural networks, which emerges from the intersection of variational quantum algorithms and quantum machine learning.



3. Quantum Neural Networks

3.1 Preview 156

3.2 Quantum Neural Networks Overview 156

3.3 History 157

3.4 How Quantum Neural Networks Work 164

3.5 Quantum Kernels 183

3.6 Characteristics of QNNs from the Literature 200

3.7 Quantum Advantage 213

3.8 State of the Field 226

3.9 Review 227

3.1 Preview

In the previous chapter, Chapter 2, I described key developments in the field, including variational quantum algorithms and quantum machine learning. Through the intersection of these two developments, the current framework of quantum neural networks emerged and they are being explored today for their potential use cases. In this chapter, I outline the history of quantum neural networks and describe the technicalities of how they work and what is already understood about them. This chapter provides crucial information for understanding the contributions and context for Chapter 6, in which we perform original research on these quantum neural networks.

3.2 Quantum Neural Networks Overview

Quantum Neural Networks (QNNs) integrate quantum computation principles with the structure and principles of neural networks. Classical neural networks have become powerful and are having a tremendous impact on society. Universities and companies are pouring a lot of resources into improving neural networks by, for example, using more data, investigating new architectures,

and harnessing more classical computational power. These are incremental improvements, but what if there could be a quantum leap in the advancement of neural networks? The hope is that QNNs will dramatically improve the performance of artificial intelligence in certain contexts by taking advantage of the quantum mechanical nature of quantum computers.

3.3 History

The first idea integrating neural networks and quantum mechanics was in 1995 published by Kak [23], just one year after Shor published the famous Shor's algorithm, proving the potential of quantum computing [1]. Since then, there have been numerous proposals of QNNs, which are reviewed by Schuld et al. in *The Quest for a Quantum Neural Network* [115]. Various QNNs implemented on different hardware architectures have also been presented in the paper titled *Parameterized quantum circuits as machine learning models* [116] (Table 2 in the paper).

Schuld et al. in Reference [115] established three minimum requirements for a meaningful QNN:

1. **Input-output relation of QNNs**: The initial state of the quantum system encodes any binary string of length N. The QNN produces a stable output configuration encoding the one state of 2^M possible output binary strings of length M which is closest to the input by some distance measure.
2. **Foundation in Neural Network Theory**: The QNN reflects one or more basic neural computing mechanisms (attractor dynamics, synaptic connections, integrate and fire, training rules, structure of a NN).
3. **Use of and consistency with quantum theory**: The evolution is based on quantum effects, such as superposition, entanglement and interference, and it is fully consistent with quantum theory.

One of the major stumbling blocks for constructing such a QNN was the difference between neural networks and quantum computing: neural network computing is nonlinear, whereas quantum computing is linear. Nonlinearity in quantum computing would contradict the linear evolution in quantum theory. One exception to this is taking measurements in quantum circuits and therefore some QNN proposals introduced measurements for nonlinearity [23, 117]. There were also other QNN proposals for resolving this conflict between neural networks and quantum computing [118, 119]. These proposals, however, did not address all three criteria that Schuld et al. [115] established. Schuld et al. [115] pointed out that (at the time of their paper) no proposal

satisfied the first requirement. Some proposals satisfied the second requirement but neglected the third requirement or vice versa. Thus they suggested that new approaches were needed for constructing a QNN.

In 2017, Cao et al. [120], proposed a construction of a quantum neuron, the fundamental unit of a neural network, that was the first explicit proposal to satisfy all three of the criteria that Schuld et al. [115] established for a meaningful QNN. This quantum neuron, which could be used as a building block of QNNs, was able to "simulate classical neurons with sigmoid or step function activation while processing inputs in quantum superposition" [120]. This proposal, however, was not a general framework for supervised learning tasks to be implemented on NISQ devices.

In the meantime, hybrid quantum-classical algorithms were developed in 2013 by Peruzzo et al. [15] and Farhi et al. [16], so that these algorithms could be implemented on NISQ devices. As the name suggests, these algorithms outsource parts of its computation to a classical computer. The reason to do this is because current quantum devices have a limited number of qubits and gates, and high error rates. Therefore these algorithms can run on quantum

devices and use fewer of its resources (qubits, gate depth, etc.).

A particular type of hybrid quantum-classical algorithm is the variational quantum algorithm (VQA) that consists of a variational quantum circuit, which is a quantum circuit with parameters that can be changed. The variational quantum algorithm uses a classical optimiser to adjust the parameters so that the variational circuit with the optimal parameters corresponds to the solution to the problem one is trying to solve. An overview of variational quantum algorithms is in the paper titled *Variational Quantum Algorithms* [20].

The first proposed examples from the class of variational algorithms include the Variational Quantum Eigensolver (VQE) [15], which finds minimum energy eigenstates of Hamiltonians, and the Quantum Approximate Optimization Algorithm (QAOA) [16], which solves a combinatorial optimisation problem, in which the aim is to find the optimal solution from a finite set of possible solutions.

After the invention of variational quantum algorithms, other researchers introduced variational circuits into quantum machine learning [19, 121, 122], though

the concept had been mentioned earlier (e.g. in the paper titled *Simulations of quantum neural networks* [119]). Benedetti et al. reviewed how parameterised quantum circuits (quantum circuits consisting of gates with parameters that can be changed) are used as machine learning models in the paper titled *Parameterized quantum circuits as machine learning models* [116]. One notable example is the paper titled *Classification with Quantum Neural Networks on Near Term Processors* in February 2018 by Farhi et al. [19], who proposed a general framework for supervised learning designed to be implemented on NISQ devices. The framework includes a method for converting the classical data into the quantum circuit; the method is called the *basis encoding circuit* (explained in Section 3.4.1). The framework also includes a variational circuit with an ancilla qubit, which is an extra qubit used as a *readout qubit*, which is used for the measurement of the classification labels (explained in Section 3.4.2). They showed that there is a construction of a QNN that can in principle express all Boolean functions (explained in Section 4.8.1), but certain functions may require an exponentially deep circuit. This is analogous to the *universal approximation theorem* [123, 124] in classical machine learning which states that "neural networks with an exponentially large hidden layer of non-linear

neurons are able to represent any Boolean function" [116]. Farhi et al.'s paper also served as a proof-of-principle that this framework could be used to classify real world data.

As one can see, there has been a shift in quantum machine learning because of the development of QNNs. In the beginning, the goal in quantum machine learning was to find quantum speedups for classical machine learning models. But then, NISQ devices were improving and variational quantum algorithms, including QNNs, were becoming more widespread as they could be run on NISQ devices; these developments made it easier to run quantum algorithms in practice but it became harder to find theoretical speedups. Therefore, the field has become more experimental and observational. There has also been a shift in the meaning of QNNs. In the beginning, the goal was to find a QNN model that unified classical neural network theory and quantum computing, by finding a model that satisfied Schuld et al.'s criteria [115]. Now, however, QNNs mostly refer to variational quantum circuits, especially when they are used in machine learning contexts.

One question that arises now is whether QNNs do truly capture the meaning

of neural networks. On the one hand, they do capture some aspects of neural networks, including its trainable parameters and layers, which mimic a deep linear neural network. They correspond to a linear neural network because the quantum circuit applies unitary transformations to the input. The measurement in the quantum circuit can correspond to a non-linear layer, as it computes the absolute square of each amplitude and linearly combines them to a scalar value. Given that quantum circuits are unitary, the non-linearity is mostly introduced through the way that the data is encoded into the QNN; in other words, the way the classical data is being converted into a quantum state. On the other hand, a QNN does not correspond to a neural network, as the latter consists of multiple layers of perceptrons, the building block of classical neural networks, which apply non-linear transformations at each layer.

Therefore, there are links between QNNs and neural networks. Schuld and Killoran [26] show that a much more direct and natural link is that of QNNs and kernel methods as described in Section 3.5.

3.4 How Quantum Neural Networks Work

I introduce QNNs and their different components, which are used in this work. In a QNN, there are three distinct components [125] shown in Figure 3.1:

1. **Encoder circuit**: this quantum circuit encodes the data into the QNN. Encoding refers to converting the classical data into quantum states so that the data can be processed by a quantum algorithm.
2. **Variational circuit**: this quantum circuit is parametrised and the parameters are updated during the learning process of the QNN. The variational circuit starts as an *ansatz*, which is defined as an educated guess of the solution to a problem and is used as a starting point. In the context of QNNs, the ansatz is the parametrised circuit that is used as a starting point or trial state which is iteratively updated during the optimisation process.
3. **Measurement operators**: information from the QNN is extracted via measurements. From these measurements, gradients and a loss function can be calculated on a classical computer and the parameters are updated via some classical optimiser, such as Adam [126].

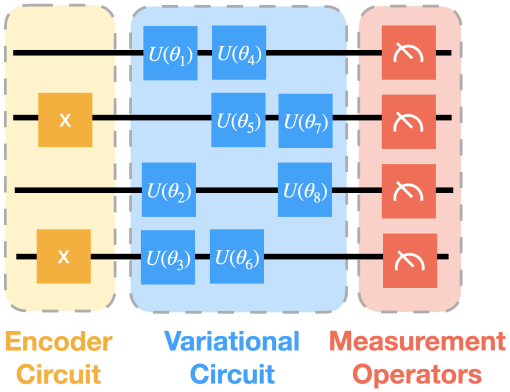


Figure 3.1: **Quantum neural network with three distinct parts.** The quantum neural network consists of the (1) Encoder Circuit, which encodes the data into the quantum neural network, the (2) Variational Circuit, which is parametrised and has its parameters optimised, and the (3) Measurement Operators, which retrieve classical information from the quantum neural network. Figure inspired by Figure 2 in Reference [125].

During the learning process, the measurement outcomes of the QNN are fed into a classical computer, which makes calculations, including the loss function and gradients. Then the parameters are updated via some classical optimiser. These parameters are updated in the variational circuit in the QNN and the process is repeated until the desired loss is reached. Figure 3.2 shows this process.

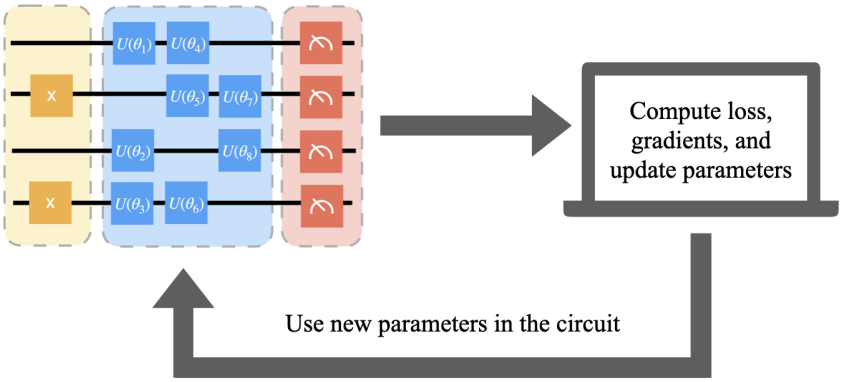


Figure 3.2: **Learning process for a quantum neural network.** The classical information retrieved from the measurement operators are fed into a classical optimiser to compute the loss, gradients, and update the parameters. These new parameters are updated in the parametrised quantum gates in the variational circuit and the process is repeated until the output of the quantum neural network converges or the process satisfies a stopping condition.

3.4.1 Encoding Methods

There are multiple ways to encode data into a quantum circuit. Below are common encoding methods that are analysed in this work:

3.4.1.1 Basis Encoding

This maps the input data (in the form of a binary string) into the computational basis of a quantum state. If our vector $\vec{x}$ is in the binary format, then basis encoding makes the following transformation: $\vec{x} = b_{n-1}b_{n-2}\cdots b_0 \rightarrow |x\rangle = |b_{n-1}b_{n-2}\cdots b_0\rangle$. For example, basis encoding transforms the input data $\vec{x} =$

$(1, 0)$ into $|10\rangle$. This requires $O(n)$ qubits. Further details are in Section 3.5.1.3.

3.4.1.2 Amplitude Encoding

This encodes the data into the amplitudes of the quantum state, so that our quantum state becomes $|\psi\rangle = \frac{1}{\|\vec{x}\|} \sum_{i=0}^{n-1} x_i |i\rangle$, where our input data is $\vec{x} = x_{n-1}x_{n-2} \cdots x_0$ and x_i is the ith element of $\vec{x}$ and $|i\rangle$ is the ith computational basis state. For example, if our vector is $\vec{x} = (1, 0, 1)$, then the norm is $\sqrt{1^2 + 0^2 + (1)^2} = \sqrt{2}$, so our encoded quantum state becomes $\frac{1}{\sqrt{2}} |00\rangle + \frac{1}{\sqrt{2}} |10\rangle$. Extra computational basis states can be encoded as zero. This requires $O(\log n)$ qubits. Further details are in Section 3.5.1.6.

3.4.1.3 ZZ Feature Map

The ZZ Feature Map is based on the paper titled *Supervised learning with quantum-enhanced feature spaces* [127] and encodes the input data into the angles of the unitary matrices in a particular way. This encoding method is designed to be biased towards the parity function. They define the feature map as $\mathcal{U}_\Phi(\vec{x}) = U_{\Phi(\vec{x})} H^{\otimes n} U_{\Phi(\vec{x})} H^{\otimes n}$, where $U_{\phi(x)} = \exp\left(i \sum_{S \subseteq [n]} \phi_S(x) \prod_{i \in S} Z_i\right)$, where H is the Hadamard gate and is equivalent to the matrix $\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$, Z is

the Pauli-Z gate and is equivalent to the matrix $\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}$, S is a test set of data, n is the number of qubits, and $\phi_i(x) = x_i, \phi_{\{i,j\}}(x) = (\pi - x_0)\,(\pi - x_1)$.

This *feature map*, which is a particular quantum circuit that encodes classical data into a quantum state, equivalent to mapping the classical features of the data into the higher-dimensional space of quantum states, can be seen more clearly in its circuit diagram for two qubits in Figure 3.3. Further details are in Section 3.5.1.4.

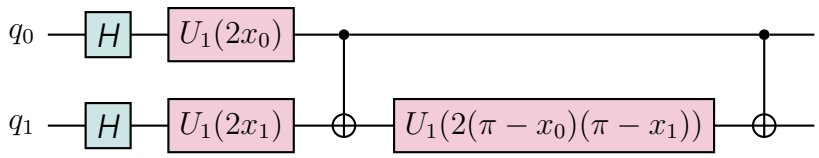


Figure 3.3: **ZZ feature map for two qubits.** This feature map is used for encoding data into a QNN. The feature map for two qubits consists of Hadamard gates, followed by U_1 gates where the first data point x_0 is encoded in the parameter for the U_1 gate acting upon the first qubit q_0 and the second data point x_1 is encoded in the parameter for the U_1 gate acting upon the second qubit q_1. A CX gate follows, then another U_1 gate on the second qubit q_1, which encodes both data points x_0 and x_1 in the parameter for the U_1 gate. Another CX gate follows. Figure inspired by Reference [128].

3.4.1.4 Random Relu Transform

Although the other encoding methods are common, we designed this encoding method to act as a random non-unitary transformation to the data. This is equivalent to the data being transformed through a single hidden layer of a classical neural network. To implement this encoding method, we encode the data using basis encoding and then apply a random unitary matrix. We then perform classical processing, which includes applying the ReLU function, which introduces nonlinearity, to the resulting statevector and normalising it. The transformation is as follows: $\psi = \frac{1}{\|\sigma(Ux)\|} \sum_{i=1}^{N} \sigma(Ux)_i |i\rangle$, where our input data is $\vec{x} = x_{n-1}x_{n-2}\cdots x_0$ and x_i is the ith element of $\vec{x}$ and $|i\rangle$ is the ith computational basis state. $\sigma(Ux)_i$ is the ith output of a single hidden layer neural network, σ is the ReLU function, and U is the random unitary matrix. Further details are in Section 3.5.1.5.

3.4.1.5 Other Encoding Methods

There are also other encoding methods; for example, *data re-uploading* re-encodes the classical input data at various places within the variational quantum circuit by using quantum gates, which are alternated with the parametrised

(the trainable) quantum gates [129, 130]. Figure 3.4 shows an example of a general structure of using such an encoding method.

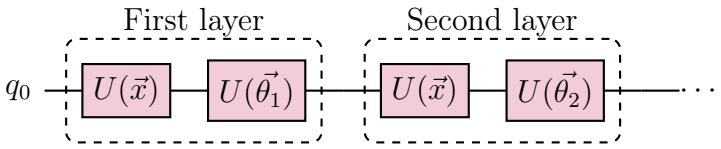


Figure 3.4: **Data re-uploading.** The classical input data $\vec{x}$ is re-uploaded into different layers in the quantum circuit, alternated with the parametrised (trainable) quantum gates with parameters $\vec{\theta}$. Figure inspired by Reference [131].

3.4.2 Variational Circuit

A variational circuit is a parametrised quantum circuit, which is a circuit containing gates with parameters (parametrised quantum gates) that can be adjusted. Every quantum gate is a unitary operation U, meaning that $U^\dagger U = I$. Therefore, a parametrised quantum gate can be represented as $U(\vec{\theta})$. The variational quantum circuit consists of a sequence of unitary gates applied to multiple qubits.

3.4.2.1 Expressivity of Variational Circuits

The expressivity of a quantum variational circuit is typically defined as the states in the Hilbert space the variational quantum circuit can express. In this work, we study the Boolean system (further details are provided in Section 4.8.1) and therefore we define a different type of expressivity, which we call the *Boolean expressivity* – which is defined as the Boolean functions the quantum variational circuit can express. The Boolean expressivity is less strict than the Hilbert space expressivity, meaning that a variational quantum circuit can have maximum Boolean expressivity (can represent all Boolean functions) but not maximum Hilbert space expressivity (can represent all possible quantum states). If a quantum circuit has maximum Hilbert space expressivity, it must also have maximum Boolean expressivity. If a quantum circuit has maximum Hilbert space expressivity, meaning that it can express all possible quantum states, then it can also express all possible Boolean functions as the Boolean functions are derived by thresholding the outcomes of quantum states.

The quantum circuit for classification in Figure 3.8 has neither maximum Hilbert space expressivity nor maximum Boolean expressivity. This can pose

a problem when studying the Boolean system, as in order to understand the QNN's true inherent bias, it should be able to express all possible Boolean functions, otherwise any bias would be a byproduct of its lack of expressivity and not from the system itself. Therefore, we introduce in the next section a quantum variational circuit that is *fully expressive* in Boolean functions.

3.4.2.2 Fully Boolean Expressive QNNs

In order to construct a QNN that can learn all Boolean functions, which we call a *fully expressive QNN*, we use the QNN presented in the paper titled *Tunable Quantum Neural Networks for Boolean Functions* [132], where the authors show that a quantum circuit can be designed to express a particular Boolean function. As a result, a general quantum circuit can have parameters that can be modified to learn any Boolean function.

(a) Quantum Boolean Circuit

An example of a Boolean function is shown in Table 3.1, and Figure 3.5 shows the corresponding quantum circuit to express this Boolean function. In general, a quantum Boolean circuit has n data qubits, corresponding to the length of

the input data $\vec{x}$ and one readout qubit q_r which corresponds to $f(\vec{x})$. The quantum Boolean circuit consists of multi-controlled X gates where the qubits with the symbol $\bullet$ are the control qubits and the qubits with the symbol $\boxed{X}$ are the target qubits. If a controlled qubit is equivalent to the state $|0\rangle$, then nothing happens to the target qubit, whereas if all controlled qubits are equivalent to the state $|1\rangle$, the X gate is applied to the target qubit, effectively flipping the target qubit.

x	$f(x)$
000	0
001	0
010	1
011	0
100	1
101	0
110	0
111	1

Table 3.1: **Example of a Boolean function.** x consists of all the possible combinations of a binary string with length three. Each input x_i is mapped to an output $f(x_i)$, which is the Boolean function 00101001.

As examples, one can see that if all the qubits are equivalent to $|0\rangle$ then the readout qubit will also remain in the $|0\rangle$ state – this corresponds to the first row in Table 3.1 where $x = 000$ and $f(x) = 0$. Then, we can see that if the first qubit is in the state $|0\rangle$, the second qubit in the state $|1\rangle$ and the third qubit

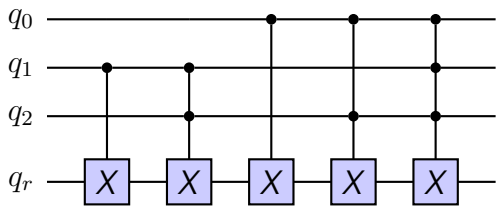


Figure 3.5: **Quantum circuit for expressing the Boolean function 00101001.** The Boolean function in Table 3.1 can be expressed as this quantum circuit. The circuit uses multi-controlled X gates to express the behaviour of the Boolean function. Quantum circuit inspired by Figure 2 in Reference [132].

in the state $|0\rangle$, then this corresponds to the first gate in the quantum circuit, which has a control on the second qubit (q_1) and target on the readout qubit (q_r). The readout qubit will be flipped to the $|1\rangle$ state and this is equivalent to the third row in the Table 3.1 where $x = 010$ and $f(x) = 1$. Finally, if all the qubits are equivalent to $|1\rangle$, then the readout qubit will be flipped and be in the state $|1\rangle$ corresponding to the last row in Table 3.1 where $x = 111$ and $f(x) = 1$.

(b) Boolean QNN

In order to extend this Quantum Boolean circuit into a Boolean QNN, the paper [132] introduces G_u, a tunable quantum gate, whose value can either be I,

the identity gate, or X, the X gate, which flips a qubit and is equivalent to the matrix $\left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$, that is $G_u \in \{I, X\}$, where $u = u_0 \cdots u_{n-1} \in \mathbb{B}^n$ and $\mathbb{B} = \{0, 1\}$ is the set of the Boolean values and $\mathbb{B}^{\mathbb{B}^n}$ the set of functions from $\mathbb{B}^n$ to $\mathbb{B}$. C_u is the multi-controlled X gate. The gates C_u in the quantum Boolean circuit commute together and I commutes with all matrices, so the order of the gates does not change the overall circuit. A QNN with n data qubits contains 2^n gates. Each gate has two possible values, so there exists in total 2^{2^n} different circuits. The fully expressive QNN for $n = 3$ is shown in Figure 3.6.

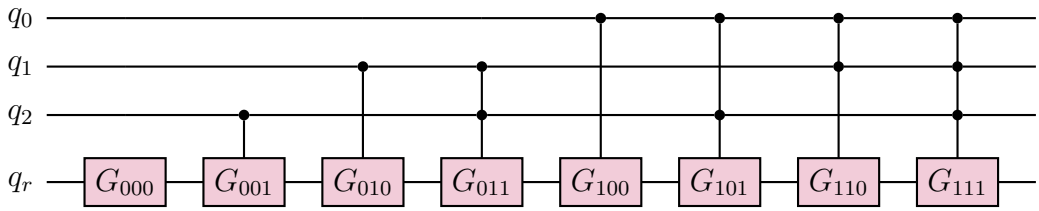


Figure 3.6: **Boolean QNN for three inputs.** This QNN can express any Boolean function with length three. Quantum circuit inspired by Figure 3 in Reference [132].

(c) Parametrised Boolean QNN

In this work, we are interested in how the functions change given randomly sampled parameters. Therefore we replace the G_u gates with U_3 gates, where

U_3 is a general single qubit rotation gate

$$U_3 = \begin{pmatrix} \cos(\theta/2) & -e^{i\lambda}\sin(\theta/2) \\ e^{i\phi}\sin(\theta/2) & e^{i(\phi+\lambda)}\cos(\theta/2) \end{pmatrix}.$$

U_3 can represent any quantum gate, including the Identity Gate and the X gate. Therefore, this parametrised boolean QNN can be equivalent to the boolean QNN when the unitary gates are selected to the be the Identity and X gates. In particular, one can see that $U_3(\theta = 0, \phi = \pi, \lambda = \pi) = I$, where $I = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$. $U_3(\theta = \pi, \lambda = \pi, \phi = 0) = X$, where $X = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$.

Therefore, the Boolean QNN in Figure 3.6 is transformed to the parametrised Boolean QNN in Figure 3.7.

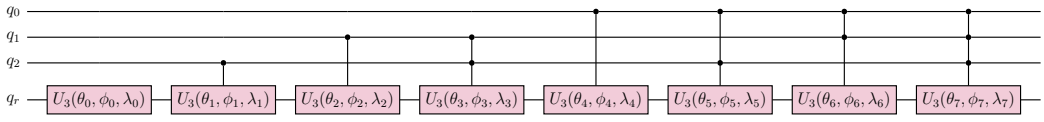


Figure 3.7: **Boolean QNN for three inputs.** This QNN can express any Boolean function of length three and is overparametrised as it contains more parameters needed to express any Boolean function of length three.

3.4.2.3 Non-Fully Expressive QNNs

We introduce a variational quantum circuit based on the paper titled *Classification with Quantum Neural Networks on Near Term Processors* [19]; this variational quantum circuit is used for classification and has been implemented in Tensorflow Quantum [24] to classify simplified MNIST data [133], which are images of handwritten digits used as a basic dataset for classification in machine learning. For classification, the QNN consists of $n+1$ qubits where n is the the number of data qubits (that encode the classical data) and an extra qubit is added as the readout qubit q_r, whose measurement is equivalent to the classification output.

This particular variational quantum circuit has the unitaries in the form $e^{i\theta \sum}$ where $\sum$ is a tensor product of operators from the Pauli matrices set: $\{\sigma_x, \sigma_y, \sigma_z\}$ acting on multiple qubits. In the Tensorflow Quantum example [133], they choose $\sum$ to be XX and ZZ in which the second qubit is the readout qubit and the first qubit is one of the other qubits. XX is the tensor

product of two X gates. In particular

$$XX = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}.$$

They construct layers of the variational quantum circuit. Each layer consists of n instances of the same gate, where n is the number of data qubits in the circuit. Figure 3.8 shows an example of this variational quantum circuit for $n = 4$. We can see that qubit q_0 is the readout qubit, as it is always being acted upon and qubits q_1, q_2, q_3, q_4 are the data qubits.

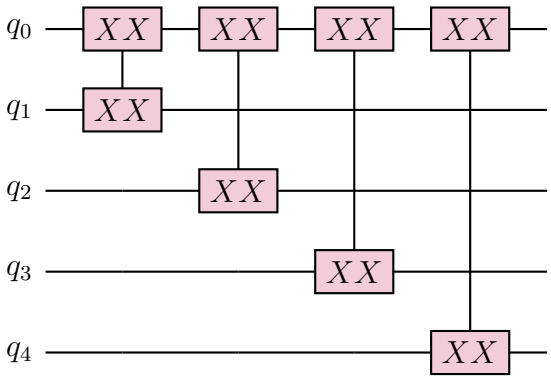


Figure 3.8: **Variational quantum circuit.** This circuit includes the XX gate, which is the $e^{i\theta X \otimes X}$ two-qubit gate. Figure inspired by Reference [133].

3.4.3 Measurement Operators

After the variational quantum circuit, we need to make measurements in order to gain information so that we can evaluate the loss function. In particular, on the readout qubit, we measure a Pauli operator, such as $\sigma_z = \left(\begin{smallmatrix} 1 & 0 \\ 0 & -1 \end{smallmatrix}\right)$, which we can call Z_{n+1}. This gives $+1$ or -1. We have to run the circuit multiple times to get the useful statistics from the circuit; in particular we want the expectation value of Z_{n+1}. The expectation value of an operator A with respect to a state ψ is $\langle\psi| A |\psi\rangle$. Our state ψ in the variational circuit is equivalent to the unitary matrix of the circuit $U(\vec{\theta})$ applied to the initial state of the circuit $|z, 1\rangle$, where $|z\rangle = |z_1 z_2 \cdots z_n\rangle$ is the conversion of the classical data string $z = z_1 z_2 \cdots z_n$ and $|1\rangle$ is the state of the readout qubit, which gives us the initial state of the circuit $|z, 1\rangle$. Thus $|\psi\rangle = U(\vec{\theta}) |z, 1\rangle$ and the expectation of the measurement operator is $\langle z, 1| U^\dagger(\vec{\theta}) Z_{n+1} U(\vec{\theta}) |z, 1\rangle$. This is equivalent to the average of the observed outcomes of Z_{n+1}. Our loss function is shown in Equation 3.1.

$$\text{loss}(\vec{\theta}, z) = 1 - l(z) \langle z, 1| U^\dagger(\vec{\theta}) Z_{n+1} U(\vec{\theta}) |z, 1\rangle , \tag{3.1}$$

where $l(z)$ is the correct label of the input string.

The expectation value of the measurement operator should correspond to the correct label $l(z)$ of the input string. In order to get our predicted label (the expectation value of the measurement operator) as close as possible to the true label, we have to find the parameters $\vec{\theta}$ in the variational quantum circuit that will get us there.

3.4.4 Learning

The learning procedure involves fitting the neural network parameters to the training set. The training set is typically a subset of the dataset we wish to classify with the neural network. The training set should be ideally as close as possible to the overall dataset so that the neural network that fits the training set can more accurately predict the properties of the overall dataset. The learning procedure is as follows:

1. Start with random parameters $\vec{\theta}$ in the variational quantum circuit.
2. Pick a string z^1 from the training set.
3. Encode the string z^1 into the circuit via the encoder circuit.
4. Measure σ_z on the readout qubit multiple times to obtain the expectation

value of Z_{n+1}.

5. Compute loss$(\vec{\theta}, z^1)$ as shown in Equation 3.1.
6. Update the parameters via some classical optimiser (such as Adam [126]), giving us new parameters $\vec{\theta^1}$.
7. Pick a new string z^2 from the training set.
8. Repeat steps 3 - 6 with z^2 to obtain $\vec{\theta^2}$.
9. Repeat steps 7 - 8 with strings z^i, where $i \in \{1, 2, \ldots, S\}$ generating a sequence $\vec{\theta^1}, \vec{\theta^2}, \ldots \vec{\theta^S}$.

The final version of the unitary matrix of the circuit $U(\vec{\theta^S})$ with parameters $\vec{\theta^S}$ acting on the state $|z, 1\rangle$ will result in measurements that will give the correct label $l(z)$ if the learning is successful.

3.4.5 Analytic Gradients

For step six in the learning process, a classical optimiser can be used to obtain the new parameters. It is also possible to evaluate the gradients using quantum circuits on quantum hardware. Schuld et al. in their paper titled *Evaluating analytic gradients on quantum hardware* [134] showed how to calculate gradients

of expectation values of quantum measurements using the same architecture that executes the quantum circuit on hardware. They show that it is sufficient in many instances to run the original quantum circuit twice while shifting a single gate parameter to obtain the corresponding component of the gradient and in general cases conditioning a single gate on an ancilla.

In variational circuits, we want to minimise (or maximise) a cost function C. If we have a variational circuit Q with parameters $\theta_1, \theta_2, ..., \theta_k$, then we can find some cost function $C(\theta_1, \theta_2, ..., \theta_k)$ from the circuit Q. In order to see how C depends on its parameters θ, we can find the analytical gradient $\frac{\partial C}{\partial \theta_j}$. Schuld et al. [134] show that

$$\frac{\partial C}{\partial \theta_j} = \frac{1}{2}\left[C\left(\theta_1, ..., \theta_j + \frac{\pi}{2}, ..., \theta_k\right) - C\left(\theta_1, ..., \theta_j - \frac{\pi}{2}, ..., \theta_k\right)\right]. \tag{3.2}$$

We can simplify this as

$$\frac{\partial C(\theta)}{\partial \theta_j} = \frac{1}{2}\left[C\left(\theta_j + \frac{\pi}{2}\right) - C\left(\theta_j - \frac{\pi}{2}\right)\right]. \tag{3.3}$$

Equation 3.3 shows how to obtain the gradient of the cost function with respect

to a parameter by shifting the parameter in the quantum circuit and calculating the cost function from the result of these parameter-shifted quantum circuits. This gradient can then be used during the optimisation process.

3.5 Quantum Kernels

Kernel methods map the input space of data to a higher-dimensional space (a feature space) so that a learning algorithm can learn linear patterns in that feature space. In classical machine learning, the first kernel learner was invented in 1964 by Aizerman, Braverman, and Rozonèr [135]. Years later, in 1992, Bernhard Boser, Isabelle Guyon and Vladimir Vapnik developed Support Vector Machines (SVMs) [136], which classify data by finding an optimal hyperplane (a decision boundary) that maximises the margin between the classes of the data; this margin is the distance between the hyperplane and the nearest data points from each class. SVMs increased the popularity of kernel methods because SVMs could become non-linear classifiers (i.e. producing non-linear decision boundaries for classifying non-linear data) by mapping the input features into a higher-dimensional space where the classes can be separated with a linear decision boundary and using the *kernel trick*, which uses a kernel

function to compute the dot product between the input data points in the higher-dimensional space. The kernel trick is less computationally expensive than mapping the data points directly to the higher-dimensional space. As of now, kernels are used in machine learning, but deep neural networks are more prominent as they scale better than kernel methods.

Figure 3.9 shows a visual example of data points in a lower-dimensional space that can not be separated linearly into classes; by applying a feature map to map the data points into a higher-dimensional space they become linearly separable.

Transitioning to quantum computing, according to a review of *Quantum Machine Learning* by Biamonte et al. [4], the first quantum support vector machine was mentioned in 2003 in a paper titled *Quantum optimization for training support vector machines* by Anguita et al. [96]. Years later, Rebentrost et al. in a paper titled *Quantum Support Vector Machine for Big Data Classification* [102] showed that the "support vector machine can be implemented on a quantum computer with complexity logarithmic in the size of the vectors and the number of training examples". There were other early works that mentioned

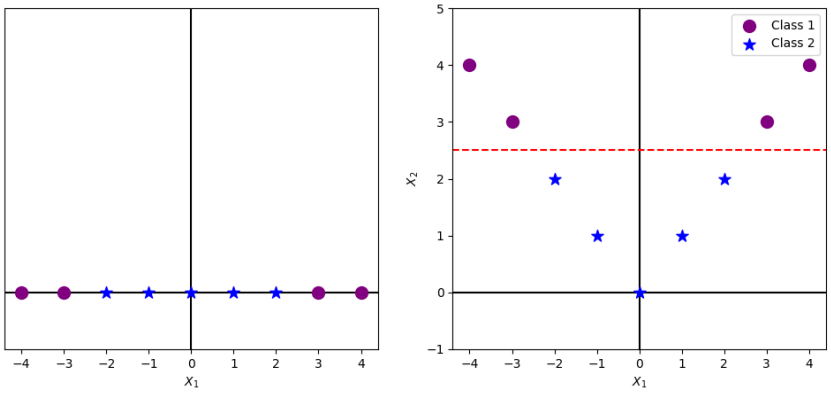


Figure 3.9: **Linear decision boundary with a feature map.** This figure shows an example of how a feature map can enable a linear decision boundary for the data. The left plot shows the original data points in 1-dimension; in 1-dimension the classes for these data points are not linearly separable. One can apply the feature map $\Phi(x) = |x|$ to the data points, which takes the absolute value of the data points and maps them to 2-dimensions - shown in the right plot. The classes for these data points are now linearly separable in 2-dimensions. These figures were inspired by Reference [137].

kernel methods for quantum computing [138, 139].

Schuld and Killoran then explored the theoretical link between kernel methods and quantum computing: that is, just like kernel methods, quantum computing also maps information to a high-dimensional inaccessible space (the Hilbert space) and efficiently performs calculations in this space [26]. Schuld expanded upon this and extended the idea of rephrasing supervised quantum models as kernel methods [125]. Schuld defines a *quantum model* as the algorithm

that runs on a quantum computer which acts as a classifier or generator. In particular, quantum models consist of the data encoding part, which maps data inputs to quantum states, and the measurement of the quantum circuit; therefore they can be referred to as QNNs. Schuld states that "a lot of quantum models can be replaced by a general support vector machine whose kernel computes distances between data-encoding quantum states" [125]. This work demonstrated that quantum models are different from classical machine learning models because of the way that data is encoded into quantum states.

As a result, one of the advantages of studying and using quantum kernels is that they can also give us insight into quantum models, i.e. QNNs, and how they are different from classical machine learning models. Given that there are analytical descriptions of kernels, there are clearer expectations of how they behave and their performance, compared to QNNs. This enables researchers to provide provable guarantees on their performance, compared to QNNs, which are difficult to make proofs about.

One guarantee is that training kernels finds the globally optimal measurement over all possible quantum models. If the loss is convex, meaning that there is

only one global minimum and no local minima, then the kernel can find the optimum cost. This is in comparison to training QNNs, which provides no guarantee as the QNN "optimises over a subspace defined by the parametrised ansatz which may or may not overlap with the training-data subspace and could therefore not have access to the global optimum" [125]. The optimisation is usually non-convex when training QNNs, meaning that global and local minima exist, thereby making the optimisation more difficult.

Furthermore, the performance of QNNs depends on the ansatz chosen and they suffer from barren plateaus, which refers to gradients that vanish exponentially in the number of qubits. An advantage of quantum kernels is that they only depend on data encoding and therefore they avoid the choice of an ansatz and they avoid the processing and measurement parts of QNNs, which thus means they do not suffer from barren plateaus. As quantum kernels only depend on data encoding, the quantum circuits evaluated for calculating quantum kernels are shallower than those of QNNs, providing another potential benefit for quantum kernels, depending on the number of quantum circuit evaluations compared to QNNs.

One may wonder with all of the benefits of quantum kernels why they are not used completely instead of QNNs. One of the main challenges is that one has to compute pairwise distances of data points, which can be computationally expensive. As Schuld has pointed out [125], the runtime for training quantum kernels is $O(M^2)$ for M training samples. For classical neural networks, training takes runtime $O(M)$. For QNNs, one uses the parameter-shift rules which adds a parameter factor to the scaling $O(|\theta|M)$. One can see that if the number of parameters scales less than linearly, then QNNs will scale the same as classical neural networks and be more advantageous than quantum kernels. If, however, just like classical neural networks, the number of parameters scales linearly with the training samples, QNNs will also have the $O(M^2)$ runtime like quantum kernels and quantum kernels will be much more advantageous.

One interesting thing to note is that fault-tolerant quantum computers could allow us to train kernel methods in $O(M)$, both for classical and quantum models. Therefore, quantum computers have the potential to enable training kernels at a larger scale and reap forth its benefits, making it a stronger alternative to neural networks - both classical and quantum.

In the literature, quantum kernels have been used to find cases for quantum advantage [27, 28]. For example, Huang et al. [28] used quantum kernels to come up with bounds for when a quantum kernel has an advantage over a classical kernel. In addition, quantum kernels have been used to perform machine-learning tasks for particular applications [140, 141]. Quantum kernels have been implemented on quantum hardware, including optical hardware [142].

3.5.1 How to Use a Quantum Kernel

This section provides further details on the calculations for quantum kernels, which is used for investigating the inductive bias in Chapter 6. There are three main steps to using a quantum kernel:

1. Define a quantum feature map.
2. Construct the kernel matrix.
3. Train the kernel.

3.5.1.1 Define a Quantum Feature Map

A kernel function k maps the input data into a higher-dimensional space $k(\vec{x}_i, \vec{x}_j) = \langle f(\vec{x}_i), f(\vec{x}_j) \rangle$, where $\vec{x}_i, \vec{x}_j$ are n dimensional inputs, f is a map from an n-dimension space to an m-dimension space, and $\langle a, b \rangle$ denotes the inner product between a and b. A kernel matrix K can be constructed from all the input data $K_{ij} = k(\vec{x}_i, \vec{x}_j)$. A kernel function measures how similar two data points are in the higher-dimensional feature space. The larger the kernel value, the more similar the data points in the feature space.

A quantum feature map $\phi(\vec{x})$ maps classical data, represented as a vector $\vec{x}$, to a quantum Hilbert space and the Kernel matrix is $K_{ij} = \left| \langle \phi^\dagger(\vec{x}_j) | \phi(\vec{x}_i) \rangle \right|^2$. The quantum feature maps are described as encoding methods in Section 3.4.1.

3.5.1.2 Construct the Kernel Matrix

To obtain the Kernel matrix from a QNN, we encode the x_i data into the quantum circuit that encodes the data and then obtain the final statevector $|\phi\rangle$. We can then compute $\left| \langle \phi^\dagger(\vec{x}_j) | \phi(\vec{x}_i) \rangle \right|^2$ for all pairs of i, j.

For example, we can compute the kernel matrices on the Boolean dataset for $n = 2$ for different encoding methods. The Boolean dataset for $n = 2$ is equivalent to $\vec{x} = (x_0, x_1, x_2, x_3)$, where $x_0 = 00, x_1 = 01, x_2 = 10, x_3 = 11$.

Figure 3.10 shows the kernel matrices for each encoding method for two qubits ($n = 2$). It should be noted that amplitude encoding can not encode the all 0s Boolean function. Therefore, its kernel matrix has dimensions $(2^n - 1 \times 2^n - 1)$ instead of $(2^n \times 2^n)$. In the following sections, we explain how to calculate each kernel matrix using its encoding method.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(a) Basis encoding

$$\begin{bmatrix} 1 & 0.167 & 0.167 & 0.325 \\ 0.167 & 1 & 0.043 & 0.254 \\ 0.167 & 0.043 & 1 & 0.254 \\ 0.325 & 0.254 & 0.254 & 1 \end{bmatrix}$$

(b) ZZ Feature Map

$$\begin{bmatrix} 1 & 0.206 & 0.255 & 0.600 \\ 0.206 & 1 & 0.376 & 0 \\ 0.255 & 0.376 & 1 & 0.036 \\ 0.600 & 0 & 0.036 & 1 \end{bmatrix}$$

(c) Random relu transform

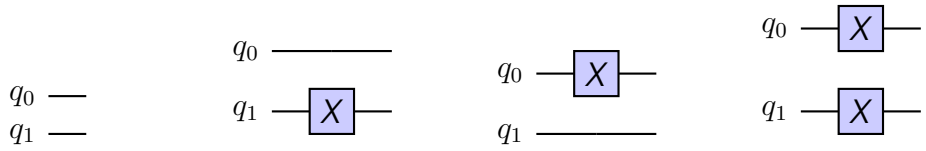
$$\begin{bmatrix} 1 & 0 & 0.5 \\ 0 & 1 & 0.5 \\ 0.5 & 0.5 & 1 \end{bmatrix}$$

(d) Amplitude encoding

Figure 3.10: **Kernel matrices for n = 2.** The kernel matrices calculated using $K_{ij} = \left|\langle\phi^\dagger(\vec{x}_j)|\phi(\vec{x}_i)\rangle\right|^2$, where $\vec{x}_i$ consist of all the combinations of binary strings of length two.

3.5.1.3 Basis Encoding Kernel Matrix

To generate the basis encoding kernel matrix, we construct the quantum circuits that encode this boolean dataset. Figure 3.11 shows the four quantum circuits to encode $\vec{x} = (x_0, x_1, x_2, x_3)$.



(a) $x_0 = 00$ (b) $x_1 = 01$ (c) $x_2 = 10$ (d) $x_3 = 11$

Figure 3.11: **Quantum circuits for basis encoding.** These quantum circuits encode the data using basis encoding. (a) This circuit which consists of no quantum gates (equivalent to Identity gates) encodes the data $x_0 = 00$ as the qubits q_0 and q_1 are initially in the state $|0\rangle |0\rangle$. (b) This circuit consists of no gate on the q_0 qubit and applies the X gate to the qubit q_1 as it flips the state of this qubit to $|1\rangle$ and therefore encodes the data $x_1 = 01$. (c) This circuit applies the X gate to the qubit q_0 as it flips the qubit to the state $|1\rangle$ and does nothing to the qubit q_1 and therefore encodes $x_2 = 10$. (d) This circuit applies the X gate to both of the qubits q_0 and q_1 so that it encodes the data $x_3 = 11$.

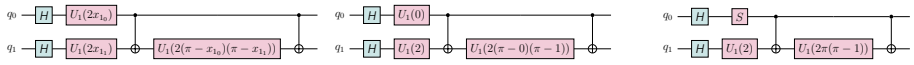
We can now obtain the statevector from the encoding quantum circuits. We calculate the statevector as $|\phi\rangle = U |0^n\rangle$, where U is the unitary matrix of the circuit and $|0^n\rangle$ is applied to n qubits. For Figure 3.11a, the unitary matrix of the circuit is equivalent to $I \otimes I$, therefore the statevector is $|\phi\rangle = |00\rangle$. For Figure 3.11b, the unitary matrix of the circuit is equivalent to $I \otimes X$, therefore

the statevector is $|\phi\rangle = |01\rangle$. For Figure 3.11c, the unitary matrix of the circuit is equivalent to $X \otimes I$, therefore the statevector is $|\phi\rangle = |10\rangle$. For Figure 3.11d, the unitary matrix of the circuit is equivalent to $X \otimes X$, therefore the statevector is $|\psi\rangle = |11\rangle$.

We can now calculate the entries of the kernel matrix as $K_{ij} = \left|\langle\phi^\dagger(\vec{x}_j)|\phi(\vec{x}_i)\rangle\right|^2$ for all possible i, j. For example, when $i = 0$ and $j = 0$, we calculate $K_{00} = \left|\langle\phi^\dagger(\vec{x}_0)|\phi(\vec{x}_0)\rangle\right|^2 = |\langle 00|\,|00\rangle|^2 = 1$. When i = 0 and j = 1, we calculate $K_{01} = \left|\langle\phi^\dagger(\vec{x}_1)|\phi(\vec{x}_0)\rangle\right|^2 = |\langle 01|\,|00\rangle|^2 = 0$. We repeat this for all possible i, j and then construct the kernel matrix from its entries. For basis encoding, the matrix becomes that shown in Figure 3.10a.

3.5.1.4 ZZ Feature Map Kernel Matrix

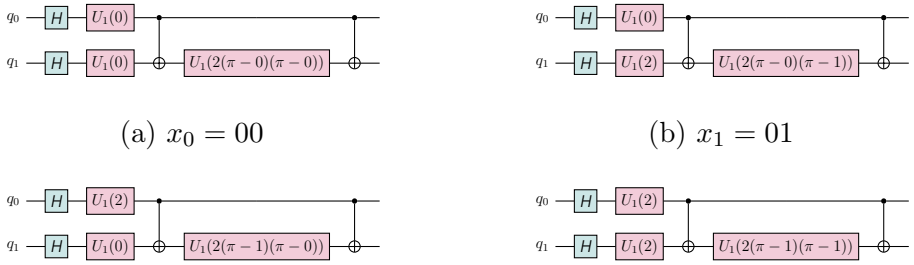
To generate the ZZ feature map kernel matrix, we construct the quantum circuits that encode this boolean dataset. Figure 3.12 shows how we encode the data into the quantum circuit and obtain a simplified form of the quantum circuit that encodes the data.



(a) (b) (c)

Figure 3.12: **ZZ Feature Map for two qubits that encodes 01.** This figure shows the quantum circuit that encodes the data $x_1 = 01$: the 0th element of x_1 is $x_{1_0} = 0$ and the 1st element of x_1 is $x_{1_1} = 1$. In Figure 3.12a, the quantum circuit consists of two H (Hadamard) gates, where $H = \frac{1}{\sqrt{2}}\left(\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right)$ followed by two U_1 gates applied to both qubits, where $U_1(\theta) = \left(\begin{smallmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{smallmatrix}\right)$. The angle θ of the U_1 gate applied to the first qubit q_0 is $2x_{1_0}$ which is equal to 0 in this example. Therefore this matrix is equivalent to $U_1(0) = \left(\begin{smallmatrix} 1 & 0 \\ 0 & e^{i0} \end{smallmatrix}\right) = \left(\begin{smallmatrix} 1 & 0 \\ 0 & i \end{smallmatrix}\right) = S$, where S is the phase matrix. The angle θ of the U_1 gate applied to the second qubit q_1 is $2x_{1_1}$ which is equal to $2 \cdot 1 = 2$ in this example. Therefore this matrix is equivalent to $U_1(2) = \left(\begin{smallmatrix} 1 & 0 \\ 0 & e^{2i} \end{smallmatrix}\right)$. These U_1 gates are followed by a CX gate and then another U_1 gate applied to q_1, where the angle of this gate is equivalent to $2(\pi - x_{1_0})(\pi - x_{1_1})$. This gate is equivalent to the matrix $\left(\begin{smallmatrix} 1 & 0 \\ 0 & e^{i(2(\pi - x_{1_0})(\pi - x_{1_1}))} \end{smallmatrix}\right) = \left(\begin{smallmatrix} 1 & 0 \\ 0 & e^{i(2\pi(\pi-1))} \end{smallmatrix}\right)$. Figures 3.12b and 3.12c show the steps of replacing the angles with the data points and simplifying the form of the quantum circuit.

Figure 3.13 shows the four quantum circuits to encode $\vec{x} = (x_0, x_1, x_2, x_3)$.



(a) $x_0 = 00$ (b) $x_1 = 01$

(c) $x_2 = 10$ (d) $x_3 = 11$

Figure 3.13: **Quantum circuits for the ZZ feature map.** These quantum circuits encode the data using the ZZ feature map.

Using the same method as we described for basis encoding (Section 3.5.1.3), we

obtain the statevector from the encoding quantum circuits and then calculate the entries of the kernel matrix. For the ZZ feature map, the matrix becomes that shown in Figure 3.10b.

3.5.1.5 Random Relu Transform Kernel Matrix

In classical machine learning, a *Rectified Linear Unit* (ReLU) is an activation function, which introduces non-linearity to a learning model. ReLU is defined as the positive part of its argument - in other words, its output is the input if it is positive, otherwise, its output is zero as shown in Figure 3.14 and Equation 3.4

$$f(x) = x^+ = \max(0, x) = \frac{x + |x|}{2} = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{3.4}$$

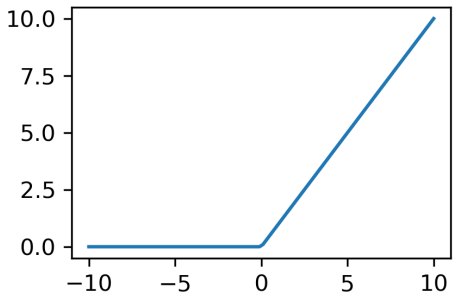


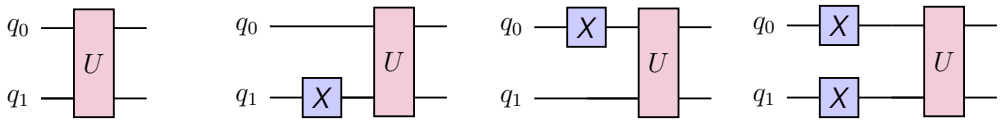
Figure 3.14: **ReLU function.** The ReLU function returns the input if it is positive, otherwise it returns zero.

For this encoding method, we apply the ReLU function to the statevector resulting from the encoding circuit. The encoding circuit consists of encoding the Boolean data using basis encoding and then applying a random unitary matrix to the qubits. Given that our statevector is a complex vector, we apply the ReLU function to the imaginary and real parts of the complex number. After applying the ReLU function to the statevector, we then re-normalise the statevector. Table 3.2 shows the input and ReLU output with a complex number.

To generate the *random relu transform*, we construct the quantum circuits that encode this Boolean dataset. Figure 3.15 shows the four quantum circuits to encode $\vec{x} = (x_0, x_1, x_2, x_3)$.

Input		**Output**
Re(x)	**Im**(x)	**ReLU**(x)
< 0	< 0	$0 + 0j$
< 0	≥ 0	$0 + \text{Im}(x)j$
≥ 0	< 0	$\text{Re}(x) + 0j$
≥ 0	≥ 0	$\text{Re}(x) + \text{Im}(x)j$

Table 3.2: **ReLU function.** The ReLU function returns the input if it is positive, otherwise it returns zero. When applied to a complex number, we apply ReLU to the imaginary and real parts of the complex number.



(a) $x_0 = 00$ (b) $x_1 = 01$ (c) $x_2 = 10$ (d) $x_3 = 11$

Figure 3.15: **Quantum circuits for random relu transform.** These quantum circuits encode the data using the random relu transform. The first step of the random relu transform is encoding the boolean data using basis encoding and then applying a random unitary matrix to the qubits. After the statevector of the quantum circuits are obtained, the ReLU function is applied to the statevector.

To summarise, the random relu transform includes the following steps:

1. Construct a quantum circuit that uses basis encoding to encode the data.
2. Apply a random unitary matrix.
3. Obtain the statevector from the encoding quantum circuit.
4. Apply the ReLU function to the statevector.
5. Normalise the resulting statevector.

From this final statevector, we then calculate the entries of the kernel matrix.

For the random relu transform, the matrix becomes that shown in Figure 3.10c.

In practice, this encoding method is inefficient as it requires an exponential number of resources to obtain the statevector.

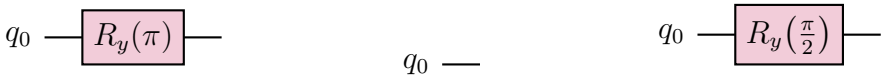
3.5.1.6 Amplitude Encoding Kernel Matrix

Amplitude encoding encodes the data into the amplitudes of the quantum state, so that our quantum state becomes $|\psi\rangle = \frac{1}{\|x_p\|}\sum_{i=0}^{n-1} x_{p_i}|i\rangle$, where our input data is $x_p = x_{p_{n-1}}x_{p_{n-2}}\cdots x_{p_0}$ and x_{p_i} is the ith element of x_p and x_p is the pth element of $\vec{x} = (x_0, x_1, x_2, x_3)$ and $|i\rangle$ is the ith computational basis state.

Given that the statevector of n qubits contains 2^n amplitudes, we only need $\log_2(n)$ qubits to encode data with length n. When $n = 2$, this means we only need one qubit as $\log_2(2) = 1$. When $n = 2$, we can encode the data into one qubit as $x_{p_0}\cdot|0\rangle + x_{p_1}\cdot|1\rangle$. When $p = 1$, our input data is $x_1 = 01$. Therefore our statevector would be $0\cdot|0\rangle + 1\cdot|1\rangle = |1\rangle$ as $x_{1_0} = 0$ and $x_{1_1} = 1$. When $p = 2$, our input data is $x_2 = 10$ and our statevector would be $1\cdot|0\rangle + 0\cdot|1\rangle = |0\rangle$. When $p = 3$, our input data is $x_3 = 11$ and our statevector would be $1\cdot|0\rangle + 1\cdot|1\rangle$. This is an invalid statevector so we need to normalise the input data. Therefore, when $p = 3$, our input data is $x_3 = \frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}}$ and our statevector would be $\frac{1}{\sqrt{2}}\cdot|0\rangle + \frac{1}{\sqrt{2}}\cdot|1\rangle$. When $p = 0$, our input data is $x_0 = 00$, but this would be equivalent to the statevector $0\cdot|0\rangle + 0\cdot|1\rangle$, which is invalid. Therefore, we can not encode the all 0s Boolean input data with amplitude

encoding. As a result, with amplitude encoding, we can only have $2^n - 1$ input data points instead of 2^n. This means the kernel matrix for it has dimensions $(2^n - 1 \times 2^n - 1)$ instead of $(2^n \times 2^n)$.

To generate the amplitude encoding, we construct the quantum circuits that encode this Boolean dataset. Figure 3.16 shows the three quantum circuits to encode $\vec{x} = (x_0, x_1, x_2, x_3)$ but x_0 is not possible to encode.



(a) $x_1 = 01$ (b) $x_2 = 10$ (c) $x_3 = 11$

Figure 3.16: **Quantum circuits for amplitude encoding**. These quantum circuits prepare the statevectors in which the amplitudes are equivalent to the input data. (a) Encode the data $x_1 = 01$, so the statevector needs to be $0 \cdot |0\rangle + 1 \cdot |1\rangle = |1\rangle$. To get this statevector, we apply the $R_y(\pi)$ gate to the qubit, where $R_y(\theta) = \begin{pmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{pmatrix}$. Therefore, $R_y(\pi) = \begin{pmatrix} \cos(\pi/2) & -\sin(\pi/2) \\ \sin(\pi/2) & \cos(\pi/2) \end{pmatrix} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$. When we apply $R_y(\pi)$ to the qubit, we obtain $\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle$. Therefore the final statevector is $|1\rangle$ which is equivalent to encoding the data $x_1 = 01$ into the state. (b) Encode the data $x_2 = 10$, so the statevector needs to be $1 \cdot |0\rangle + 0 \cdot |1\rangle = |0\rangle$. Given that the qubits are initialised to $|0\rangle$, we do not need to apply any quantum gates to obtain this statevector. (c) The data is $x_3 = 11$, which we normalise to $x_3 = \frac{1}{\sqrt{2}} \frac{1}{\sqrt{2}}$ so the statevector needs to be $\frac{1}{\sqrt{2}} \cdot |0\rangle + \frac{1}{\sqrt{2}} \cdot |1\rangle$. To get this statevector we apply the $R_y(\frac{\pi}{2})$ gate to the qubit, where $R_y(\frac{\pi}{2}) = \begin{pmatrix} \cos(\pi/4) & -\sin(\pi/4) \\ \sin(\pi/4) & \cos(\pi/4) \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}$. When we apply $R_y(\frac{\pi}{2})$ to the qubit, we obtain $\begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} = \frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle$. Therefore this encodes $x_3 = 11$ into the state.

After obtaining the statevector from the encoding quantum circuits, we calculate the entries of the kernel matrix using the method described in the section for basis encoding (Section 3.5.1.3). For amplitude encoding, the matrix becomes that shown in Figure 3.10d.

3.6 Characteristics of QNNs from the Literature

As variational quantum algorithms became the paradigm of QNNs, there was excitement to use this new framework, but there was still uncertainty about their use, potential, and limitations. Therefore, new research emerged to analyse and understand the properties of these QNNs. One of the most notable papers to analyse QNNs and provide a sobering message to the community was in 2018 when McClean et al. discovered barren plateaus in QNN training landscapes [25].

3.6.1 Barren Plateaus

McClean et al. [25] showed that for a "large class of random circuits, the average value of the gradient of the objective function is zero, and the probability that any given instance of such a random circuit deviates from this average value

by a small constant ϵ is exponentially small in the number of qubits" [25]. In classical neural networks, there is the vanishing gradient problem, where the gradient can vanish exponentially in the number of layers. In contrast, in QNNs, the gradient can vanish exponentially in the number of qubits.

This means that it is difficult to train QNNs effectively, because during the training process, the gradients tend towards zero and therefore gradient-based optimisation methods can barely find the direction of descent. Not only that, but the probability of encountering these barren plateaus increases as the number of qubit increases, which hinders the scalability of QNNs.

This research paper was significant as it revealed a major challenge of QNNs and spurred further research into analysing the properties of QNNs and ways to overcome this challenge. One way to mitigate this issue is to use structured variational quantum circuits, not random variational quantum circuits. These structured variational quantum circuits are tailored towards the specific problem, whereas random variational quantum circuits are general and do not have a structure (i.e. the gates chosen and their arrangement) that is specific to the problem. The reduced expressivity of structured variational circuits can help

mitigate the occurrence of barren plateaus [143].

Another approach is to use structured initial guesses for the parameters, not random parameter initialisation [144]. In addition, training QNNs a few layers at a time has been shown to reduce barren plateaus [145]. Additionally, one can use an encoding circuit that is harder to simulate classically, as it has been shown that this shows resilience to the barren plateau phenomenon [29]. Furthermore, using local cost functions, which only use measurements from a few qubits, instead of global cost functions, which use measurement information from all qubits, can alleviate barren plateaus [146].

As researchers were investigating the limitations of QNNs, they were simultaneously (in true quantum fashion) investigating the potential of QNNs, in particular trying to understand where QNNs have an advantage over classical neural networks. There are several metrics by which to measure quantum advantage in QNNs, including expressivity, training speed, and generalisation performance.

3.6.2 Model Capacity

Wright et al. in August 2019 [147] tried to understand quantum advantage by analysing the expressive power of parametrised quantum circuits using memory capacity, a measure of the expressive power of a QNN that they adapted from classical neural network theory. The model's capacity determines the model's ability to fit various functions. A higher capacity could potentially lead to a better fit to the data. They showed that the capacity C of a QNN is bounded by the information W that can be trained into its parameters $C \leq W$. This means that QNNs that are parametrised classically do not have an advantage in capacity over classical neural networks with the same number of parameters, as the bound $C \leq W$ applies universally, regardless of the learning machine and its features. QNNs parametrised with quantum states, however, could have exponentially larger capacities. They acknowledge that even though these classically-parametrised QNNs have limited advantages over classical neural networks in terms of capacity, there may still be advantages in other metrics, such as training speed or performance on specific tasks.

This paper was important because it was one of the early works to adapt the

notion of model capacity to QNNs. It also suggested that there was little quantum advantage with respect to capacity for classically-parametrised QNNs. This paper motivated researchers to explore other ways to measure capacity.

Abbas et al. in October 2020 [29] proposed a new way to measure the capacity of models, by using the *effective dimension*, which is data-dependent and depends on the Fisher Information, which captures the sensitivity of a neural network's output relative to movements in the parameter space. The effective dimension tells us how many of the models parameters are actively being used in the neural network and are therefore not redundant. One advantage of the effective dimension is that it takes into account the data distribution, whereas the memory capacity used in Wright et al.'s paper [147] did not. Furthermore, the effective dimension can be directly related to the generalisation performance of the model, which can provide better understanding of quantum advantage with respect to this metric. The effective dimension has been used in classical neural networks; Abbas et al. were the first to apply it to QNNs.

They show numerically that there is a class of QNNs that achieves a higher effective dimension than comparable classical neural networks. In particular,

this class of QNNs includes an encoding circuit proposed in the paper titled *Supervised learning with quantum-enhanced feature spaces* by Havlicek et al. [127], whereas the class of QNNs that do not achieve a better effective dimension include an encoding circuit that is easy to simulate classically.

This paper was important [148, 29] because it proposed a novel way, the effective dimension, to capture the capacity of any model. Using this measure, they showed numerically the possibility for quantum advantage with QNNs. They connected this measure to the model's generalisability and trainability and suggested quantum advantage for trainability, but this was later shown to not necessarily be a quantum advantage because they used a weaker classical neural network for comparison [30]. At the time of their publication, this paper boosted the field's confidence in QNN's ability for an advantage.

3.6.3 Role of Data

Shortly after, in November 2020, Huang et al. [28] shed more light on the possibilities of quantum advantage by analysing quantum kernels and examining the role that data plays in quantum and classical models. This paper brought

together several concepts that were circulating in the field around that time, including the use of quantum kernels [26], the role of data in the performance of models [29], tools to analyse quantum advantage [147, 29], and the excitement and questions around the quantum supremacy experiment by Google [85, 84].

3.6.3.1 Classical Models Can Be Elevated through Data

Huang et al. in their paper titled *Power of Data in Quantum Machine Learning* [28] shed light on the types of problems where QNNs may not necessarily outperform classical neural networks. In particular, one motivation for using QNNs was that quantum circuits can easily produce distributions that are hard to produce classically. This was demonstrated in the quantum supremacy by Google [85] in which they showed that a classical computer could not efficiently simulate the distributions that were produced by a quantum circuit run on their superconducting quantum hardware.

As a result, people speculated that QNNs could possibly easily recognise patterns in such distributions, whilst classical neural networks would have difficulty in doing this. Huang et al., however, showed that this is not always necessarily the case. Even if some distributions are classically hard to compute,

they showed that they can be easily predicted by classical machines learning from data. Thus providing data for classical models can elevate classical models to rival quantum models even when the quantum circuits generating the data are hard to compute classically. They show this via a motivating example and complexity arguments.

3.6.3.2 Error Bounds

Huang et al. show prediction error bounds (which are theoretical limits of the model's worst-case error, which is the maximum difference between the model's predicted outcomes and the actual outcomes) for training classical and quantum ML methods based on kernel functions. From these bounds, they create a test to determine potential quantum advantage by looking at the difference between prediction errors of quantum and classical models for a fixed amount of training data. The test is a geometric difference between kernel functions, where they define the geometric difference by the closest efficient classical machine learning model. For a fixed amount of training data, one can calculate the geometric difference between kernel functions. A quantum advantage can be obtained if the geometric difference between a

classical and quantum kernel is large. This geometric difference g is as follows $g_{12} = g\left(K^1 \| K^2\right) = \sqrt{\left\| \sqrt{K^2}\left(K^1\right)^{-1}\sqrt{K^2}\right\|_\infty}$, where K^1 and K^2 are the kernel matrices for comparison and $\|\cdot\|_\infty$ is the spectral norm of the matrix and the assumption is $\mathrm{Tr}(K^1) = \mathrm{Tr}(K^2) = N$, where N is the number of training examples. This geometric difference g depends on the dataset as the kernels K are constructed from the dataset. To determine quantum advantage between classical and quantum models, the classical kernel can be equal to K^1 and the quantum kernel can be equal to K^2.

To show how this geometric difference connects to quantum advantage, they define a quantum model as $f(x) = \mathrm{Tr}(O^U \rho(x))$ where $\rho(x) = |\phi(x)\rangle\langle\phi(x)|$ and $\phi(x)$ is the feature map, mapping x to a higher-dimensional space. $O^U = U^\dagger_{\mathrm{QNN}} O U_{\mathrm{QNN}}$ where O is an observable and U_{QNN} is a unitary matrix.

They show that the model complexity of the trained model is $s_K(N) = \sum_{i=1}^N \sum_{j=1}^N (K^{-1})_{ij} \mathrm{Tr}(O^U \rho(x_i)) \mathrm{Tr}(O^U \rho(x_j))$. A smaller value of $s_K(N)$ implies better generalisation performance, as $s_K(N)$ is measuring whether the similarity between x_i, x_j as defined by the kernel function $k(x_i, x_j)$ (when two points are similar, the kernel value is larger) aligns well with the similarity of

the observable expectation of $\rho(x_i), \rho(x_j)$.

One can then show that $s_{K^1} \leq g_{12}^2 s_{K^2}$. If $K^1 = K^C$ is the classical model and $K^2 = K^Q$ is the quantum model, then our comparison between quantum and classical models is $s_{K^C} \leq g_{CQ}^2 s_{K^Q}$. If g_{CQ} is small, then the classical model will have a similar or better model complexity $s_K(N)$ compared to the quantum model, implying that the generalisation performance of the classical model is similar or better. If g_{CQ} is large, however, then there exists a dataset where $s_{K^C} = g_{CQ}^2 s_{K^Q}$, meaning that the quantum model would have a better generalisation performance compared to the classical model. Huang et al. construct such a dataset and show numerical results of this quantum advantage.

3.6.3.3 Projected Kernels

In order to obtain a greater geometric difference between classical and quantum kernels, Huang et al. show that one must project the quantum states, produced from encoding the classical data, back into the classical space, as the large Hilbert space from quantum kernels reduces the geometric difference.

3.6.3.4 Constructed Datasets

Huang et al. look at what type of data is needed to get a QNN to outperform a classical neural network. It is unclear whether QNNs will be able to outperform classical neural networks on classical data. QNNs, however, can provide a quantum advantage compared to classical neural networks on quantum data. They constructed a dataset to maximise the geometric difference between classical and quantum kernels and they showed numerical results that demonstrate a quantum advantage. In particular, Huang et al. modified the *fashion MNIST* dataset (which are images of clothing items such as dresses, t-shirts, and trousers) into a quantum dataset and engineered it so that the QNN would intentionally beat the classical neural network. Therefore, they showed that "datasets that are easy for quantum models to learn and hard for classical models to learn do exist, regardless of model architecture or training algorithms used" [28]. A tutorial on how to engineer such a dataset is in Reference [149].

This paper was important [86] because it challenged previous assumptions about which tasks quantum machine learning may be good for: in particular, just because a distribution is hard to simulate classically does not mean a classical

neural network will find it difficult to learn such a distribution, because when the classical neural network has access to data, it can become more powerful. Furthermore, the paper developed a novel test, the geometric difference test, to assess potential quantum advantage and proposed a novel technique, projected quantum kernels, for obtaining a quantum advantage. Additionally, they numerically showed quantum advantage on a constructed dataset and it was one of the first papers to demonstrate numerically the largest quantum advantage and with the largest simulation (up to 30 qubits). Their paper highlighted the importance of data in obtaining a quantum advantage. The dataset they used was artificial as it was constructed for maximal quantum advantage and therefore has no utility, but their research has provided hope for finding quantum advantage for real datasets.

3.6.4 Inductive Bias in Quantum Kernels

In the spirit of using quantum kernels to assess quantum advantage, Kubler et al. in June 2021 [37] investigated the inductive bias (a model's inherent preferences for certain solutions) in quantum kernels. They showed that quantum kernel methods will fail to generalise as soon as the data embedding (i.e. mapping

the data into a feature space) into the quantum Hilbert space is too expressive. Therefore, one way to generalise well and obtain a quantum advantage is to restrict the expressivity of the quantum Hilbert space, but not in an arbitrary way, as that cannot guarantee the target function will be reflected in the smaller Hilbert space, but to embed the data in such a way that reflects the target function, which is the function that maps the input data to the true output labels and is the function that the model is trying to predict. Their results suggest that "we can only achieve a quantum advantage if we know something about the data generating process and cannot efficiently encode this classically, yet are able use this information to bias the quantum model" [37].

They also demonstrate their results via the eigenvalues and eigenvectors of the kernel matrix. They state that a target function can easily be learned if it is well aligned with the principal components (corresponding to the eigenvectors of the matrix) of a kernel, meaning that the principal components with the largest eigenvalues closely approximate the target function. If the target function is easily computed with a quantum algorithm but requires exponential resources to approximate classically, and a kernel is constructed so that it aligns well with the target function, then a quantum advantage could be achieved.

This paper is important because it discusses a new angle of quantum advantage in the context of the classical concept of inductive bias and kernel methods. Their main contribution is "the analysis of the inductive bias of quantum machine learning models based on the spectral properties of quantum kernels" [37]. (Spectral properties referring to the characteristics related to the eigenvalues and eigenvectors of the kernel matrix). This analysis provides more context for the quantum advantage obtained in other research papers: that is, they show that the projected quantum kernel in Huang et al.'s paper [28] can help provide a quantum advantage as it restricts the expressive power of the kernel and enables an inductive bias that is hard to create classically.

They note that it is still unclear how to encode a strong inductive bias about a classical dataset and they conclude that at the moment there is not yet any evidence that supervised quantum machine learning will provide an advantage on classical datasets.

3.7 Quantum Advantage

One of the primary goals of quantum computing research is to understand more

about quantum advantage, which is when a quantum computer could perform a task faster than a classical computer. The hope of quantum advantage for real-world problems is one of the main reasons why commercial companies conduct quantum computing research, such as Goldman Sachs that hopes to use quantum algorithms to perform financial calculations faster as "computing speed is a giant advantage" in the financial markets [150]. This hope is also one of the reasons why the quantum computing industry is receiving large investments for the industry's development, including, most recently on 5 February 2024, the UK government's investment of "£45 million in the UK's quantum sector - as part of its commitment to transforming into a quantum-enabled economy by 2033 – seizing this technology's potential to overhaul healthcare, energy, transport and more" [151].

Especially given the influence of quantum advantage in society, it is important to understand what quantum advantage is and whether there are actual cases of it scientifically. In the book *Machine Learning with Quantum Computers*, Schuld and Petruccione [38] identify four important markers of quantum advantage in the machine learning context:

1. **Performance:** The algorithm solves a learning task well, which means that it generalises from data samples to unseen data.
2. **Speedup:** The algorithm runs faster than classical competitors that could be used in practice.
3. **Relevance:** The algorithm solves a problem that is relevant to machine learning.
4. **Availability:** The technology to implement the algorithm has a reasonable chance to be available soon.

Today, there are no known quantum machine learning algorithms that fulfill all four criteria. Rather, there are tradeoffs between the different markers. On the one hand, fault-tolerant quantum machine learning algorithms (algorithms to be run on fault-tolerant quantum computers) can usually prove a quantum speedup for a problem, but the technology to implement the algorithm is not available yet. On the other hand, near-term quantum algorithms (algorithms that can be run on near-term quantum computers, which are the quantum devices that are available today or in the near future), including QNNs, are available for implementation on near-term quantum devices today, but it is difficult to prove a quantum speedup. As an example, the research we did in Appendix A solves a relevant problem in the mining industry and can be run on today's quantum devices (fulfilling the last two criteria), but there is no

provable quantum speedup and it is only for small-scale problems.

3.7.0.1 Quantum Speedup Cases

The basis of a quantum advantage is a quantum speedup. The field of quantum computing became popular because of a quantum speedup with societal implications (Shor's algorithm) [1]. Throughout the progression of quantum computing, various quantum speedups have been claimed theoretically, as researchers have endeavoured to create quantum algorithms that can outperform the theoretical complexity of equivalent classical algorithms. The first quantum speedups appeared as early as the 1990s with the invention of the Deutsch-Jozsa algorithm [57], the Bernstein–Vazirani algorithm [152], Simon's algorithm [59], Shor's algorithm [1], and Grover's algorithm [2] as mentioned in Chapter 2 Section 2.5.

Since then, these algorithms have been modified in various contexts to demonstrate quantum speedups, such as Reference [153] in 2018 which introduced a problem (based on the Bernstein-Vazirani problem [152]) and proved that the number of steps to solve this problem with quantum circuits was independent of the problem size, whereas this number grew for comparable classical circuits.

This paper demonstrated that it may not be necessary to have deep complex quantum circuits for a quantum speedup but shallow circuits (quantum circuits with a lower depth) could achieve a quantum speedup.

Researchers also used these traditional quantum algorithms in subroutines of quantum machine learning algorithms and other techniques to gain quantum speedups for machine learning tasks [22, 110, 111]. These approaches, however, typically require significant assumptions about loading the data [106] and it has been shown in 2019 that the quantum recommendation algorithm [103], which claimed to have an exponential quantum speedup in 2016, only has a polynomial quantum speedup when a classical algorithm also makes strong input assumptions [107]. For unsupervised learning, a lot of the quantum algorithms [109] that gain a quantum speedup are some modification of Grover's algorithm [2] as mentioned in Chapter 2 Section 2.14.1.1.

Quantum advantage has been shown for particular cases, including the processing of quantum data as demonstrated in 2022 by Huang et al. [154], in which they proved that quantum devices could learn from exponentially fewer experiments than the number required by conventional experiments. They

argue that this advantage has several use cases, including predicting properties of physical systems. Also in 2022, Daley et al. argued that quantum advantage exists by using special-purpose analogue quantum simulators for specialised applications [155].

3.7.0.2 Quantum Speedup in Supervised Learning

Therefore, a variety of quantum speedups have been claimed for traditional quantum algorithms, traditional quantum machine learning, and physical quantum experiments. In this section, we are interested in quantum speedups in supervised learning, dealing with the training of models with access to data that has been correctly labelled. Due to the lack of theoretical guarantees in variational quantum algorithms, the basis of the standard framework of QNNs, claims of quantum advantage in supervised learning come from quantum kernels, which can be more easily theoretically analysed.

In 2019, Schuld and Killoran explored the theoretical link between QNNs and kernel methods [26], which are another class of machine learning algorithms, and proposed strategies inspired by kernel methods to find patterns in data more efficiently with quantum computers: one strategy is to estimate "intractable

quantum kernels on the quantum device and feed them into a classical kernel method" [26] – this means that a quantum kernel could be constructed on a quantum computer that is inefficient to construct on a classical computer and then the resulting quantum kernel could be used in classical methods, such as Support Vector Machine, to perform the classification task.

Such a strategy was used for the first robust and rigorous analysis of a quantum speedup for supervised learning by Liu et al. [27]. The learning task for this quantum speedup comes from the *discrete logarithm problem*, which can be solved by Shor's algorithm [1] in polynomial time, but not for a classical algorithm. Liu et al. [27] define the *discrete logarithm problem* (DLP) as:

> DLP: given a prime p, a primitive element g of $\mathbb{Z}_p^* = \{1, 2, \ldots, p-1\}$ and $y \in \mathbb{Z}_p^*$, find $x \in \mathbb{Z}_p^*$ such that $g^x \equiv y \pmod{p}$.

They state "For a fixed p, g, we let DLP(p, g) be the discrete log problem with inputs $y \in \mathbb{Z}_p^*$. The input to the DLP problem can be described by $n = \lceil \log_2(p) \rceil$ bits". They also state that "For a large prime number p and a generator g of $\mathbb{Z}_p^* = \{1, 2, \ldots, p-1\}$ it is a widely believed conjecture that no classical algorithm can compute $\log_g x$ on input $x \in \mathbb{Z}_p^*$, in time polynomial in

$n = \lceil \log_2(p) \rceil$, the number of bits needed to represent p" [27]. Therefore the discrete logarithm problem is a well-known problem where quantum computers have an exponential speedup. In order to apply this problem to supervised learning, Liu et al. construct a class, which is a collection of functions that map from a set of inputs (features) to outputs (labels), where for any $s \in \mathbb{Z}_p^*$, they define a concept $f_z : \mathbb{Z}_p^* \to \{-1, 1\}$ over the data space $\mathcal{X} = \mathbb{Z}_p^* \subseteq \{0,1\}^n$:

$$\text{Discrete logarithm concept class: } f_s(x) = \begin{cases} +1, \text{ if } \log_p x \in \left[s, s + \frac{p-3}{2}\right], \\ -1, \text{ else.} \end{cases}$$

Liu et al. show that given the classical hardness of solving the discrete logarithm problem, it is also classically hard to learn the concept class. To learn this concept class efficiently with a quantum kernel, they construct a quantum feature map as shown in Equation 3.5

$$x \mapsto \Phi(x) = |\phi(x)\rangle\langle\phi(x)|, \text{ where } |\phi(x)\rangle = \frac{1}{\sqrt{2^k}} \sum_{i=0}^{2^k-1} \left|xg^i\right\rangle, \tag{3.5}$$

"which maps a classical data point $x \in \mathbb{Z}_p^*$ to an n-qubit quantum state, and $k = n - t \log n$ for some constant t" [27]. To efficiently compute this feature map, one must prepare the state $|\phi(x)\rangle = U(x)\,|0^n\rangle$, which can be done by

running a circuit $U(x)$ that uses Shor's algorithm as a subroutine; this circuit can be efficiently run using a fault-tolerant quantum computer, which was shown in Reference [156].

This quantum feature map can be used to obtain the kernel function as shown in Equation 3.6 by running the quantum circuit $U^{\dagger}(x_i)U(x_j)$ on input $|0^n\rangle$, and measuring the probability of the 0^n output.

$$\begin{aligned} K\left(x_i, x_j\right) &= \operatorname{Tr}\left[\Phi\left(x_i\right)\Phi\left(x_j\right)\right] \\ &= \operatorname{Tr}\left[|\phi(x_i)\rangle\langle\phi(x_i)| \cdot |\phi(x_j)\rangle\langle\phi(x_j)|\right] \\ &= \left|\langle\phi\left(x_i\right) \mid \phi\left(x_j\right)\rangle\right|^2 \\ &= \left|\left\langle 0^n \left| U^{\dagger}\left(x_i\right) U\left(x_j\right)\right| 0^n\right\rangle\right|^2 . \end{aligned} \tag{3.6}$$

This quantum kernel can then be used in the support vector machine algorithm (SVM) to find an optimal hyperplane that separates the +1, -1 classes in the feature space with the largest margin. Liu et al. show that the +1, -1 classes are separated by a hyperplane with a large margin by using this quantum feature map, which is what enables its success with SVM.

This procedure can be generalised to find a quantum speedup for supervised learning problems:

1. Construct a quantum kernel $K(x_i, x_j) = \mathrm{Tr}\left[\Phi\left(x_i\right)\Phi\left(x_j\right)\right]$ that separates inputs x from different classes by finding a quantum circuit $U(x)$ that can prepare the state $|\phi(x)\rangle$ to obtain $\Phi(x) = |\phi(x)\rangle\langle\phi(x)|$.
2. Show that it is classically hard to construct a classical kernel that separates the inputs x from different classes.

This procedure shows that a quantum speedup is possible for quantum machine-learning models if one has knowledge about the problem and can then encode that knowledge into a quantum circuit, whilst encoding the same knowledge into a classical model would be inefficient [37]. Similar strategies have resulted in other quantum speedups in supervised learning [28].

3.7.0.3 Challenges for Demonstrating Quantum Advantage in Machine Learning

As Schuld and Petruccione pointed out in *Supervised Learning with Quantum Computers* [157], provable quantum speedups are desired in quantum machine

learning because of its roots in quantum computing. Provable speedups had to be made in the early days of quantum computing, as the hardware wasn't even built so there was no way to test the speedup of the algorithm empirically. As Schuld and Killoran pointed out [31], on the other hand, classical machine learning is largely based on empirical speedups and benchmarking as it is difficult to make proofs in the classical neural network case. Therefore, whilst the most ideal case is a proof for a quantum speedup, in reality, it could be the case that an empirical speedup is sufficient for using quantum computers for machine learning tasks.

Even though quantum hardware exists today, however, it is still difficult to make claims of empirical speedups as the quantum hardware today is not large and error-free enough. Therefore, one of the challenges for provable quantum advantage in machine learning is that the theory is still lacking for classical deep neural networks, thereby making it hard to compare to theoretically. Instead, classical deep learning thrives on empirical benchmarking to demonstrate the success of the algorithm, but this poses another challenge: it is only possible to run a quantum algorithm with a small number of qubits given the computational resources of a classical simulator or the noisy small-scale hardware we have today.

Therefore it is not really possible to show a quantum advantage empirically against a comparable classical machine learning algorithm, which can work on a large number of inputs and parameters. An additional challenge is that we are comparing quantum machine learning algorithms to a moving target; classical deep learning is getting better day-by-day as more data and more parameters are being used for their models.

The challenges can be summarised below:

1. **Mismatched scales**: quantum algorithms are limited by the number of qubits, both for simulation and hardware; whereas classical algorithms work with a large input dimension.
2. **Mismatched benchmarking**: quantum algorithms are deemed successful by their computational complexity; whereas classical machine learning algorithms are deemed successful by their empirical performance.

The provable cases of quantum advantage have therefore had to focus on specific areas of classical machine learning, where there is a stronger theoretical basis, including kernel methods, in order to prove a quantum speedup. The

empirical cases of quantum advantage that have been claimed had to reduce the dimensions of the inputs so that it can be run on a classical simulator or quantum hardware. Therefore, it is only possible at the moment to show quantum advantage for very specific cases with kernel methods or on a small-scale for QNNs. For the latter case, it is also difficult to generalise to a larger-scale, as both quantum and classical behaviour could change on a different scale regime.

Schuld and Killoran in 2022 highlighted these challenges of applying quantum computing to machine learning [31] and have argued that focusing on the goal of quantum advantage limits research ultimately, as there are few problems that can be studied in the context of machine learning, given the discrepancies between classical and quantum machine learning. They believe that "three ingredients - the right quantum models to study, theoretical tools with which we can study them, and software solutions to scale experiments - are important for a meaningful attempt to explore the benefit of quantum computing for machine learning in the future" [31] and that more research should aim towards enriching these three ingredients. Therefore, they suggest alternative research questions, including searching for a building block of quantum models, quantum kernels,

and quantum gradients and software. The hope is that the development of these three areas, rather than specific cases of quantum advantage, can ultimately lead to future quantum applications.

3.8 State of the Field

To summarise, the field has been trying to understand the characteristics of QNNs and how quantum advantage can be obtained in quantum machine learning. Some research has made use of quantum kernels to assess quantum advantage as the optimisation of QNNs is non-convex and suffers from barren plateaus. Research in the field has enabled more certainty that the power and potential for quantum advantage is in how data is embedded into quantum models [125, 28, 29, 27]. So far, the research has showed some hope for quantum advantage using certain quantum feature maps [28, 29, 27], but whether we can obtain a quantum advantage on practical classical datasets is still unclear.

Our research fits into this current context of quantum machine learning. One of our contributions is providing the analysis and a method for investigating the inductive bias of QNNs in the context of quantum machine learning. In

addition, we bring in classical concepts in classical deep learning, including simplicity bias, to shed more light on the performance of QNNs on classical data and how it compares to classical neural networks. Our research connects multiple threads circulating in the field that we have reviewed in this chapter, including those of quantum feature maps, model capacity, inductive bias, QNNs, quantum kernels, and quantum advantage. In particular, our research provides an important analysis of the inductive bias of QNNs and quantum kernels, including how different data embeddings and the expressivity of QNNs affect the inductive bias and generalisation performance.

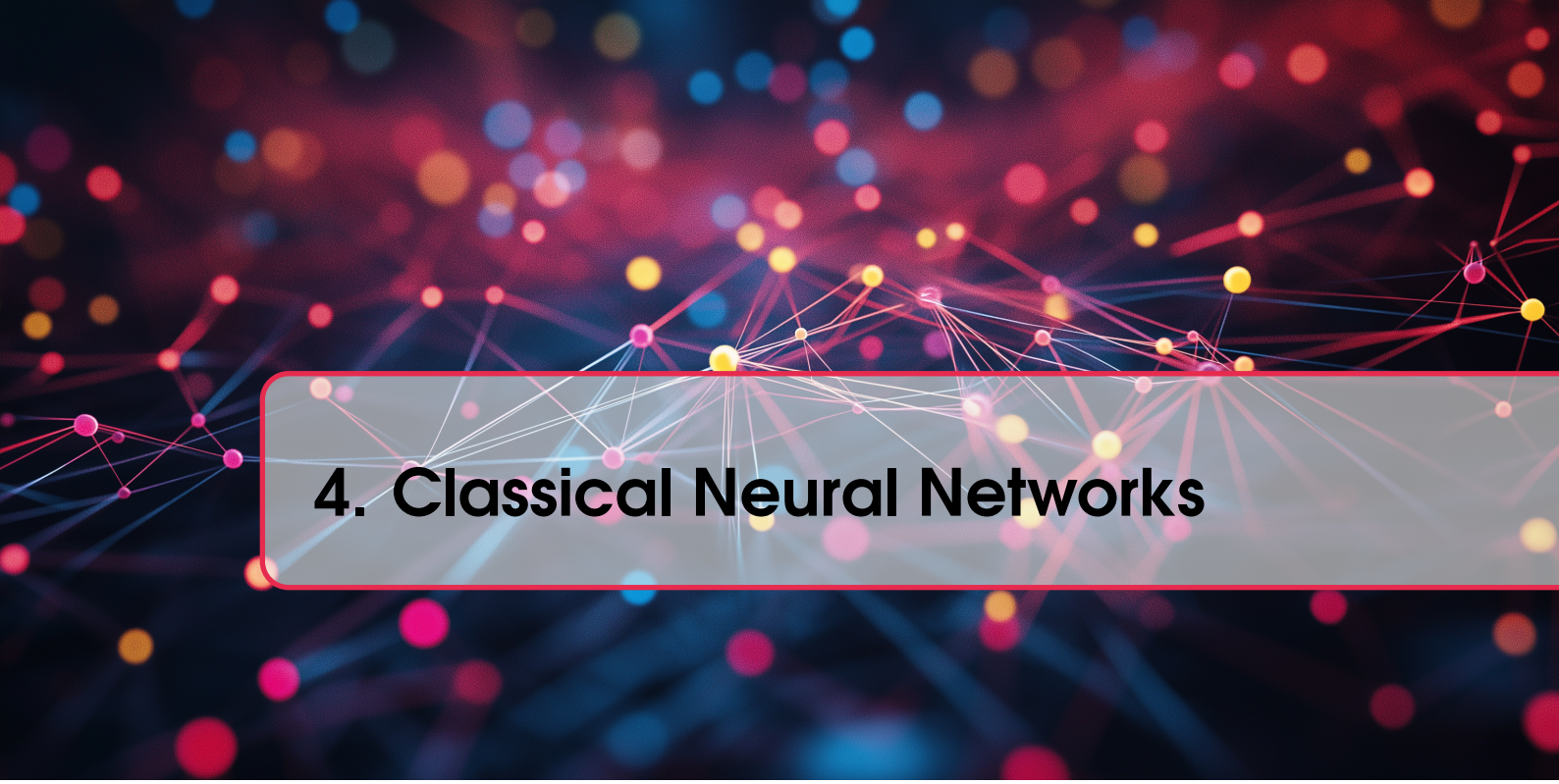
In addition, we introduce a new angle and a new thread into the field as a way to think about the performance of quantum models, including the thread of simplicity bias. By explicitly asking questions about simplicity bias, we show how the performance of quantum models differs from classical models with regards to classical datasets.

3.9 Review

In this chapter, I outlined the beginnings of QNNs and how it evolved into its

standard framework, based on variational quantum algorithms, and the shift in research goals that occurred as a result. I described how QNNs work, including particular structures that will be used in the original research in Chapter 6. This chapter provides detailed information of the technical terms and concepts used in Chapter 6. I then discussed quantum kernels and how they compare to QNNs. I also presented the technicalities of how they work. Moreover, I reviewed properties of QNNs based on the literature, which included an understanding of their limitations and potential advantages. I then focused on the concept of quantum advantage, including areas where it can be found and techniques to find it. Finally, I discussed challenges for demonstrating quantum advantage in machine learning and then summarised the current state of this subfield and how our research fits into it.

In the following chapter, Chapter 4, I turn to classical neural networks, which are the inspiration for QNNs, and I explain additional concepts, including inductive bias, that are important for a deeper understanding of the original research presented in Chapter 6.



4. Classical Neural Networks

4.1 Preview 230

4.2 Neural Networks Overview 230

4.3 History 231

4.4 Understanding Neural Networks 235

4.5 Inductive Bias 242

4.6 Algorithmic Information Theory 248

4.7 How Neural Networks Work 254

4.8 Dataset for Inductive Bias 261

4.9 Review 263

4.1 Preview

In the previous chapter, Chapter 3, I focused on quantum neural networks, one of the main topics of this thesis. Quantum neural networks are inspired by classical neural networks, which are being widely used today for a variety of applications. In this chapter, I outline the history of classical neural networks and how the understanding of them has changed over time. I also provide the essential background information for inductive bias in the context of classical neural networks, which is important for understanding the inductive bias in quantum neural networks, which I bring together in original research performed in Chapter 6. I also present the technicalities of how classical neural networks work, which can contribute to a better understanding of the information presented.

4.2 Neural Networks Overview

Neural networks are at the heart of deep learning and artificial intelligence (AI), which have had a significant impact on society and still have potential. AI is being used across a variety of industries including healthcare, finance, energy,

and education. It is now common for people to interact with AI regularly through websites, smartphones, email, and social media [158]. Most recently, ChatGPT has transformed the industry, with over 100 million users as of this writing [32]. At the core of ChatGPT and other AI applications is neural networks; therefore understanding how neural networks work and how they can be improved can have an enormous impact on the daily lives of people around the world.

4.3 History

Neural networks used in machine learning, known as *artificial neural networks*, were inspired by biological neural networks, which consist of neurons, which are cells that generate electric signals to transmit information across a network. Artificial neural networks became a mathematical model influenced by the workings of biological neurons in the brain. The concept of artificial neurons was first proposed by Warren McCullouch and Walter Pitts in 1943 [159]. In 1949, psychologist Donald Hebb published a book called *The Organization of Behaviour* [160], which introduced the concept of neuron communication, including the Hebbian learning rule summarised as "cells that fire together

wire together". Hebb's theories led to further innovation in artificial neural networks.

In 1950, Alan Turing, wrote his seminal paper *Computing Machinery and Intelligence* [161], in which he started with the line "I propose to consider the question, 'Can machines think?' ". This paper introduced the concept of a learning machine and also posited the Turing Test, a test on whether the behaviour of a machine is indistinguishable from a human. This paper laid the groundwork for artificial intelligence, a term which was first used during the Dartmouth Summer Research Project on Artificial Intelligence (DSRPAI), considered to to be the event that started AI as a field, hosted by John McCarthy and Marvin Minsky in 1956 [162].

In 1957, Frank Rosenblatt introduced the *perceptron* [163], which is a single-layer neural network capable of separating data into distinct classes by a linear boundary, such as a line (one-dimension), a plane (two-dimensions), or a hyperplane (higher dimensions); if data can be separated this way, it is known as a linearly separable task. In 1965, the first multi-layer neural network without stochastic gradient descent, which is an iterative algorithm

for optimising an objective function, was published by Alexey Grigorevich Ivakhnenko and Valentin Lapa [164]. A couple of years later in 1967, a multi-layer neural network using stochastic gradient descent and capable of solving non-linearly separable tasks was published by Shun'ichi Amari [165].

In 1969, Marvin Minsky and Seymour Papert published their famous book *Perceptrons* [166] that pointed out the limitations of perceptrons, which led to a period of stagnation in the field known as the *AI winter*.

In 1970, the backpropagation method, a method to train multi-layer neural networks, was published by Seppo Linnainmaa [167] and was first applied in a way that has become standard in 1982 by Paul Werbos [168].

The backpropagation algorithm led to a renewed interest for neural networks. In 2000s, there was a growing resurgence of enthusiasm for neural networks driven by the growing computational power and the availability of large datasets as deep neural networks (DNNs), which are neural networks with many layers between the input and output layer, became increasingly viable.

In 2010s the deep learning revolution began, sparked by key developments such as Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton creating AlexNet [169], a DNN that significantly improved the accuracy of image classification winning the ImageNet challenge, a challenge to classify images in the ImageNet dataset published in 2009 by Fei-Fei Li at Stanford [170].

In 2017, a pivotal paper titled *Attention Is All You Need* [171] was published by eight researchers who were working at Google; they proposed the *transformer*, a DNN architecture that learns context by tracking relationships in sequential data, such as words in a sentence, and generates new data from that. The transformer is based on attention mechanisms, which allows a model to focus on specific parts of the input data with different levels of importance, introduced by Bahdanau et al. in 2014 [172]. Transformers became well-suited for large language models (LLMs), which are models designed to understand and generate human language. As of today, transformers are the main architectures for large language models.

Most recently, in 2022, ChatGPT, a large language model by OpenAI, was released for public use and quickly impacted society, with its wide range of

applications and startling capabilities. It became the fastest-growing consumer application in history, reaching 100 million monthly active users two months after its launch [32].

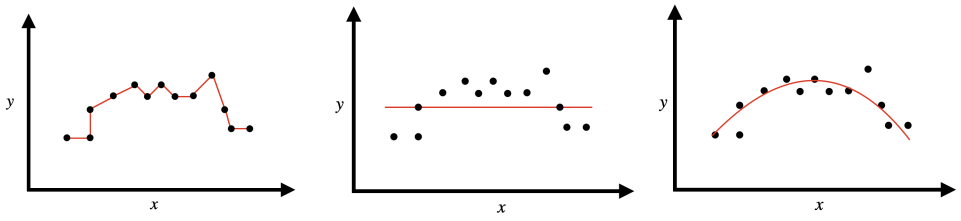
4.4 Understanding Neural Networks

In 1992, Geman, Bienenstock, and Doursat discussed the topics of bias and variance in neural networks in their paper *Neural Networks and the Bias/Variance Dilemma* [173]. The bias refers to the preferences and assumptions of a model. The bias error is the error resulting from too simplistic assumptions in a model. The variance refers to how much the model prediction varies given a dataset. The variance error is the error resulting from sensitivity to fluctuations in the given dataset. They discussed that the bias-variance dilemma is the problem of minimising these two sources of error in a model.

The bias and variance errors relate to the model's capacity, which is the complexity of the set of hypotheses a model can express. If a model has high capacity, the model can fit the training data well, but it may be over-responsive to random fluctuations in the data (i.e. it overfits the data) and therefore

may have a high variance error, which means its predictions vary greatly with small changes in the input data. The model would have a low bias error as it can make complex-enough assumptions due to its high capacity. On the other hand, if a model has low capacity, it may have a high bias error as it may make simple assumptions and may not be able to capture the underlying processes in the data (i.e. it underfits the data). The model would have a low variance error though as it is not sensitive to the fluctuations of the data due to its low capacity.

Figure 4.1 shows an example of a model (red line) that overfits, underfits, and fits just right the data points.



(a) Overfitting (high variance)

(b) Underfitting (high bias)

(c) Just right (low bias, low variance)

Figure 4.1: **Fitting data points with a model.** The black points are data points and the red line is a model. (a) A model that fits the data points exactly and has high variance. (b) A model that fits the data points poorly and has high bias. (c) A model that does not fit the data points exactly but fits the data points according to its pattern and has low bias and low variance.

Neural networks have parameters, which are variables that can be changed during training of the model in order for the model to fit the data. These parameters allow the neural network to make predictions on new data. The bias-variance tradeoff implied that neural networks needed to have enough parameters to fit the key structure of the data to avoid underfitting (which occurs when the model is too simple to fit the data), but not too many parameters to avoid fitting random fluctuations in the data, known as overfitting. This bias-variance tradeoff was used to develop ideal models, for example by including regularisation, which is a technique to reduce the model's complexity to prevent overfitting.

In practice, however, as deep learning began to mature in the 2010s, researchers observed that adding even more parameters to the point where neural networks are overparametrised (having more parameters then needed to fit the data) improves their performance, which is measured by their generalisation error, which is how accurately the model is able to predict outcome values for previously unseen data. In other words, these neural networks were able to generalise well (i.e. have a low generalisation error) despite fitting the training data exactly. According to the bias-variance tradeoff, neural networks that

were overfitting the data should have high generalisation error. This dilemma was highlighted by the seminal paper by Zhang et al. [33].

Zhang et al.'s paper [33] was a significant paper in the deep learning community that showed how the traditional views of generalisation in neural networks fail to explain why DNNs generalise well in practice. By using a dataset where they changed the true labels to random labels, they showed that DNNs are still able to train well on them, demonstrating that DNNs have the capacity to memorise training data. According to the traditional views, this would imply overfitting and therefore poor generalisation. They show, however, that there is a small gap between training and generalisation performance and therefore DNNs train and generalise well when given the correct labels. Traditional views state that regularisation methods prevent memorisation, but Zhang et al. show that this is not the case. Regularisation is a technique to prevent overfitting of a model by encouraging the model to be less complex by adding a penalty to the model's loss function. Therefore, regularisation aims to reduce variance errors in the model. Zhang et al. state that regularisation "appears to be more of a tuning parameter that often helps improve the final test error of a model, but the absence of all regularisation does not necessarily imply poor

generalisation error" [33]. Therefore, DNNs defy the bias-variance tradeoff.

There have been several papers trying to address this apparent contradiction between the bias-variance tradeoff and what is observed in practice with overparametrised neural networks. One notable paper is by Belkin et al. in 2019 [174] where they introduced the concept of the double-descent curve. This double-descent demonstrated that the historical bias-variance tradeoff was occurring for underparameterised models, so that as the complexity of the model increases, the error was decreasing as the model is able to fit the underlying structure of the data. But as the complexity of the model increases beyond a certain point, the error increases; this is depicted by a U-shaped curve as shown in Figure 4.2. The bias-variance tradeoff story ends there, but the double-descent curve demonstrates that if the complexity of the model increases even further, the error decreases as shown in Figure 4.3.

One may expect DNNs to overfit the data and therefore generalise poorly. DNNs, however, generalise well despite the fact that they are overparametrised, meaning they have many more parameters than data points. There are many more functions that can fit a training data set (i.e. give zero training error) that

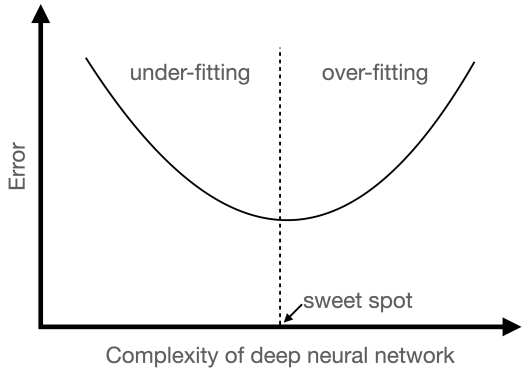


Figure 4.2: **Bias-Variance U-shaped curve.** This curve shows how the test error changes as the complexity of the deep neural network changes. Figure inspired by Figure 1 in Reference [174].

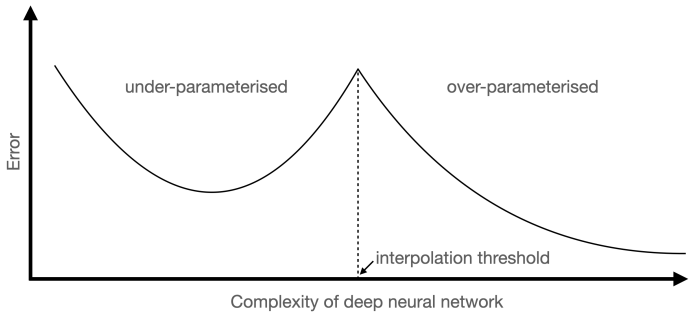


Figure 4.3: **Double descent curve.** This curve shows how the test error changes as the complexity of the deep neural network changes. This double descent curve includes the U-shaped curve of the bias-variance tradeoff and includes deep neural networks with higher complexity, which leads to the double descent. Figure inspired by Figure 1 in Reference [174].

generalise poorly compared to functions that fit the training set that generalise successfully. Given that overparametrised DNNs are highly expressive and can express the former type of functions, the question is why DNNs choose the functions that fit the training set that generalise successfully? An active area of

research studies the question of why these overparametrised models generalise well. Overparametrisation seems to act as an implicit regulariser, preventing overfitting and leading to good generalisation. One of the leading explanations is that stochastic gradient descent, used to train these overparametrised models, acts as the regulariser [175, 176, 177]. An alternative explanation is that DNNs are biased towards simple functions [178, 179]. Further research suggests that this intrinsic simplicity bias is the primary explanation, rather than some special property of stochastic gradient descent, for the good generalisation of overparametrised DNNs [180].

The paper [178] by Vallé-Perez et al. was one of the first papers to provide an explanation for this conundrum by showing that a DNN's inductive bias towards simple functions is what enables it to still generalise well in the overparametrised regime. A probability-complexity bound derived from *algorithmic information theory* demonstrates that certain maps produce low-complexity functions with an exponentially higher probability, which is called simplicity bias. Vallé-Perez et al. apply this bound to the parameter-function map of DNNs to argue that they should also exhibit a simplicity bias. They use Boolean functions and random neural networks (untrained neural networks with randomly sampled

parameters) to demonstrate practically that DNNs do have a bias towards simple Boolean functions. They argue that as real-world datasets are expected to be highly structured (therefore having a lower-complexity), that is why DNNs generalise well on these problems, because of their simplicity bias.

4.5 Inductive Bias

4.5.1 Defining and Quantifying Bias

The concept of simplicity bias is based on the inductive bias of a model. The inductive bias is the set of assumptions that a learning algorithm makes to predict outputs for unseen inputs. Having some form of an inductive bias is essential to a learning algorithm, as without it, a network wouldn't be able to prefer one solution to choose over another. Learning algorithms that are highly expressive (i.e. can express many functions) need an inductive bias to generalise well.

In order to quantify the inductive bias in neural networks, one can look at the prior over boolean functions $P(f)$, which is the probability that a neural

network, with parameters randomly sampled from a distribution, will produce a particular boolean function before it has been trained on any data. The boolean function refers to how the neural network maps binary inputs to boolean outputs (i.e. 0 or 1). A neural network aims to find the boolean function f that can map the binary inputs x to boolean outputs y, i.e. $y = f(x)$. A concrete example of boolean functions will be shown in the section on the boolean system (Section 4.8.1). The samples for calculating this probability $P(f)$ are the boolean functions produced after randomly sampling parameters chosen from some parameter initialisation distribution.

To see which types of boolean functions the neural network has a bias towards, we choose a way to quantify types of functions. A common metric is the complexity of the boolean function. This will show us whether a neural network has any bias towards high complexity or low complexity functions, the latter being a simplicity bias.

4.5.2 Simplicity Bias in Deep Neural Networks

Several papers in the DNN literature have shown that DNNs have a bias towards low complexity (i.e. simple) functions [178, 181, 182, 183, 180, 184]. In particular, it was shown in Reference [178] that for Boolean systems (see Section 4.8.1) DNNs exhibit a bias in the form of Equation 4.1

$$P(f) \lesssim 2^{-a\tilde{K}(f)+b}, \tag{4.1}$$

which is an upper bound for the probability $P(f)$ of an output f (so that $f \in O$) of a map g where $g : I \to O$ is an input-output map from the set of randomly sampled inputs I to the outputs O, a and b are scalar parameters which depend on the input-output map g but not on f, and $\tilde{K}(f)$ is an approximation of the (uncomputable) Kolmogorov complexity, which is a measure of complexity for the output f; this Equation is discussed further in Section 4.6.

The input-output map of DNNs, which is the parameter-function map, satisfies the map conditions for exhibiting a simplicity bias introduced in Section 4.6 and that is why we would expect the DNNs to exhibit a simplicity bias. The

$P(f)$ in Equation 4.1 in practice in DNNs is the same prior over functions mentioned in Section 4.5.1.

4.5.3 Simplicity Bias and Generalisation

Given that DNNs exhibit a simplicity bias, the next question is why that would lead to good generalisation performance. The argument is that real-world data tends to have descriptions that are simple rather than complex, so the simplicity bias in DNNs aligns with the structure of real-world data.

The preference for simple descriptions in real-world data, rather than complex descriptions [185] is known by the principle of Occam's razor. One possible explanation for this phenomenon is that real-world datasets (natural images, sounds, text, etc.) are an exponentially small subset of all possible datasets because the laws of physics favours this small subset (for example, data with symmetry), which is typically mathematically simple and easier for DNNs to learn as studied in Reference [186].

Furthermore, a dataset's actual dimension can be larger than the dataset's

effective dimension, describing the number of dimensions needed to capture the properties of the data, which can be a certain type of measure of the simplicity of the data. For example, data points that lie on a two-dimensional surface within a three-dimensional space have an effective dimension of two, though the data points may be described with an actual dimension of three. Studies have shown that the effective dimension of the MNIST dataset (images of handwritten digits used as a basic dataset in machine learning) is 15, compared to its actual dimension of $28^2 = 784$, implying that simple functions can fit the MNIST dataset [187].

As DNNs have an inductive bias towards simple functions, the argument is that they can generalise well on real-world data as the simple functions a DNN finds with higher probabilities are closer to the simple descriptions underlying real-world data compared to complex functions.

4.5.4 No Free Lunch Theorem

It should be noted that whilst the presence of a simplicity bias can improve the generalisation performance for simple data, its presence will likely worsen

the generalisation performance for more complex data. This is because of the *No Free Lunch Theorem*, which states that the average performance of all algorithms across all problems is the same, implying that there is no optimum algorithm for all problems.

In our context, the problems are all possible target functions (the functions that map the input data to the true output labels and is the function that the model is trying to predict) and data distributions. As a random algorithm will perform the same as any other algorithm under this theorem, this means that no algorithm will perform better than random chance across all possible target functions and data distributions. If we study data with an input dimension of seven, which is the $n = 7$ system, this theorem tells us that when averaged over all $N = 2^{2^7}$ possible Boolean target functions (see the *Boolean System* section, Section 4.8.1) and data distributions, on average the generalisation error will be 50%.

This tells us that since the DNNs perform better than random chance on low-complexity functions, they must do worse on target functions with higher complexities in order to compensate. The experiments of Reference [181] show

that the DNN performance on the parity function (a higher complexity function) is worse than random chance, which is an example of this compensation. Therefore, simplicity bias can improve generalisation for structured data, but will likely have the opposite effect for more complex data.

4.6 Algorithmic Information Theory

4.6.1 Simplicity Bias

This concept of simplicity bias is not unique to DNNs, but has been shown to occur in a range of maps in nature, and has its roots in *algorithmic information theory*, a field primarily interested in the complexity measures for data. Dingle et al. in Reference [35] showed that many input-output maps are biased towards simplicity. They demonstrate this is the case for maps ranging from the RNA sequence to option pricing in financial mathematics to the parameter-function map of DNNs [178].

In order to show this, they use results from Algorithmic Information Theory (AIT) founded by Solomonoff [188, 189], Kolmogorov [190] and Chaitin

[191, 192]. Solomonoff studied the probability $P_U(x)$ (called the *algorithmic probability*) that random input programs fed into a Universal Turing Machine (UTM) U, which is a Turing machine that can simulate any other Turing machine, generate output x. We consider *prefix UTMs*, which are UTMs that only accept prefix codes, which means that no program is a prefix of another. This implies that there is no need for a spacer or other symbol to mark the beginning and end of concatenated programs. Hence any string of bits can be unambiguously decoded into separate programs.

This helps technically to derive the following: for a prefix UTM, the probability that a random binary input program of length l is generated is 2^{-l}. This is because these input programs are binary, so a program of length one equivalent to '1' has probability $\frac{1}{2}$ and the same is for a program equivalent to '0'. When we increase the length of the program to l, each program has a probability of $\left(\frac{1}{2}\right)^l$. We can sum over the probability of all possible programs that generate x on a prefix UTM to obtain the algorithmic probability $P_U(x)$ of generating x

as shown in Equation 4.2

$$P_U(x) = \sum_{q:U(q)=x} 2^{-l(q)} = 2^{-K_U(x)} + \ldots, \tag{4.2}$$

which is the probability that a prefix UTM U outputs x when given a random input program q and where $l(q)$ is the length of the program q [35]. This means that the most likely input program that generates x is the shortest program; the length of this shortest program is defined as the Kolmogorov complexity $K_U(x)$ of the string x. This leads to the lower bound $2^{-K_U(x)} \leq P_U(x)$ since there can be programs with a length longer than the Kolmogorov complexity that produce x.

Although there may be more programs with a longer length that produce a given output x, the total probability of all these longer programs is still smaller than the total probability for shorter programs, because the probability assigned to individual programs diminishes exponentially ($2^{-l(q)}$) as the length of the program increases. Therefore, the contribution of probabilities from longer programs becomes negligible compared to the contribution of probabilities from shorter programs.

Later, Levin [193] also proved an upper bound in what is now called the AIT coding theorem as shown in Equation 4.3

$$2^{-K(x)} \leq P(x) \leq 2^{-K(x)+O(1)}. \tag{4.3}$$

This result means that simple outputs, with smaller $K(x)$, have an exponentially higher probability of being generated by random input programmes for a UTM than complex outputs with larger $K(x)$. Therefore, not all outputs are equally likely.

This result is hard to apply in practice for several reasons. One reason is because of the halting problem, which is the problem of deciding whether a program will terminate or run indefinitely in an infinite loop when given a particular input. This problem is undecidable, which means that no algorithm exists to solve the problem for any arbitrary computer program and any input to that program. Due to the halting problem, $K(x)$ is formally uncomputable.

Therefore, Dingle et al. in Reference [35] derive a weaker version of the AIT coding theorem, which approximately preserves the exponential preference for

low complexity. This modified version of the coding theorem allows one to make practical predictions for a broad class of maps. They derive an upper bound for the probability $P(x)$ that an output $x \in O$ of an input-output map $g : I \to O$ obtains upon random sampling of inputs I as shown in Equation 4.4

$$P(x) \leq 2^{-a\tilde{K}(x)+b}, \tag{4.4}$$

where $\tilde{K}(x)$ is an approximation to the uncomputable Kolmogorov complexity of x, and a and b are scalar parameters which depend on the map g but not on x. (Note that Equation 4.1 is the same as Equation 4.4 and I include them both for better understanding and convenience).

A computable input-output map $g : I \to O$, mapping N_I inputs from the set I to N_O outputs x from the set O may exhibit simplicity bias if the following restrictions are satisfied (quoted from Appendix E in Reference [178]):

1. **Map simplicity:** The map should have limited complexity, that is its Kolmogorov complexity $K(g)$ should asymptotically satisfy $K(g) + K(n) \ll K(x) + O(1)$, for typical $x \in O$ where n is a measure of the size of the input set (e.g. for binary input sequences, $N_I = 2n$).
2. **Redundancy:** There should be many more inputs than outputs ($N_I \gg N_O$) so that the probability $P(x)$ that the map generates output x upon random selection of inputs $\in I$ can in principle vary significantly.
3. **Finite size:** $N_O \gg 1$ to avoid potential finite size effects.
4. **Nonlinearity:** The map g must be a nonlinear function since linear functions do not exhibit bias.
5. **Well behaved:** The map should not primarily produce pseudorandom outputs (such as the digits of π), because complexity approximators needed for practical applications will mistakenly label these as highly complex.

While these arguments do not prove that maps are biased, they do say that if there is bias, it will follow Equation 4.4.

4.6.2 Complexity of a Function

One concept of measuring the complexity of a function is the Kolmogorov complexity. The Kolmogorov complexity $K(x)$ of a string x is the length of a shortest computer program (in a predetermined programming language) that produces x. The Kolmogorov complexity is uncomputable due to the halting problem. Therefore, we use an approximation of the Kolmogorov complexity

$\tilde{K}(x)$. One such approximation is the Lempel-Ziv (LZ) complexity. The LZ complexity counts the number of distinct substrings that appear in the string. In effect, this measures how well the string can be compressed by identifying and encoding repeated patterns or substrings.

4.7 How Neural Networks Work

4.7.1 Neurons

Neurons are the fundamental building blocks of neural networks. Each neuron takes multiple input values. For the first layer in the neural network, the input values are represented by $\vec{x}$. Then the neuron performs a weighted sum of these inputs. If $\vec{w}$ is our weight vector, then $\vec{w} \cdot \vec{x}$ is our weighted sum. A scalar bias term b is added to this weighted sum, so that an intermediate step in the output of the neuron, which we call the *integration*, is $\vec{w} \cdot \vec{x} + b$. The next step is then to apply an activation function $a(x)$ to the integration, so that the output of the neuron is $a(\vec{w} \cdot \vec{x} + b)$. This activation introduces non-linearity into the network. Common activation functions include the sigmoid, ReLU (Rectified Linear Unit), and tanh functions.

Figure 4.4 shows the diagram of a *perceptron*, which is a single layer neural network. We can see that the inputs $\vec{x}$ and the weights $\vec{w}$ are going to be summed up in the neuron through the linear function z, which is the weighted sum $\sum \vec{w} \cdot \vec{x}$. Then the neuron will apply the activation function, for example the sigmoid function $\frac{1}{1+e^{-z}}$ to obtain the output y.

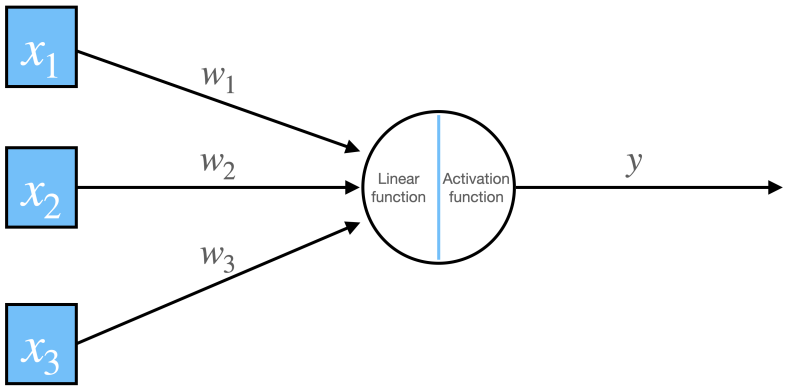


Figure 4.4: **Perceptron: a single-layer neural network.** The inputs x_i combined with the weights w_i are fed into a neuron, which calculates a linear function $z = \sum \vec{w} \cdot \vec{x}$ and then an activation function. The output of the neuron is y which is the result of the activation function calculation. Figure inspired by Reference [194].

One of the limitations of the perceptron is that it is not fully expressive; in particular, it can not express the XOR function, which is the parity function for two inputs. The parity function's value is one if and only if the input vector has an odd number of ones. This limitation was pointed out by Minsky in his book *Perceptrons* [166] and contributed to the *AI winter*, where enthusiasm for

neural networks diminished.

4.7.2 Multi-Layer Neural Networks

Neural networks are organised into layers, with each layer consisting of multiple neurons. The outputs of neurons in one layer are used as inputs for neurons in subsequent layers. There are three main layers in neural networks:

1. **Input layer:** the input layer receives the input to the neural network. The number of neurons in the input layer is calculated from the dimensionality of the input data.
2. **Hidden Layers:** these are intermediary layers between the input and output layers. The number of hidden layers and neurons in each layer is determined by the architecture of the neural network and the complexity of the task.
3. **Output Layers:** the output layer produces the final results or predictions. The number of neurons in the output layer depends on the nature of the problem (e.g. binary classification, multi-class classification, regression).

Figure 4.5 shows a diagram of a neural network.

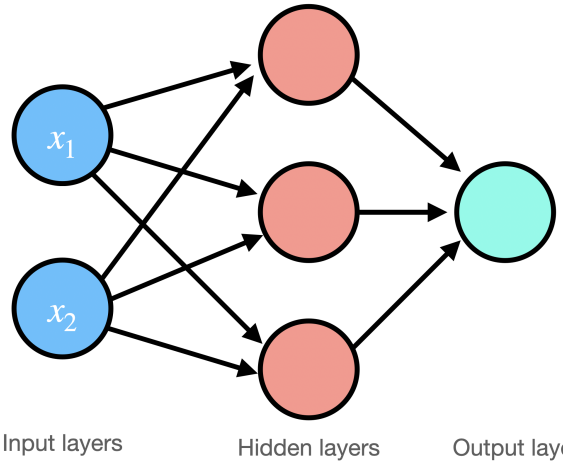


Figure 4.5: **Neural network: a multi-layer perceptron.** The inputs x_i are fed into multiple neurons which calculate the linear and activation functions. The output of these calculations are then fed as inputs into the neurons in the next layer. The last layer of the neural network is the output layer.

4.7.3 Training a Neural Network

Training a neural network involves changing the weights to minimise the difference between the neural network's predictions and the actual target values.

Training a neural network consists of the following steps:

1. **Feedforward process:** Data is passed from the input layer to the output layer, to obtain the neural network prediction on the input layer.
2. **Loss calculation:** The difference between the network's predictions $\tilde{y}$

and the actual target values y is measured using a loss or cost function. Common loss functions include mean squared error MSE $= \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2$ for regression problems and cross-entropy for classification problems: Loss $= -\sum_{i=1}^{n} y_i \cdot \log \tilde{y}_i$.

3. **Backpropagation:** The gradient of the loss with respect to the network's weights is computed using the chain rule of calculus. $\frac{\partial C}{\partial w_i} = \frac{\partial C}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial z} \times \frac{\partial z}{\partial w_i}$, where C is the cost function and z is the integration $z = \vec{w} \cdot \vec{x} + b$.
4. **Optimisation:** This gradient information is used to update the weights to reduce the loss. One such optimisation technique is gradient descent $w' = w - \eta \nabla C(w)$, where w' are the updated weights based on the gradient of the cost function with respect to the previous weights. η is the learning rate, which is a scalar determining the magnitude of the parameter updates.
5. **Iterations:** The training process is typically performed over multiple iterations, until the model's performance converges or reaches a satisfactory level.

4.7.4 Generalisation Error

Once a neural network is trained and evaluated, it can be used for inference: making predictions on unseen data. The new data is fed into the trained network as in the feedforward process and a prediction is obtained at the output layer. To calculate the generalisation error of the neural network (i.e. how well it performs on unseen data), the feedforward process is performed on a separate dataset, called the test set, and the loss or accuracy is calculated between the actual target values and the neural network's predicted values.

4.7.5 Expressivity

Generally, adding more layers to a neural network can make it more expressive, meaning that it can express more functions, which increases the ability of the neural network to achieve its goal, which is to find the function H (e.g. a continuous or boolean function) that can map the inputs x to outputs y, i.e. $y = H(x)$. The *expressivity* of a neural network refers to how many functions H the neural network can express. If the neural network can express more functions, it has a higher chance of finding the true function.

The Universal Approximation Theorem (UAT) states that there is a feedforward neural network with a single hidden layer containing a finite number of neurons that can approximate any arbitrary continuous function H, given an appropriate activation function [123, 195, 124]. UAT provides a powerful argument for the potential of neural networks to represent a wide range of functions.

Whilst a single hidden layer can theoretically approximate any function with many neurons (making the neural network wider), a DNN with multiple layers often requires fewer parameters to approximate a function and can learn better compared to a shallow network [196, 197].

DNNs are highly expressive due to their depth and capacity, as deeper networks (i.e. those with more layers) are generally more expressive than shallow ones with the same number of parameters. The combined properties of high expressivity and a simplicity bias in DNNs enable their successful performance. These two properties are key for comparing the performance of quantum neural networks to DNNs, which we do in Chapter 6.

4.8 Dataset for Inductive Bias

4.8.1 The Boolean System

The Boolean system, a key object of study in computer science, was used as the dataset to study the simplicity bias in DNNs [34]. The boolean dataset consists of all possible permutations of zeros and ones up to a length n. In other words, the dataset covers $\{0,1\}^n$. For example, if $n = 3$, then the boolean dataset $\vec{x}$ is $x_0 = 000, x_1 = 001, x_2 = 010, x_3 = 011, x_4 = 100, x_5 = 101, x_6 = 110, x_7 = 111$. To obtain the boolean function, for each binary input x_i, we obtain a boolean output y_i, where $y_i = f(x_i) \in \{0,1\}$, where f is a boolean function that maps the input x_i to the output y_i. Therefore, the boolean function is

$$f : \{0,1\}^n \to \{0,1\}.$$

There are 2^{2^n} possible different functions f. The n-dimensional boolean dataset is $\mathcal{B}(f)_n = \{(x_1, f(x_1)) \dots (x_{2^n}, f(x_{2^n}))\}$, where x_i covers $\{0,1\}^n$ and $f(x_i) \in \{0,1\}$.

An example of a Boolean function is shown in Table 4.1.

x	$f(x)$
000	0
001	0
010	1
011	0
100	1
101	0
110	0
111	1

Table 4.1: **Example of a Boolean function.** x consists of all the possible combinations of a binary string with length three. Each input x_i is mapped to an output $f(x_i)$, which is the boolean function 00101001.

In this example $f = 00101001$. f is therefore a binary string of length 2^n and there are 2^{2^n} possible different functions.

4.8.2 Boolean System with Deep Neural Networks

In relation to DNNs, the inputs to the DNN are the binary inputs $\vec{x}$, and the prediction of the DNN is the output $y = H(\vec{x})$, where H represents the function of the DNN. When all the binary inputs are run through the DNN during the feedforward process with the same set of weights, a continuous output y is obtained. The continuous output y is then thresholded to obtain a boolean function f. If the output $y \geq 0.5$, then the boolean output is 1. If the output $y < 0.5$, then the boolean output is 0. By choosing another random set of

weights and running the feedforward process with the binary inputs again, another boolean function f is obtained. This is repeated with m samples of random sets of weights (i.e. parameters) that are chosen from some parameter initialisation distribution. Therefore, we obtain m boolean functions and can then calculate $P(f)$.

The complexity of the boolean function $K(f)$ can be computed using an approximation to the Kolmogorov complexity, such as the Lempel-Ziv complexity $\tilde{K}(f)$. With $P(f)$ and $\tilde{K}(f)$, one can see how $P(f)$ changes as $\tilde{K}(f)$ changes, which shows the bias of the DNN.

4.9 Review

In this chapter, I outlined the history that led to the well-known DNNs we have today. I showed how the theory of the bias-variance tradeoff and the empirical performance of DNNs were at odds with each other, which led to new hypotheses to the reasons behind the success of DNNs. One of these hypotheses included simplicity bias. I then explained inductive bias in the context of DNNs, as well as the roots of simplicity bias in algorithmic information theory.

I described how classical neural networks work and presented the boolean system, which will be used for the research in Chapter 6.

In the following chapters, I present my new contributions to the field, from the contributions of *frameworks* in Chapter 5 to research contributions in Chapters 6, 7, and Appendix A.

In Chapter 6, we apply the information in this chapter, Chapter 4, to the context of quantum neural networks. In particular, we investigate the presence of simplicity bias in quantum neural networks by studying their interaction with the boolean system.

5. Categories of Quantum Advantage and Algorithms

5.1 Preview 265

5.2 Quantum Algorithms 266

5.3 Quantum Advantage 271

5.4 Review 292

5.1 Preview

In Chapter 2, I introduced important concepts in the field, including quantum algorithms and quantum advantage. In Chapters 3 and 4, I provided background information on quantum neural networks and classical neural networks respectively. In this chapter, I present several frameworks I have created to

help clarify and organise various concepts and terms in the field that have been introduced in the previous chapters. These concepts and terms are central for understanding the work in this thesis.

5.2 Quantum Algorithms

In this thesis, I use several terms for different types or classes of quantum algorithms. In Figure 5.1, I have created a Venn diagram that displays how these terms fit together.

The broadest term, corresponding to the largest circle, is *quantum algorithms*. A *quantum algorithm* is a sequence of instructions to solve a particular problem with a quantum computing model. The standard model of quantum computing is the *quantum circuit* model, where a *quantum circuit* is a sequence of quantum gates and measurements. These quantum gates, which are equivalent to matrices mathematically, are the operations that change the state of the qubits (equivalent to vectors mathematically) so that a calculation is performed; the measurements of the qubits, along with some processing, give the solution to the problem.

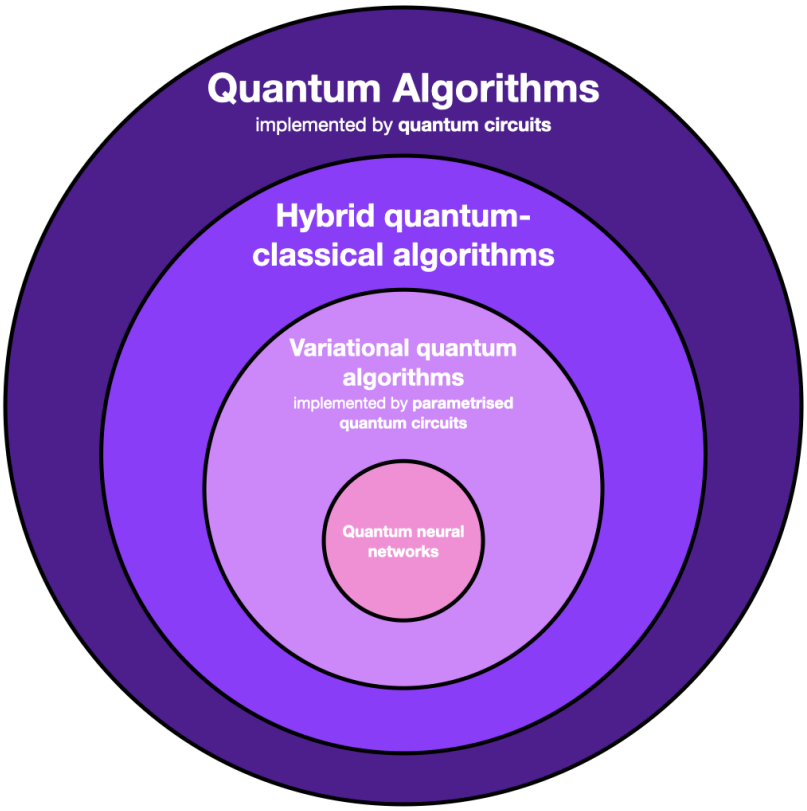


Figure 5.1: **Quantum algorithms framework.** This Venn diagram distinguishes classes of quantum algorithms, including hybrid-quantum classical algorithms, variational quantum algorithms, and quantum neural networks.

There are many different classes of quantum algorithms; in this thesis, I focus on a class of quantum algorithms called *hybrid quantum-classical algorithms*, which are algorithms that use both quantum and classical resources together to solve a particular problem. One significant subclass of *hybrid quantum-classical algorithms* is *variational quantum algorithms*, which are implemented

with *parametrised quantum circuits*, which are quantum circuits consisting of quantum gates with parameters that can be changed (mathematically, these parameters are variables in the matrices). The gate parameters are changed and optimised by a classical optimiser so that the optimal gate parameters correspond to an optimised quantum circuit that gives the solution to the problem.

A subclass of *variational quantum algorithms* is *quantum neural networks*, which are variational quantum algorithms used for machine learning tasks. In particular, in supervised learning (which is a category of machine learning that uses correctly labeled datasets to train a machine learning model to predict labels in other datasets), the prediction is calculated from the measurements of a subset of qubits in the algorithm; these qubits are called the *readout qubits*. Other qubits may be used in the algorithm, but they are only used for the processing and their final measurements do not matter. *Quantum neural networks* in supervised learning consist of quantum circuits with at least one readout qubit, whose measurement is used for predicting a label on some dataset. In Table 5.1, I clarify the differences between variational quantum algorithms and quantum neural networks, based on the way the problem is

encoded and the goal of the algorithm.

	Variational Quantum Algorithm	**Quantum Neural Network (for supervised learning)**
Problem	Encodes the problem into the cost function (i.e. the Hamiltonian).	Encodes the problem (the input data) into the quantum circuit by using an encoding method.
Goal	The goal is to optimise the parameters of the quantum circuit to obtain the minimum cost function, corresponding to the solution to the problem.	The goal is to optimise the parameters so that they fit the training data and then use the optimised quantum circuit to make predictions on the unseen data.

Table 5.1: **Variational quantum algorithms vs quantum neural networks.** The difference between the problem and goal for these two types of quantum algorithms. The encoding method is the way to convert the classical input data into a quantum state (see Chapter 3 Section 3.4.1 for further details).

In the field, the terms *hybrid quantum-classical algorithms*, *variational quantum algorithms* and *quantum neural networks* are often used interchangeably, but I have distinguished the subtle differences between them so that their meanings are clearer. I think it is also important to distinguish quantum neural networks

from variational quantum algorithms because it is possible in the future that alternative frameworks for quantum neural networks are found that do not fit into the standard variational quantum algorithm framework (and thus quantum neural networks would no longer be considered a subclass of variational quantum algorithms); the need for alternative quantum neural network frameworks is discussed in Chapter 6. Therefore, the placement of quantum neural networks in the Venn diagram could change in the future, depending on whether better alternative frameworks are found.

In this thesis, my coauthors and I have done research in order to:

1. Study the *foundation* of **quantum neural networks** by investigating their inductive bias and expressivity.
2. Develop the *optimisation* of **quantum circuits**, which are used to implement **quantum algorithms** by creating a quantum compiler with new techniques.
3. Discover an *application* of quantum computing to a real-world problem by using a **variational quantum algorithm**.

5.3 Quantum Advantage

5.3.1 Purpose of Quantum Advantage

A quantum algorithm has *quantum advantage* if it can solve a problem with some sort of advantage, such as speed, over a classical algorithm.

There are probably three main purposes for the study of quantum computing:

1. Pursuit of work itself, regardless of the known utility of the work and the increased knowledge resulting from the work.
2. Pursuit of knowledge regardless of its usefulness.
3. Pursuit of practical applications of quantum computing to impact society in a tangible way.

The pursuit of quantum advantage can fulfill all three purposes, with the third purpose being its focus. The pursuit of practical applications of quantum computing, however, is not enough. Because we already have classical computers that can solve a variety of problems, finding applications of quantum computing

is only useful for society if it can have some sort of advantage over classical computers, so that a quantum computer, instead of (or in addition to) a classical computer, is used to solve a problem.

In the scientific community and in the media, there have been various terms with various meanings for concepts related to quantum advantage. In this section, my aim is to provide a clearer understanding around discussions of quantum advantage by:

1. **Categorising types of quantum advantage**, which could lead to better communication within and between the scientific community and media.
2. **Identifying key questions**, which could help researchers and the public identify what is important.
3. **Highlighting important goals**, which could lead to new research topics.

5.3.2 Measures for Quantum Advantage

A quantum advantage occurs when there is a quantum solution to a problem that has some sort of advantage over the classical solution. The most common way to claim a quantum advantage is to show that the quantum solution is faster than the classical solution. Most of the research in quantum computing is concerned with the speed of quantum solutions. Little attention, however, is given to other types of advantage, which is one of the reasons why I want to highlight other areas of quantum advantage in this section.

In Figure 5.2, I split the measures of *advantage* into two categories: *performance* and *resources*. The *performance* is concerned with metrics for comparing the performance of a quantum algorithm versus a classical algorithm. The two main metrics of *performance* are *speed* and *accuracy*. An advantage in the area of speed is known in the literature as a *quantum speedup*. There are also different types of quantum speedups, including polynomial and exponential speedups. An advantage in the area of accuracy is considered a *quantum accuracy advantage*, which usually occurs for quantum machine learning algorithms, where one needs to make a prediction. If a quantum algorithm can make a prediction

that is more accurate compared to a classical algorithm, then this would be considered a *quantum accuracy advantage*. Another example of *quantum accuracy advantage* is where a quantum strategy can win a game with a higher probability compared to a classical strategy, such as in the odd-cycle game [198].

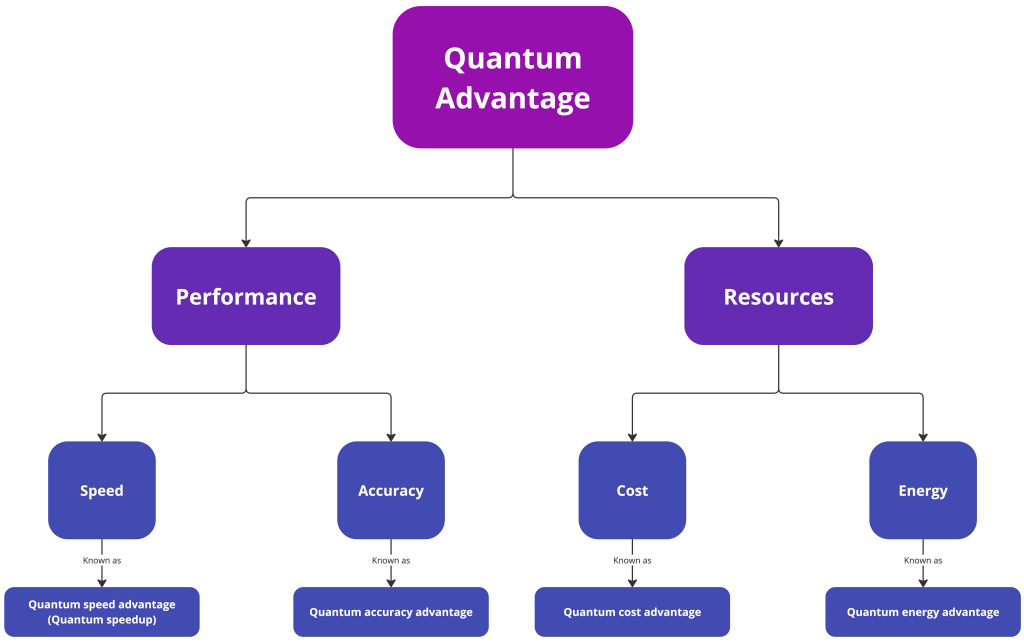


Figure 5.2: **Types of quantum advantage.** Classifying types of quantum advantage into performance and resources, in which performance includes speed and accuracy, and resources includes cost and energy.

The other category of quantum advantage is regarding the *resources*, which refers to all of the resources to run the quantum algorithm and obtain the

solution. It could be the case, for example, that a quantum algorithm is not faster than its equivalent classical algorithm for solving a problem, but the quantum algorithm consumes fewer resources, such as electricity, than the classical algorithm and therefore it may be better to run the quantum algorithm instead. I have split the *resources* into two main metrics, including *cost* and *energy*. The former is concerned with all of the costs for running the solution, including equipment and cost of labour. The latter is concerned with the consumption of energy, in particular regarding environmental concerns. It should be noted that these metrics can influence each other; for instance, using fewer energy resources could reduce costs.

The scientific community is mostly concerned with the *performance* of quantum algorithms and there is not as much analysis of the *resource* advantages of quantum algorithms. Therefore, an opportunity for finding quantum advantage could come from analysing the *resources* used for quantum algorithms in comparison to classical algorithms, even if no *performance* advantage has been found. This may be relevant for *variational quantum algorithms*, where *performance* advantages are difficult to obtain; perhaps advantages of these algorithms could be found in other ways.

5.3.3 Categories of Quantum Advantage

5.3.3.1 Flowchart Symbols

In Figure 5.4, I have created a flowchart to identify:

1. **Definitions** of different types of quantum advantage (shown by the purple boxes in the flowchart).
2. **Key Questions** to ask about quantum advantage (shown by the pink diamonds in the flowchart).
3. **Goals** to get to quantum advantage (shown by the black arrows in the flowchart).

The orange star is a definition that is equivalent to the ideal ultimate goal of a quantum algorithm. The black arrows (i.e. the goals) move towards the direction of this orange star.

5.3.3.2 Flowchart Explanation

One starts at the top of the flowchart with a *problem* to be solved, as shown by the blue box. The first question is whether 'there is a quantum algorithm to the

problem that has some sort of advantage (e.g. speed, accuracy, cost, energy) compared to classical solutions?'. If the answer is 'No' then the quantum algorithm is a *Quantum Solution* but has no quantum advantage. The goal of such algorithms is to find an advantage for the quantum solution so that one can get a *Quantum Advantage* for the solution, as shown by the black arrow in the flowchart.

If there is no quantum advantage: One then asks 'Is the problem of practical interest? (i.e. is it useful for real-world scenarios?)'. If the answer is 'No', then the algorithm is a *Nonpractical Quantum Solution (NQS)*. One can then ask 'Can the solution be implemented on a device? (i.e. is the solution available?)'. If the answer is 'No', then the solution is a *Nonpractical Unavailable Quantum Solution (NUQS)*. If the answer is 'Yes', then the solution is a *Nonpractical Available Quantum Solution (NAQS)*.

Returning to the question of 'Is the problem of practical interest? (i.e. is it useful for real-world scenarios?)'. If the answer is 'Yes', then the algorithm is a *Practical Quantum Solution (PQS)*. One can then ask the question 'Can the solution be implemented on a device? (i.e. is the solution available?)'.

If the answer is 'No', then the solution is a *Practical Unavailable Quantum Solution (PUQS)*. If the answer is 'Yes', then the solution is a *Practical Available Quantum Solution (PAQS)*. **It should be noted that many variational quantum algorithms in the literature are *Practical Available Quantum Solutions (PAQS)*.**

If there is quantum advantage: Returning to the first question of whether 'there is a quantum algorithm to the problem that has some sort of advantage (e.g. speed, accuracy, cost, energy) compared to classical solutions?'. If the answer is 'Yes' then the quantum algorithm has a *Quantum Advantage.* The next question is what type of quantum advantage is shown by the quantum algorithm: speed, accuracy, cost, or energy. This question is related to Figure 5.2. After specifying the type of quantum advantage, one then asks 'Is the problem of practical interest? (i.e. is it useful for real-world scenarios?)'. If the answer is 'No', then the algorithm has a *Nonpractical Quantum Advantage (NQA)*. The goal of such algorithms is to find a use-case of the problem that the quantum algorithm is solving, as shown by the black arrow in the flowchart.

If the problem is of practical interest: Returning to the question of whether 'the problem is of practical interest? (i.e. is it useful for real-world scenarios?)'. If the answer is 'Yes', then the quantum algorithm has a *Practical Quantum Advantage (PQA)*. Similar to above, the next question to ask is 'Can the solution be implemented on a device? (i.e. is the solution available?)'. If the answer is 'Yes', then the quantum algorithm has a *Practical Available Quantum Advantage (PAQA)*. If the answer is 'No', then the quantum algorithm has a *Practical Unavailable Quantum Advantage (PUQA)*. The goal of such algorithms is to reduce the resources (e.g. number of qubits and gates) required so that the quantum algorithm can be run on a device and therefore have *Practical Available Quantum Advantage (PAQA)*. The simplified diagram in Figure 5.3 stops here, as the simplified diagram is only interested in the key questions, which are:

1. **Practicality**: is the **problem** of practical interest?
2. **Availability**: is the **solution** available?

The questions of relevance and availability are inspired by Schuld and Petruc-

cione's markers for quantum advantage in Reference [38].

If the solution is unavailable: Returning to the question of 'Can the solution be implemented on a device? (i.e. is the solution available?)', if the answer is 'No', one can also ask additional questions for further clarification and categorisation as shown in Figure 5.4. The next question to ask is 'Is there a theoretical basis for the advantage? (i.e. is the advantage not just empirical?)'. It is not possible to answer 'No' to this question for a quantum algorithm with *Practical Unavailable Quantum Advantage (PUQA)*, because the algorithm is unavailable to be run on devices and thus is only theoretical. In order to demonstrate an empirical quantum advantage, the solution must be available to be run on a device. Therefore, a quantum algorithm with *Practical Unavailable Empirical Quantum Advantage (PUEQA)* can not exist. If the answer is 'Yes', then the quantum algorithm has a *Practical Unavailable Theoretical Quantum Advantage (PUTQA)*. Note that in this case, the term 'theoretical' in this context does not necessarily need to exclude empirical, but given that the quantum algorithm is unavailable to be run on a quantum device, 'theoretical' here means that the quantum algorithm can demonstrate only a theoretical advantage and not an

empirical advantage.

If the solution is available: Returning to the question of 'Can the solution be implemented on a device? (i.e. is the solution available?)', if the answer is 'Yes', then the next question is 'Is there a theoretical basis for the advantage (i.e. is the advantage not just empirical?)'. If the answer is 'Yes', then the quantum algorithm has a *Practical Available Theoretical Quantum Advantage (PATQA)*. Note that in this case, 'theoretical' does not exclude empirical, so such a quantum algorithm can demonstrate a theoretical and empirical advantage since the solution is available to be run on a device. Whereas, the quantum algorithm with a *Practical Available Empirical Quantum Advantage (PAEQA)* just has an empirical advantage. If the answer is 'No', then the quantum algorithm has a *Practical Available Empirical Quantum Advantage (PAEQA)*. The goal of such algorithms is to ideally find a theoretical reason for the quantum advantage so that the quantum algorithm can have a *Practical Available Theoretical Quantum Advantage (PATQA)*, as shown by the black arrow.

Is the quantum advantage scalable: The next question is 'Can the quantum advantage be scaled up on available devices to solve real-world problems at scale?'. If the answer is 'Yes' for quantum algorithms with *Practical Available Theoretical Quantum Advantage*, then the quantum algorithm has a *Practical Available Theoretical Scalable Quantum Advantage (PATSQA)*, which is the ideal ultimate goal of quantum algorithms, as shown by the orange star. If the answer is 'No', then the quantum algorithm has a *Practical Available Theoretical Nonscalable Quantum Advantage (PATNQA)*. The goal of such algorithms is to scale up the device, so that they can have a *Practical Available Theoretical Scalable Quantum Advantage (PATSQA)* as shown by the black arrow. Similarly, If the answer is 'Yes' for quantum algorithms with *Practical Available Empirical Quantum Advantage (PAEQA)*, then the quantum algorithm has a *Practical Available Empirical Scalable Quantum Advantage (PAESQA)*. If the answer is 'No', then the quantum algorithm has a *Practical Available Empirical Nonscalable Quantum Advantage (PAENQA)*.

The device required for quantum advantage: After asking the above questions about the categorisation of quantum advantage, one can then ask

what device the quantum advantage can be obtained on. There are three main types of devices I present, including a quantum computer (which is universal, meaning that it can implement any arbitrary quantum algorithm), a non-universal quantum device, such as a quantum anneler, gaussian boson sampler, or quantum analogue device (see Section 2.6 for more information on these devices), or a classical simulator. The last one is interesting as it shows that it may not be necessary to have a quantum device to show some sort of quantum advantage; a classical simulator could simulate a quantum algorithm and achieve an advantage over a classical algorithm; this is unlikely, however, as a classical simulator requires exponential resources to simulate a quantum algorithm, and thus a quantum device is more likely to obtain some sort of advantage.

After specifying the device, we return to the question of the type of advantage; we show that it is possible to add a '-X' to the end of the phrase or acronym to specify whether the advantage comes from speed, accuracy, cost, or energy. For example, if a quantum algorithm with a *Practical Available Theoretical Quantum Advantage (PATQA)* comes from speed, one could say that the algorithm has a *Practical Available Theoretical Quantum Speed Advantage (PATQA-S)*.

Nonpractical quantum advantage: Note that the *Nonpractical Quantum Advantage (NQA)* path has the same branches as the *Practical Quantum Advantage (PQA)* described above, except the branch with the question 'Can the quantum advantage be scaled up on available devices to solve real-world problems at scale?' because nonpractical problems can not solve real-world problems.

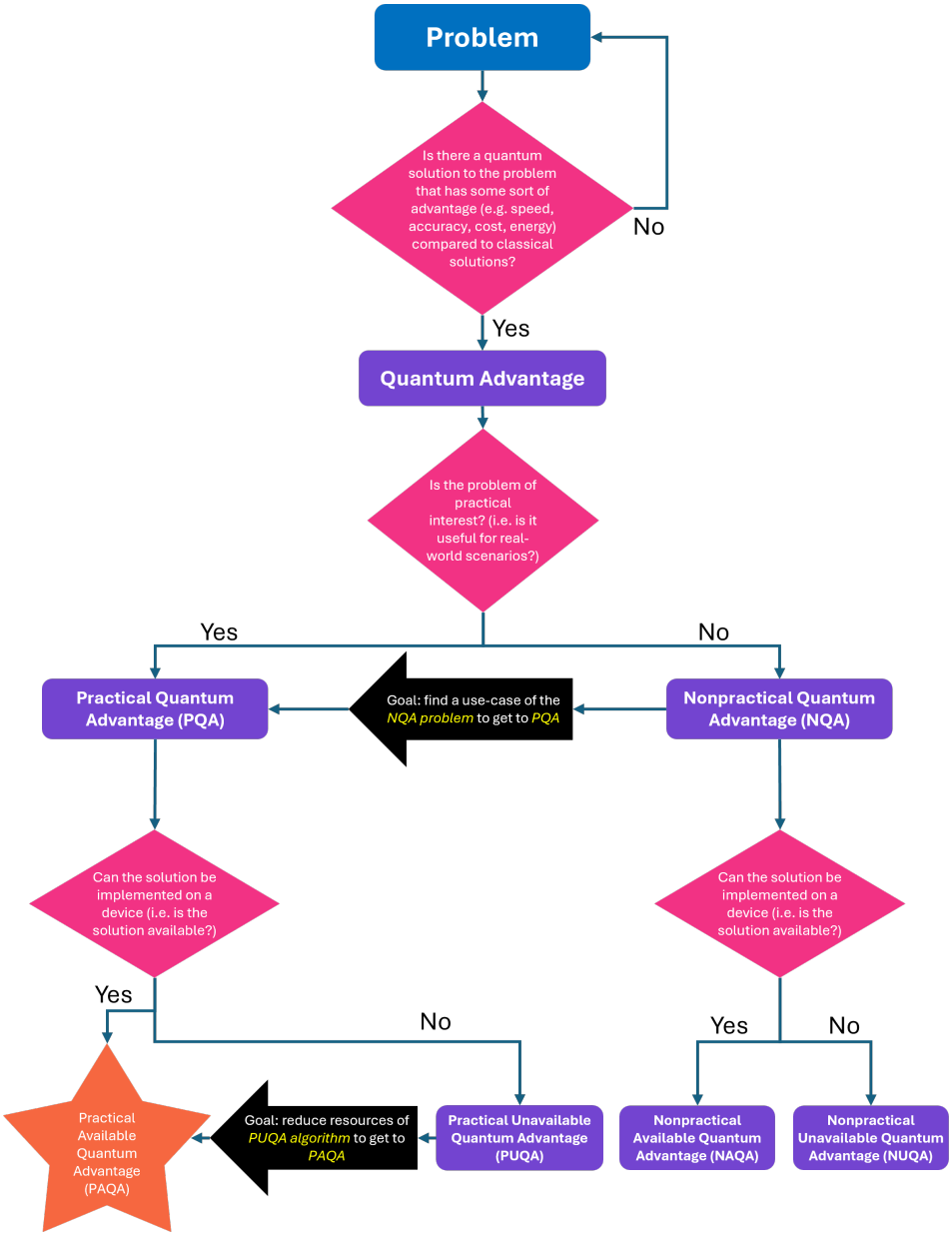


Figure 5.3: **Quantum advantage flowchart simplified.** Classifying types of quantum advantage by asking key questions and identifying the goals of certain classes of quantum advantages (a simplified version of Figure 5.4). The purple boxes are definitions, the pink diamonds are questions, and the black arrows are goals to move towards the orange star, which is a definition that embodies the ideal ultimate goal of quantum advantage.

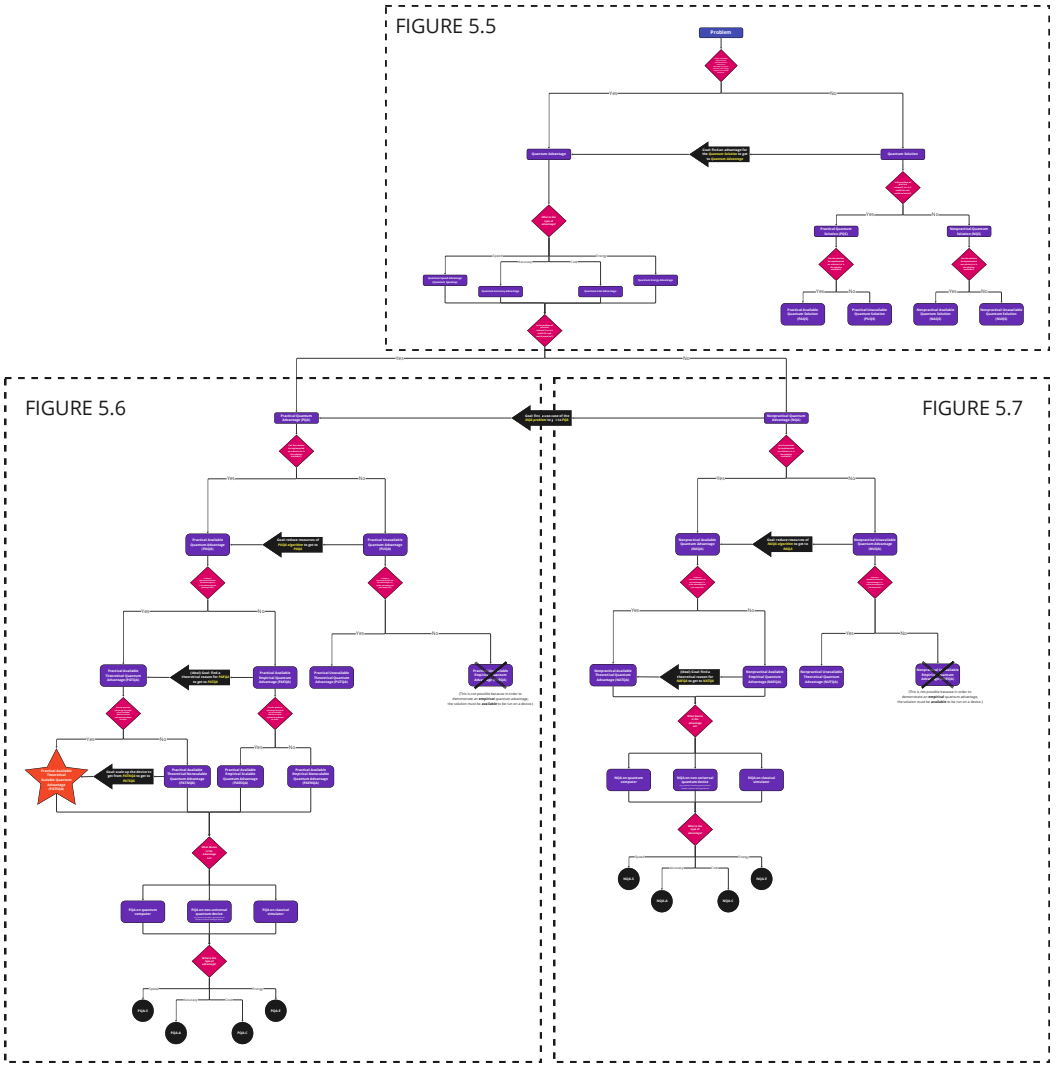


Figure 5.4: **Quantum advantage flowchart.** Classifying types of quantum advantage by asking key questions and identifying the goals of certain classes of quantum advantages. The purple boxes are definitions, the pink diamonds are questions, and the black arrows are goals to move towards the orange star, which is a definition that embodies the ideal ultimate goal of quantum advantage. Figures 5.5, 5.6, and 5.7 display larger versions of parts of this flowchart as shown in the marked boxes in this figure.

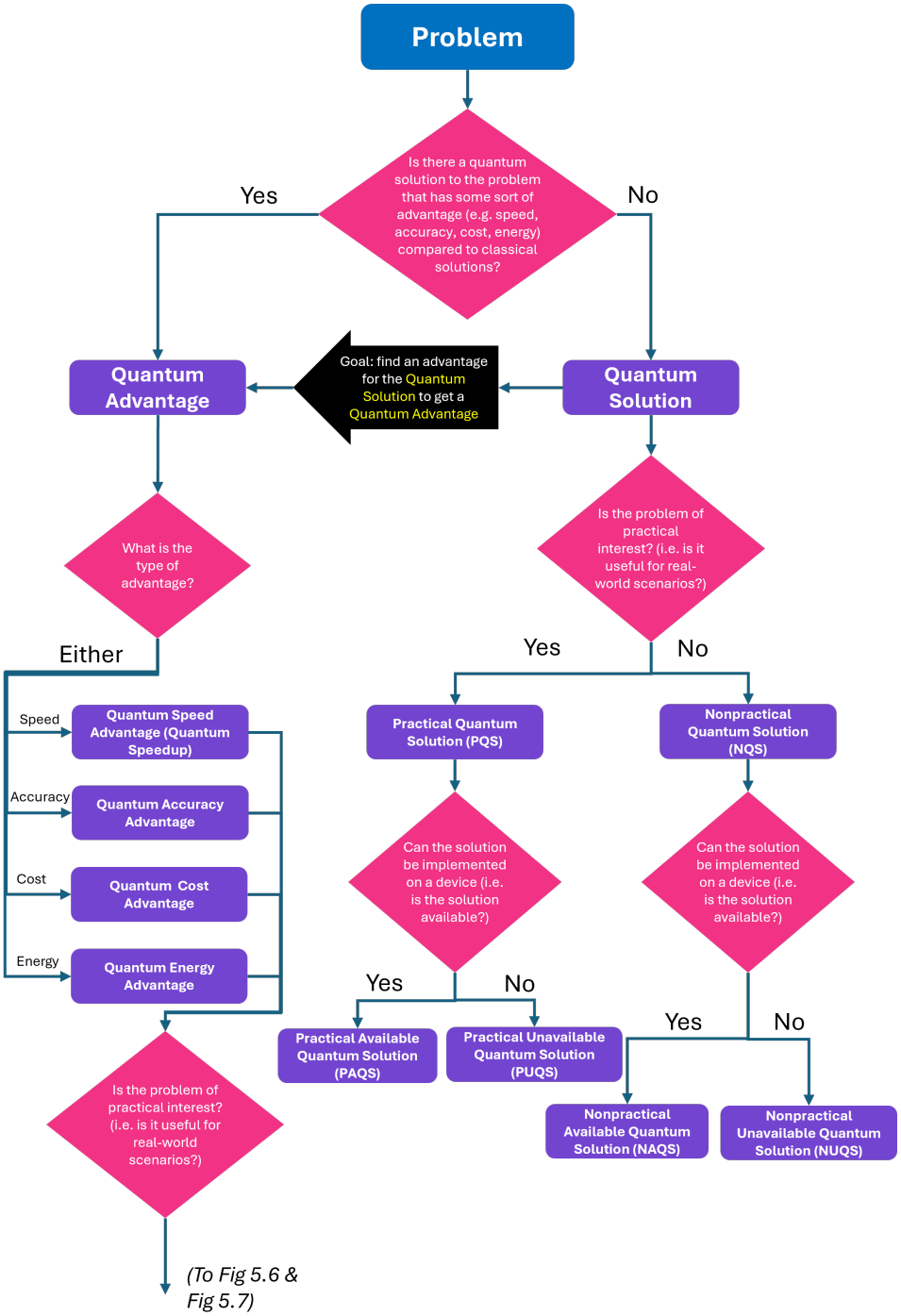


Figure 5.5: **Quantum advantage flowchart starting from the problem.**

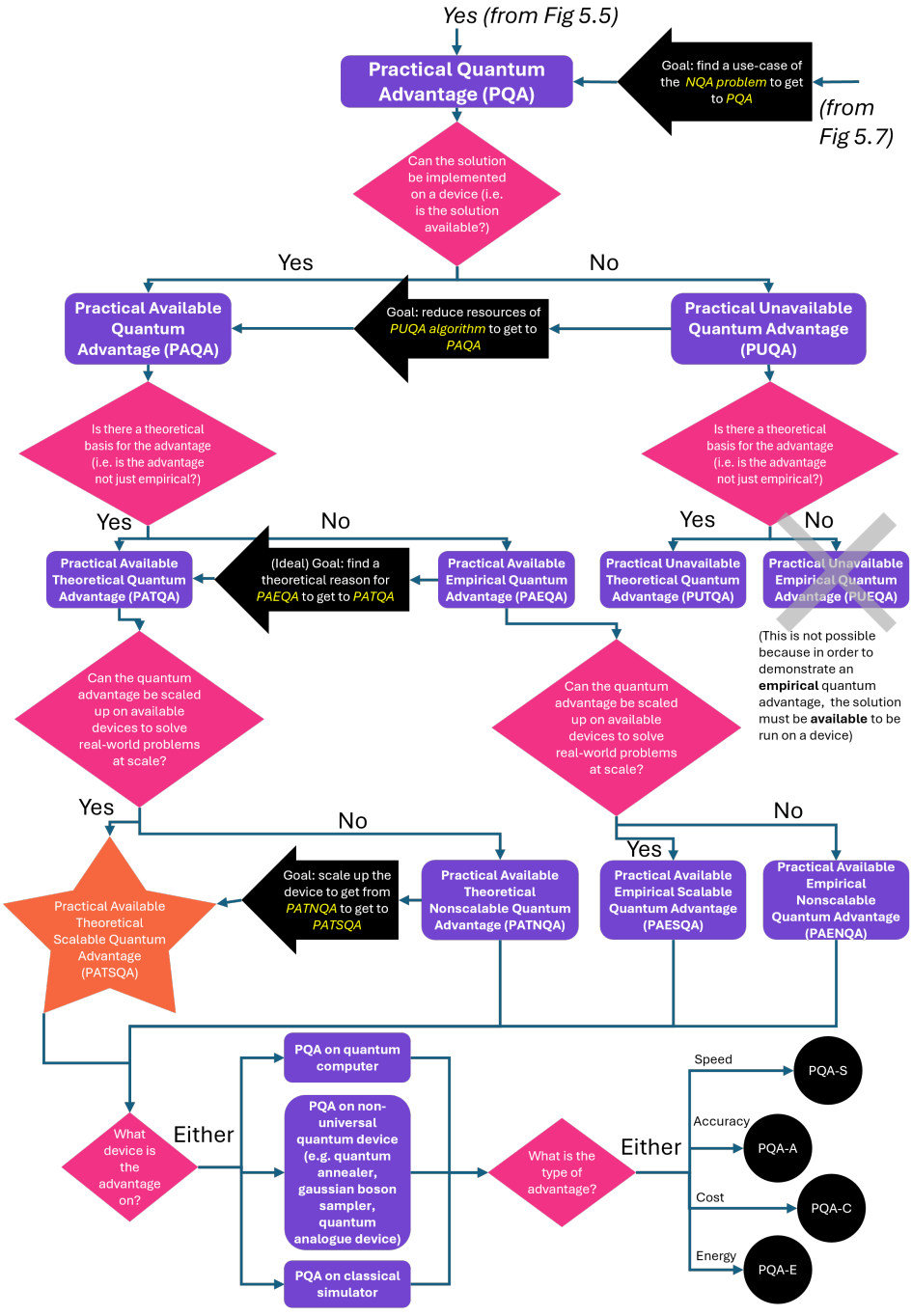


Figure 5.6: **Quantum advantage flowchart starting from Practical Quantum Advantage.**

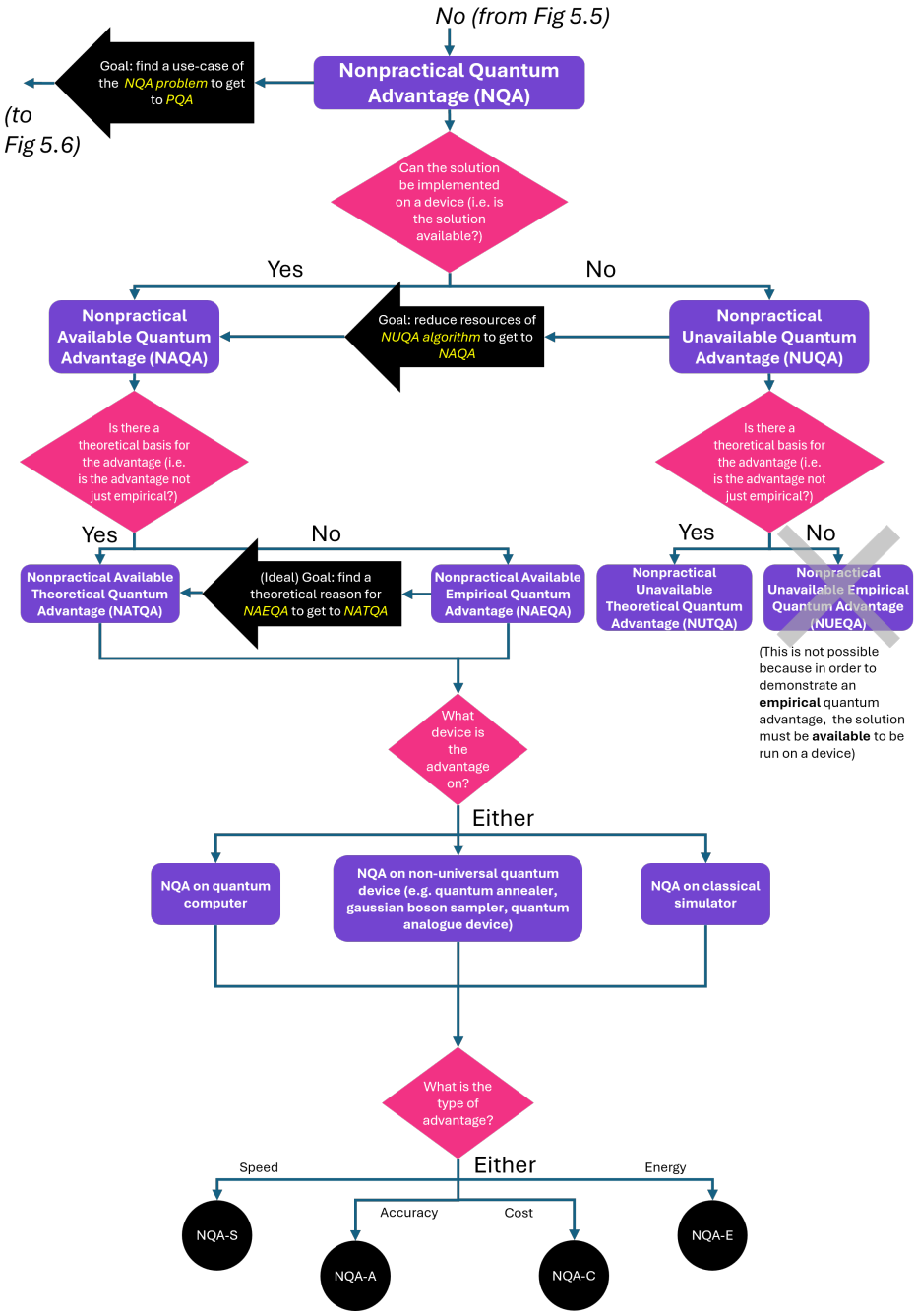


Figure 5.7: **Quantum advantage flowchart starting from Nonpractical Quantum Advantage.**

5.3.3.3 Goals

In Figures 5.4 and 5.3, the goals, shown by black arrows, can help researchers identify new research questions. For example, if there is a quantum algorithm that has a *Practical* ***Unavailable*** *Quantum Advantage (PUQA)*, then a research goal could be to reduce the resources for the quantum algorithm so that it can become a quantum algorithm with a *Practical* ***Available*** *Quantum Advantage (PAQA)*. From the flowchart, we can identify five main goals of quantum algorithms to achieve the ideal ultimate goal of *Practical Available Theoretical Scalable Quantum Advantage (PATSQA)*:

1. Find an advantage for *quantum* ***solutions*** so that they can have a *quantum* ***advantage***.
2. Find a use-case for the problem that can be solved with a quantum algorithm with a ***Nonpractical*** *Quantum Advantage (NQA)* so that the advantage can become a ***Practical*** *Quantum Advantage (PQA)*.
3. Reduce the resources of a quantum algorithm with a *Practical* ***Unavailable*** *Quantum Advantage (PUQA)* so that it can have a *Practical* ***Available*** *Quantum Advantage (PAQA)*.

4. Find a theoretical reason for a quantum algorithm with a *Practical Available* ***Empirical*** *Quantum Advantage (PAEQA)* so that it can have a *Practical Unavailable* ***Theoretical*** *Quantum Advantage (PUTQA)*.
5. Scale up the device for a quantum algorithm with a *Practical Available Theoretical* ***Nonscalable*** *Quantum Advantage (PATNQA)* so that it can have a *Practical Available Theoretical* ***Scalable*** *Quantum Advantage (PATSQA)*.

Therefore, it should be noted that the categorisation of a quantum algorithm is not fixed. A quantum algorithm could initially be solving a nonpractical problem, but later a use-case is found for the problem and it becomes a practical problem, and thus the quantum algorithm moves from the categorisation of ***Nonpractical*** *Quantum Advantage (NQA)* to ***Practical*** *Quantum Advantage (PQA)*. It should also be noted that empirical claims of quantum advantage can be difficult as they depend on specific experiments; it is possible that someone makes an empirical claim, but later a new experiment disproves that claim. In such a case, a quantum algorithm could have a categorisation of *Practical Available Empirical Quantum* ***Advantage*** *(PAEQA)*, but then it moves to *Practical Available Empirical Quantum* ***Solution*** *(PAEQS)*.

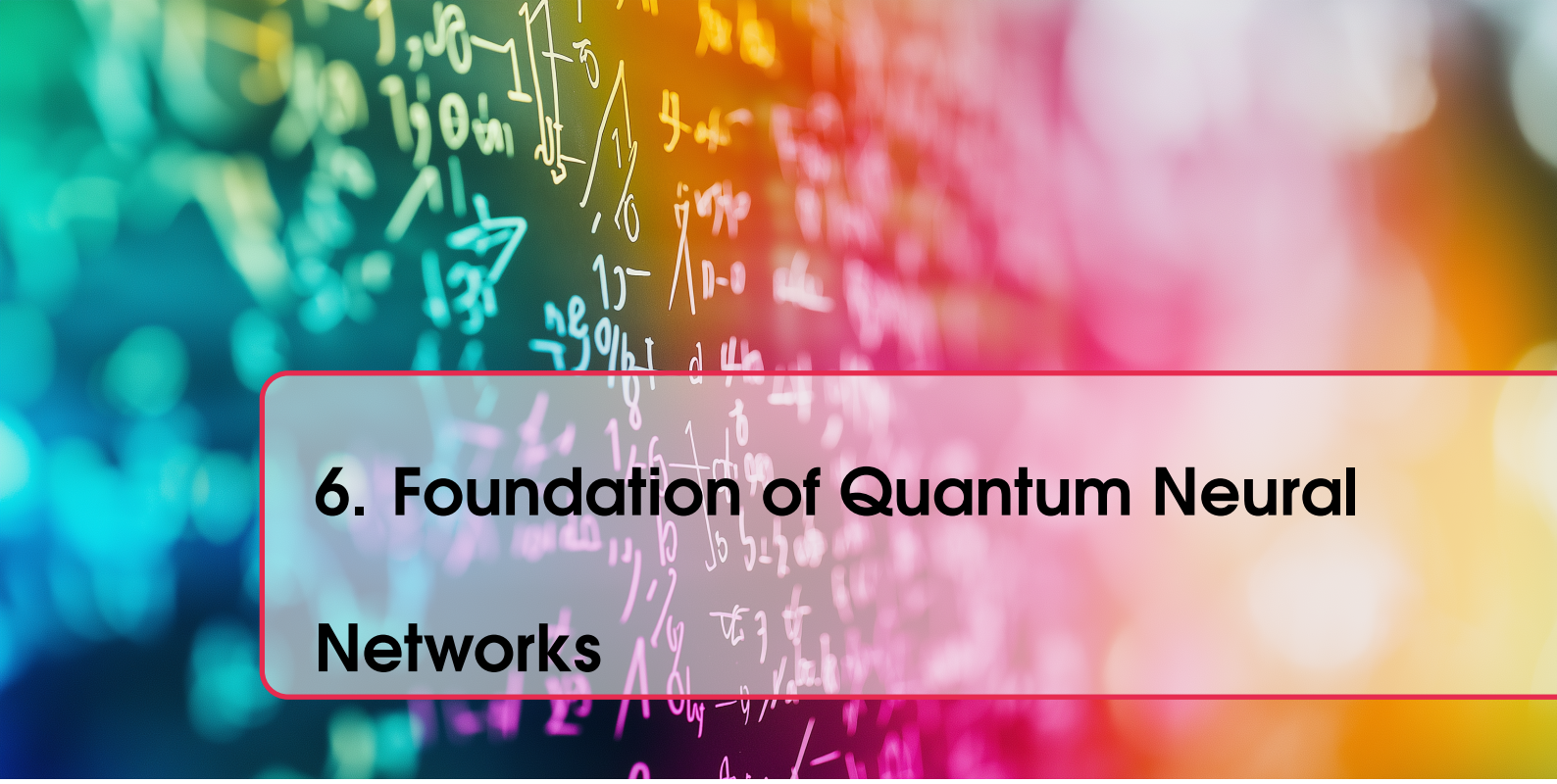
I provide several examples of categorising claims of quantum advantage or quantum solutions in the literature below:

1. **Practical Available Quantum Solution (PAQS):** Application of the Variational Quantum Eigensolver to the Ultimate Pit Problem [44] (Appendix A).
2. **Practical Available Theoretical Nonscalable Quantum Accuracy Advantage (PATNQA-A):** Experimental Quantum Advantage in the Odd-Cycle Game [198].
3. **Nonpractical Available Theoretical Quantum Speed Advantage (NATQA-S):** Power of Data in Quantum Machine Learning [28].
4. **Nonpractical Available Theoretical Quantum Speed Advantage (NATQA-S):** Quantum supremacy using a programmable superconducting processor [85].

5.4 Review

In this chapter, I presented multiple frameworks that relate to the topics in this thesis. I showed the different classes of quantum algorithms and how the

research in this thesis uses or studies these different classes. I then described a flowchart for categorising quantum advantage and quantum solutions and connected that to the research in this thesis. In the next chapter, Chapter 6, we will look at *quantum neural networks*, and understand their potential for *quantum accuracy advantage* in the context of quantum machine learning.



6. Foundation of Quantum Neural Networks

6.1 Preview 295

6.2 Summary 296

6.3 Introduction 297

6.4 Results 300

6.5 Methods 317

6.6 Discussion 324

6.7 Proofs 330

6.8 Additional Experiments and Results 348

6.9 Review 381

6.1 Preview

We have explored the history of quantum computing, its trends, and applications, including quantum machine learning in Chapter 2. In Chapter 3, we delved deeper into quantum neural networks, emerging from the context of quantum machine learning, and the technicalities of how they work and the literature about them. In Chapter 4, we looked into classical neural networks and the theoretical and practical trends in the field that led to an understanding of simplicity bias in deep neural networks. In Chapter 5, we clarified the different types of quantum algorithms related to this thesis and how they fit together, as well as categorisations for quantum advantage. This chapter, Chapter 6, ties together the background information and frameworks I constructed in these previous chapters and presents original research on simplicity bias in quantum neural networks. As explained in the previous chapters, still little was understood about the use of quantum neural networks as general-purpose learning tools, as previous work in the literature typically focused on specific applications of quantum neural networks with a known quantum speedup. At the same time, research papers in the classical deep learning field argued that deep neural networks are successful as general-purpose learning tools on

real-world data because of their inherent simplicity bias. In this chapter, our goal is to answer the question of whether quantum neural networks have a simplicity bias, which would give us insight into their performance as general-purpose learning tools. This research illuminates the potential of quantum neural networks to have *quantum accuracy advantage*, as we are looking at their ability to make accurate predictions on unseen data. This chapter aims to understand the *foundation* of quantum algorithms, the first conceptual layer of the framework presented in Chapter 1, Section 1.2. This chapter is based on the paper titled 'Do Quantum Neural Networks have Simplicity Bias?' [36].

6.2 Summary

One hypothesis for the success of deep neural networks (DNNs) is that they are highly expressive, which enables them to be applied to many problems, and they have a strong inductive bias towards solutions that are simple, known as simplicity bias, which allows them to generalise well on unseen data because most real-world data is structured (i.e. simple). In this work, we explore the inductive bias and expressivity of quantum neural networks (QNNs), which gives us a way to compare their performance to those of DNNs. Our results

show that it is possible to have simplicity bias with certain QNNs, but we prove that this type of QNN limits the expressivity of the QNN. We also show that it is possible to have QNNs with high expressivity, but they either have no inductive bias or a poor inductive bias and result in a worse generalisation performance compared to DNNs. We demonstrate that an artificial (restricted) inductive bias can be produced by intentionally restricting the expressivity of a QNN. Our results suggest a bias-expressivity tradeoff. Our conclusion is that the QNNs we studied can not generally offer an advantage over DNNs, because these QNNs either have a poor inductive bias or poor expressivity compared to DNNs.

6.3 Introduction

Classical deep neural networks (DNNs) are successful in a wide range of tasks because they are firstly highly expressive, leading to their ability to express the functions they are trying to learn, and secondly they have a good inductive bias, meaning that their inherent bias towards certain types of functions before seeing any data aligns well with the types of functions they are trying to learn. These two properties are key for any learning agent.

Quantum neural networks (QNNs) aim to combine the power of neural network models and quantum mechanics. The first idea integrating neural networks and quantum mechanics was in 1995 published by Kak [23], just one year after Shor published the famous Shor's algorithm for quantum prime factorisation, proving the potential of quantum computing [1]. Since then, there have been numerous proposals of QNNs [115].

In order for a QNN to be able to be a good learning agent, it should also have high expressivity and a good inductive bias. There are numerous papers looking at the expressivity of QNNs [29, 199, 200, 201]. Kubler et al. studied the inductive bias of quantum kernels [37]. This research paper aims to provide further understanding on the inductive bias of QNNs and how it relates to expressivity. Through doing this, we provide insight on how the inductive bias of QNNs compares to the inherent inductive bias of DNNs, which has been shown to be a simplicity bias. This simplicity bias is beneficial for DNNs because real-world data tends to have descriptions that are simple rather than complex. Therefore, it has been suggested DNNs can generalise well on real-world data because the DNN's inductive bias towards simple functions matches the simple descriptions of real-world data [34]. Understanding the inductive bias of QNNs

is crucial as it tells us whether QNNs are a good learning agent and it opens the pathway to seeing where QNNs have a disadvantage or advantage over classical neural networks. In particular, understanding whether QNNs have a simplicity bias can shed light on whether they will perform well on real-world data.

Several works have explored the inductive bias of DNNs by studying their output when given Boolean data [178]. In this work, we explore the inductive bias of QNNs by studying their output when given Boolean data. We study how the inductive bias changes when using different encoding methods, which describes the way the input data should be encoded into the QNN. QNNs are closely related to kernel methods, which map the input data into a higher dimensional space [125]. Quantum kernels have been used in analysing quantum advantage in quantum machine learning [28] and the inductive bias of quantum kernels has been explored in Reference [37]. We also use our methods to investigate the inductive bias of quantum kernels and offer comparisons to the corresponding QNNs.

6.4 Results

There are different methods for encoding data into a QNN and our work shows that each method changes the inductive bias of the QNN. Our work sheds light on why some encoding methods perform better than others as we show the effect of the encoding method on the inductive bias. In particular, we look at four major types of encoding methods: Basis Encoding, ZZ Feature Map, Amplitude Encoding, and Random Relu Transform.

To categorise the inductive bias produced by the encoding methods, we choose two ways to quantify the types of functions the QNN produces: entropy and Lempel-Ziv (LZ) complexity. We calculate a Boolean function from the outputs of the QNN given different inputs, where the output is a thresholded value of the expectation value of a measurement operator on the readout qubit, and the input is the Boolean dataset (see the *Methods* Section 6.5.1 for more detail). We represent this Boolean function as a binary string in order to measure the complexity of the Boolean function. For example, a Boolean function that maps the input 00 to output 1, input 01 to output 1, input 10 to output 0, and input 11 to output 1 is the binary string 1101. To measure the complexity of

this string (and therefore the Boolean function), we can use the LZ complexity, which counts the number of distinct substrings that appear in the string. In effect, this measures how well the string can be compressed by identifying and encoding repeated patterns or substrings. The entropy is $\min\{p, n-p\}$, where p is the number of 0s in the string and n is the length of the string. Measuring the LZ complexity tells us whether a neural network has a bias towards low-complexity functions, known as simplicity bias. Measuring the entropy tells us whether a neural network has a bias towards low-entropy functions, which is a trivial bias because such a neural network is just predicting the output of the neural network (which is either 0 or 1) with the highest frequency.

Many low-complexity functions are also low-entropy, so if we were just to measure the LZ complexity of the functions, it would seem that many of the encoding methods have a bias towards low-complexity functions, but by measuring entropy as well, we can distinguish the encoding methods that have a bias towards low-complexity and those that have a bias towards low-entropy.

We want to make this distinction as entropy is not a measure of the 'true' complexity of a function, but rather a measure of the frequencies of 0s and 1s.

The entropy of a binary string depends on the probability of a 0 or a 1 appearing in the string. The entropy is highest when the number of 0s and 1s is equal, as one has maximum uncertainty about the content of the string. The entropy is lowest when the string is all 0s or all 1s, as one has minimum uncertainty about the content of the string. When a binary string has low-entropy, then one knows there is a high number of 0s or 1s; if a neural network is given a learning task for such a function, and the neural network produces functions with low-entropy, then the neural network is just predicting whether it is 0 or 1 that has the highest frequency, which is a trivial task. Therefore, we consider the encoding methods that have a bias towards low-entropy functions as trivial. On the other hand, when a neural network is given a learning task on a binary string that has low-LZ-complexity, then a successful neural network is not necessarily producing 0s and 1s based on some frequency, but is rather predicting the simplicity in the pattern structure of the string, which is a truer form of complexity (rather than frequencies) and not trivial. Therefore, we consider the encoding methods that have a bias towards low-LZ-complexity functions to be more advanced as they also recognise structural simplicity rather than just frequencies.

That being said, there are many low-entropy functions that are low-complexity, as binary strings with a higher number of 0s or 1s tend to be simpler to describe (such as 0000). In such a case, the low-LZ-complexity of this function is based on the low entropy of the function. Therefore, we consider the low-entropy classification of this function to override its low-complexity classification. This is why we state that certain encoding methods have a low-entropy bias, because although they do have a bias towards certain low-LZ-complexity functions, these functions also have low-entropy. Furthermore, there are low-LZ-complexity functions (such as 1010) that have a high entropy and the encoding methods with a low-entropy bias do not have a bias towards these low-LZ-complexity high-entropy functions.

6.4.1 Summary of the Bias and Expressivity of Different Encoding Methods

In Table 6.1, we summarise the findings of our results for the inductive bias and expressivity of QNNs with different encoding methods.

Encoding method	**Inductive Bias**	**Expressivity (for $n = 3$ system)**
Basis encoding	**Very Poor** No Inductive Bias	**Good** Full expressivity
ZZ feature map	**Poor** Low-entropy bias (and bias towards parity function)	**Good** Full expressivity
Random relu transform	**Poor** Low-entropy bias	**Good** Full expressivity
Amplitude encoding	**Good** Low-complexity bias (i.e. simplicity bias)	**Poor** Can not express parity function

Table 6.1: **Summary of inductive bias and expressivity produced by different encoding methods.** A low-entropy bias means that the neural network produces functions with a low entropy with a greater probability than functions with a high entropy. Similarly, a low-complexity bias means that the neural network produces functions with a low Lempel-Ziv (LZ) complexity with a greater probability than functions with a high LZ complexity. *Full expressivity* means that the QNN with that encoding method can express all of the possible Boolean functions. See Section 3.4.2.1 for more information about the expressivity of QNNs. For the $n = 3$ system, there are $2^{2^3} = 256$ possible Boolean functions so *Full expressivity* means that all 256 of these functions can be expressed. The expressivity of basis and amplitude encoding have been proven (see Section 3.4.2.2 and 6.7.1 respectively). The expressivity of the ZZ feature map and random relu transform were verified with 10^7 and 10^4 samples respectively from the QNN.

Our results show that basis encoding has no inductive bias. This result is also shown by various papers in the literature [37, 125]. We also have come up with a proof-by-induction that arrives at the same conclusion. We prove that a fully expressive random QNN with basis encoding has no inductive bias. We prove this by induction by decomposing a QNN into layers that depend on previous layers. Details of this proof are in Section 6.7.2.

Our experimental results also reveal that other encoding methods can induce various types of inductive bias into the QNN. The ZZ feature map and random relu transform have a trivial bias, that is a bias towards low-entropy functions. These encoding methods perform poorly (i.e. have high generalisation error) on functions with low complexity but high entropy, showing that the QNN with these encoding methods have a low-entropy bias but not a low-complexity bias. Low complexity, high entropy functions, such as '01' repeated (i.e. $0101\ldots1010$) are representative of patterns in classical data and therefore this shows a disadvantage on these types of classical boolean data compared to the DNN, which has a bias towards low-complexity functions and can therefore generalise well on simple functions. We show, however, that the QNN with the ZZ feature map has a strong inductive bias towards the parity function, even though the

parity function has high complexity and high entropy.

Amplitude encoding has the closest bias to that of a DNN, a simplicity bias. Using this encoding method, however, limits the expressivity of the QNN – in particular, the parity function can not be expressed for the $n = 3$ system and more functions can not be expressed for larger systems. Therefore, there seems to be a bias-expressivity tradeoff. An encoding method with a good inductive bias has lower expressivity, and encoding methods with poor inductive bias have higher expressivity.

6.4.2 Summary of Results

1. Amplitude encoding has a simplicity bias but limits the expressivity of the QNN, which we prove can not express the parity function (see Section 6.4.3 for the inductive bias results and Section 6.7.1 for the proof).
2. ZZ feature map has a low-entropy bias, which does not match the inherent inductive bias of a DNN (see Section 6.4.3).
3. Random relu transform also has a low-entropy bias (see Section 6.4.3).
4. QNNs with basis encoding produce no inductive bias on the Boolean

system when fully expressive, which we prove by induction for random QNNs (see Section 6.7.2) and show numerically (see Section 6.4.3).

5. QNNs can exhibit an 'artificial' inductive bias on the Boolean system when not fully expressive and using basis encoding (see Section 6.4.3).
6. Inducing a certain inductive bias into a QNN can negatively affect its ability to train and generalise (see Sections 6.4.4, 6.8.4, 6.8.3, 6.8.6).
7. A good generalisation performance could be obtained for a QNN by constructing an encoding method that induces an inductive bias towards the function one is learning – this can be seen for the ZZ feature map which has an inductive bias towards the parity function and low generalisation error (see Section 6.4.4 and Section 6.8.4)
8. A QNN will have a poor generalisation performance on functions it has a weak bias towards and these functions can be fairly simple functions that a DNN has a strong bias towards (see Section 6.4.4 and Section 6.8.4).
9. The inductive bias of quantum kernels aligns with the inductive bias of QNNs (see Section 6.8.2).
10. There is a correlation between the probability of sampling a function from a QNN and its generalisation error. A function that appears more

likely in sampling has a lower generalisation error (see Section 6.8.4).

Section 6.8 provides details on additional experiments and results.

6.4.3 Inductive Bias of QNNs

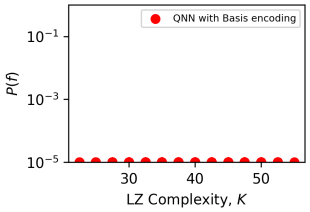
We study the inductive bias of QNNs by studying how the QNN interacts with Boolean data. Background information on the Boolean system can be read in Section 4.8.1 and on the QNN can be read in Section 3.4.

We are interested in how the probability of generating a function from an untrained fully-expressive QNN changes as the complexity of that function changes. Additional details about the fully expressive QNN are in Section 3.4.2.2. In order to generate such data, we encode the Boolean dataset (i.e. the input), via an encoding method, into a fully expressive QNN with randomly sampled parameters and measure the QNN's output. The mapping from the input to output is the Boolean function f. We generate many Boolean functions (i.e. the samples) from the QNN by changing the randomly sampled parameters. Then, we can calculate the probability of a particular Boolean

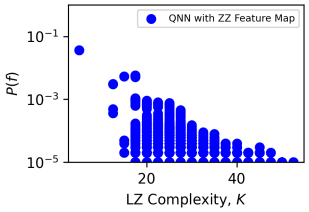
function occurring in the samples, which we denote as $P(f)$ in Figure 6.1. We measure the complexity of the Boolean function, using the Lempel-Ziv (LZ) complexity measure and plot the probability of a boolean function versus its LZ complexity. Further details about this method can be read in the Methods Section 6.5.1.

Figure 6.1 shows $P(f)$ vs K, where K is the Lempel-Ziv complexity of the boolean function, for different encoding methods. Figure 6.1e is the plot for a classical DNN for comparison. The DNN has a bias towards low-complexity Boolean functions, which is the 'simplicity bias' as proposed in Reference [178].

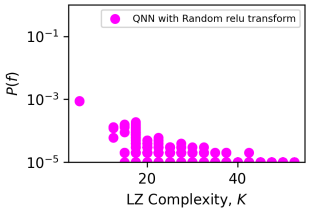
We can see in Figure 6.1a that the probability of generating a particular function does not change with the complexity of the function, implying that such a QNN is generating boolean functions randomly. This means that encoding the data using basis encoding produces no bias in the system. As a result, using basis encoding results in a poor learning agent, which corresponds with our proof for these numerical results in Section 6.7.2.



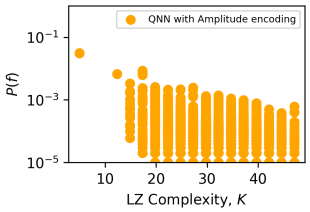
(a) Prior $P(f)$ vs complexity for fully expressive QNN with basis encoding



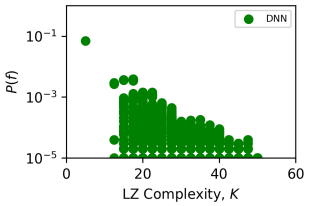
(b) Prior $P(f)$ vs complexity for fully expressive QNN with ZZ Feature Map



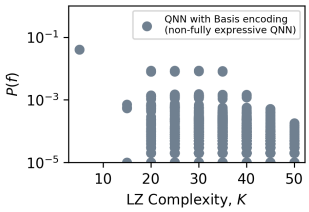
(c) Prior $P(f)$ vs complexity for fully expressive QNN with random relu transform



(d) Prior $P(f)$ vs complexity for fully expressive QNN with amplitude encoding



(e) Prior $P(f)$ vs complexity for classical DNN



(f) Prior $P(f)$ vs complexity for non-fully expressive QNN with basis encoding

Figure 6.1: **Probability of a boolean function (f) versus its complexity for five data qubits and 10^5 samples.** $P(f)$ versus Lempel-Ziv complexity, K for a fully expressive QNN with (a) basis encoding (red), (b) ZZ feature map (blue), (c) random relu transform (pink), (d) amplitude encoding (orange), (e) a DNN (green), and (f) a non-fully expressive QNN with basis encoding (grey). $P(f)$ is calculated by generating 10^5 samples of functions from the QNN by using random samples of parameters Θ over a uniform distribution. 10^7 samples were generated from the DNN and 10^5 of the samples were randomly selected to match the scale of the QNN.

For the other encoding methods, functions with a lower LZ complexity have a higher probability of occurring, implying a bias towards low-complexity functions. As mentioned earlier in this text, however, many low-complexity functions are also low-entropy functions. Therefore, additional experiments are needed, as shown in Section 6.4.4 and Section 6.8.1.2, to reveal that some of these encoding methods actually have a bias towards low-entropy functions.

Figure 6.1f shows the inductive bias in a non-fully expressive QNN with basis encoding. This QNN was proposed in Reference [19], which is described in further detail in Section 3.4.2.3. We can see that in contrast to the fully expressive quantum circuit with basis encoding, we seem to see a bias towards simple functions. We prove that fully expressive random QNNs with basis encoding have no inductive bias in Section 6.7.2. Therefore, one of the reasons we see a bias in Figure 6.1f is because the QNN can not express all functions. As the QNN is restricted, it can only produce certain functions which leads to this 'artificial' inductive bias. The problem, however, with inducing a bias this way is that the QNN will not be able to learn effectively on functions it can not express, which reduces the generalisation performance of the QNN.

6.4.4 Generalisation Error of QNNs

The generalisation error informs us of how good a learning agent the neural network is, as it is a measure of the trained neural network's error on unseen data. A low generalisation error means that the neural network can predict unseen data well. Therefore, looking at the QNN's generalisation error on unseen random boolean functions informs us of how good a learning agent the QNN is and how it compares to a classical DNN. Looking at the generalisation error can also illuminate the inductive bias of the QNNs with different encoding methods.

To study this, we train the QNN with an encoding method to learn a particular boolean function, which is the target function. To bypass barren plateau issues [25] with traditional training methods, which make it difficult to obtain zero error on the training set, we instead train the QNNs by finding a set of parameters that obtain zero error on the training set. To do this, we randomly sample a set of parameters and use them in the QNN and then run the QNN with the training inputs and obtain the predicted y_{predict} label. We calculate the training error and repeat this process until we obtain a set of parameters

where the training error is zero. Once we have obtained that set of parameters (which ensures that that the QNN is trained to zero training error) we use these parameters (i.e. the trained QNN) to calculate the error (i.e. the generalisation error) on the test set. More details about the training method can be read in Section 6.8.3.

We find the generalisation error for target functions with different complexities. We have constructed certain target functions so that they have the properties we want to study, in particular functions with low-entropy and low-complexity, low-entropy and high-complexity, high-entropy and low-complexity, and high-entropy and high-complexity. We also generate random target functions. More details about the target functions and further results on the generalisation error can be read in Section 6.8.4.

Figure 6.2 shows how the generalisation error (i.e. test error) varies by the Lempel-Ziv (LZ) complexity and entropy of the function across different encoding methods.

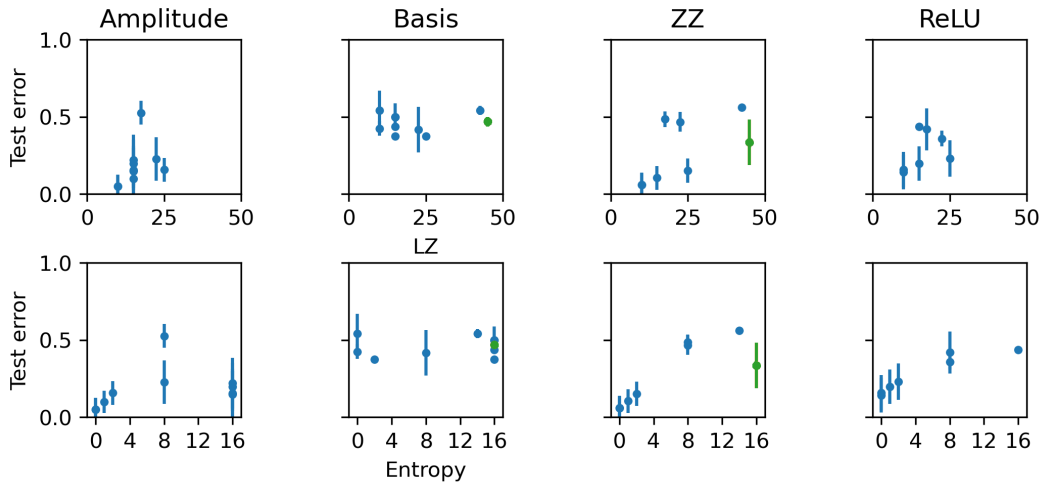


Figure 6.2: **Generalisation error versus Lempel-Ziv (LZ) complexity (top figures) and entropy (bottom figures) for five data qubits for a fully expressive QNN with different encoding methods.** The QNN is trained to zero training error on a training set S of size $m = 16$ and the generalisation error is calculated on the remaining 16 functions. The QNNs are trained by randomly sampling parameters 10^5 times and finding the generalisation error on the set of parameters that obtain zero training error. Error bars are one standard deviation. The data points show the error on the test dataset for different target functions with different LZ complexties and entropies. The maximum entropy is 16, as the boolean functions have length $2^5 = 32$ and entropy is the minimum number of 0s or 1s in the function. The green datpoint is the parity function.

For basis encoding, the test error is around 0.5 for all values of complexity and entropy, which is to be expected, as it has no inductive bias and acts as a random learner. A random learner randomly predicts outputs on the test set and has a 50% chance of obtaining the correct output. Looking at the generalisation error further confirms this inductive bias.

For amplitude encoding, the test error increases as the LZ complexity of the function increases, which implies an inductive bias towards low-complexity functions. Low-complexity functions have a low test error, as such a QNN is most likely to predict simple functions when given the test set given its bias towards low-complexity functions. There is no straightforward pattern of the test error as the entropy increases, however, implying there is no bias towards low-entropy functions. We see that high-entropy functions can have a low test error.

For the ZZ feature map and random relu transform, the test error generally increases as the complexity increases, but this pattern is stronger as the entropy increases, implying a bias towards low-entropy functions. For the ZZ feature map, we can see that there are a couple of target functions that have a low-moderate complexity (below 25) but generalise poorly, with a test error around 0.5. Further inspection into these target functions reveal that they have a high entropy, showing that the ZZ feature map does not have a bias towards low-complexity functions when the entropy is high.

6.4.5 Expressivity of QNNs

A stronger inductive bias seems to imply a weaker expressivity and a weaker inductive bias seems to imply a stronger expressivity for the encoding methods we have studied.

We have shown that amplitude encoding has a simplicity bias, similar to the bias of the DNN. Amplitude encoding, however, limits the expressivity of the QNN. This can be a problem because we would need to know something about our function to know whether it could be expressed by a QNN with amplitude encoding.

In 1969, Minksy published a book *Perceptrons* [166] that highlighted one of the limitations of the perceptron, which was that it was not fully expressive; it could not express the XOR function, which is the parity function for two inputs. The parity function's value is one if and only if the input vector has an odd number of ones. Similarly, we highlight that the QNN with amplitude encoding has a limitation, which is that it is not fully expressive and can not express the parity function. The details of the proof are in Section 6.7.1. As

the system gets larger, the QNN is unable to express more functions. For the $n = 3$ system, the QNN can not express a minimum of two Boolean functions; they are the parity function (01101001) and its reverse (10010110). For the $n = 4$ system, the QNN with amplitude encoding can not express a minimum of 5612 Boolean functions, including the parity function.

On the other hand, a QNN with basis encoding is fully expressive as is shown in Section 3.4.2.2. A QNN with the ZZ feature map and random relu transform are also highly expressive, as we verify for the $n = 3$ system that they are able to express all of the possible Boolean functions by generating 10^7 and 10^4 samples from the QNN respectively. These encoding methods, however, suffer from a weaker inductive bias, which is a low-entropy bias.

6.5 Methods

6.5.1 Finding Inductive Bias in QNNs

In order to find the inductive bias in a QNN, we implement these steps:

(a) Generate Boolean data: We generate all possible Boolean data given n

data qubits. There are 2^n data points. For example, for $n = 2$, our input data $\vec{x}$ is $x_0 = (0, 0)$, $x_1 = (0, 1)$, $x_2 = (1, 0)$, $x_3 = (1, 1)$.

(b) Create a random set of parameters: We create a random set of parameters by sampling from the uniform distribution. These set of parameters are used in the variational circuit of the QNN.

(c) Encode input data into the QNN: We implement an encoder circuit using one of the encoding methods in Section 3.4.1 to encode the data into the QNN.

(d) Construct QNN: We construct a QNN that consists of the encoder circuit in Step **(c)** and the variational circuit mentioned in Section 3.4.2. The QNN has n data qubits and one readout qubit q_r which is the output of the QNN.

(e) Calculate the expectation value of the measurement operator: We calculate the expectation value $\langle\psi| \sigma_z |\psi\rangle$, where $\sigma_z = I \otimes \cdots Z_{q_r} \cdots \otimes I$, where Z is applied only to the readout qubit q_r and I is applied to all other qubits. $|\psi\rangle = U(\vec{\theta_j}) |x_i\rangle$, where $U(\vec{\theta_j})$ is the variational circuit with the parameters $\vec{\theta}$

drawn from the uniform distribution for parameter iteration j and $|x_i\rangle$ is the output state of the encoder circuit for the ith data.

(f) Threshold the expectation value: Given that we are working with the Boolean system, we threshold the expectation value. We will obtain an expectation value $\langle\sigma_z\rangle$ in the range $[-1, 1]$. If $\langle\sigma_z\rangle < 0$, then the classification of the QNN is $y = 1$. If $\langle\sigma_z\rangle \geq 0$, $y = 0$.

(g) Construct the Boolean function: We can now construct the Boolean function by repeating Steps **(c) - (f)** with all of the input data. For each input data, we map it to its output of the QNN to obtain the Boolean function. For example, for $n = 2$, if $x_0 = (0, 0)$ produces $y_0 = 0$, $x_1 = (0, 1)$ produces $y_1 = 1$, $x_2 = (1, 0)$ produces $y_2 = 1$, and $x_3 = (1, 1)$ produces $y_3 = 0$, then our boolean function $= 0110$.

(h) Calculate the probabilities of obtaining certain Boolean functions: We now repeat Steps **(b) - (g)** for different random sets of parameters. If we use m sets of parameters, we produce m data points for the boolean functions and we can calculate the probability of a particular boolean function occurring

in the m samples.

6.5.2 Finding Inductive Bias in Quantum Kernels

In order to find the inductive bias in a quantum kernel, we implement these steps:

(a) Construct the kernel matrix: The kernel matrix is $K_{ij} = \left|\langle\phi^\dagger(\vec{x}_j)|\phi(\vec{x}_i)\rangle\right|^2$, where $\phi(\vec{x}_i)$ is the statevector from the encoder circuit with input data x_i. Therefore, we can construct the kernel matrix by calculating each entry for all pairs of input data.

(b) Sample from the kernel matrix: In order to produce samples over different possible Boolean functions for the calculation of the inductive bias, we use a *multivariate normal distribution*, which is an extension of the normal (or Gaussian) distribution to multiple variables. The multivariate normal distribution is a probability distribution over possible functions that fit a set of data points (input-output pairs). Instead of using just one individual function $f(x)$ that fits a set of data points, the multivariate normal distribution allows

us to use a range of possible functions that fit the data points. The shape of this distribution is determined by the kernel matrix (known as the covariance matrix in this context), which incorporates assumptions about the properties of the functions, such as periodicity for example. Therefore, the kernel matrix determines the types of functions we sample from and thus the inductive bias of those functions.

When we sample from this multivariate normal distribution, a random function is chosen from the distribution of functions that is determined by the kernel matrix. The sample gives us the function values at the input data points. For example, if our input data points are x_1, x_2, x_3, x_4, then the sample gives us $f(x_1), f(x_2), f(x_3), f(x_4)$. The function $f(x)$ is likely to follow the correlations determined by the kernel matrix. For example, if the kernel matrix suggests that input data points x_1 and x_2 are similar, then the function $f(x)$ is likely to assign similar values to $f(x_1)$ and $f(x_2)$. When we produce a new sample from the distribution, we get new function output values for a new possible function $f'(x_1), f'(x_2), f'(x_3), f'(x_4)$.

The multivariate normal distribution formula is shown in Equation 6.1

$$f_{\mathbf{X}}\left(x_1, \ldots, x_k\right) = \frac{\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\mathbf{K}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)}{\sqrt{(2\pi)^k|\mathbf{K}|}}, \tag{6.1}$$

where $\mathbf{x}$ is the input data, $\mathbf{K}$ is the kernel matrix, and the mean $\boldsymbol{\mu} = 0$.

(c) Threshold the sample: We threshold $f_{\mathbf{X}}$ so that if $f_{\mathbf{X}_i} < 0$, where i is the ith bit in the sample, then the ith bit of the thresholded sample y is $y_i = 1$. If $f_{\mathbf{X}_i} \geq 0$, then $y_i = 0$. The thresholded version of $f_{\mathbf{X}}$ is equivalent to the Boolean function.

(d) Calculate the probabilities of obtaining certain Boolean functions: We sample m times from the multivariate normal distribution 6.1 with the specified kernel matrix, which allows us to calculate the probability of a boolean function from the m samples, similar to Step **(h)** in Section 6.5.1.

6.5.3 Comparison Between the Methods

Table 6.2 shows the comparison between finding the inductive bias in QNNs versus finding the inductive bias in quantum kernels.

Task	Performing the Task on a QNN	Performing the Task with a Quantum Kernel
Encoding the input data $\vec{x}$	The data is encoded using an encoder circuit as described in Step (c) in Section 6.5.1.	The data is encoded using a quantum feature map (which can be implemented with a QNN) and a kernel matrix is constructed from these encoded quantum states as described in Step (a) in Section 6.5.2.
Obtaining the output $f(\vec{x})$	The output is the thresholded value of the expectation value of a measurement operator on the readout qubit as described in Step (e) in Section 6.5.1.	The output is the thresholded value of the output of the function $f_{\mathbf{X}}$ given an input $\vec{x}$, where $f_{\mathbf{X}}$ is a random function obtained from the multivariate normal distribution determined by the kernel matrix as described in Step (c) in Section 6.5.2.
Randomly sampling different functions f	For each set of parameters of the variational circuit, we define a different function and thus by sampling new sets of parameters from the uniform distribution we sample different functions as described in Step (b) in Section 6.5.1. From these functions, we can calculate the probabilities of the Boolean functions as described in Step (h) in Section 6.5.1.	We generate multiple samples from the multivariate normal distribution which chooses different functions for each sample and gives us the function outputs $f_{\mathbf{X}}$ as described in Step (b) in Section 6.5.2. From these functions, we can calculate the probabilities of Boolean functions as described in Step (d) in Section 6.5.2.

Table 6.2: **Comparison of the methods for calculating the inductive bias in QNNs vs quantum kernels.** This table shows the difference between the input data encoding, output calculation, and function sampling tasks for QNNs and quantum kernels.

6.5.4 Training QNNs

We train the QNN to learn a particular boolean function, which is the target function. In order to construct the training and test set, we randomly select half of the indices of the bits in the target function for the training set and the other half for the test set. The training inputs then consist of the Boolean input data at the indices of the bits of the training set. The test inputs consist of the Boolean input data at the indices of the bits of the test set. The training labels are the values of the bits at the indices of the training set in the target function and the test labels are the values of the bits at the indices of the test set in the target function. Further details and an example are in Section 6.8.3.

6.6 Discussion

The success of deep neural networks (DNNs) has prompted research into what makes DNNs good learning agents. Recently, it has been argued that their natural bias towards simple functions enables them to generalise well on real-world data, which also has simple descriptions [178]. The aim of this paper was to investigate the inductive bias of QNNs and answer whether they also have

simplicity bias, to give us a clue into their performance on real-world data and how they compare to DNNs.

We have demonstrated that by randomly sampling parameters and seeing which boolean functions are generated by a QNN, we can see which functions are more likely to be produced and therefore which functions the QNN has a biased towards. Unlike DNNs, which naturally have simplicity bias, the inductive bias in QNNs is produced via the encoding method of the QNN.

Our experimental results, along with a proof-by-induction, show that a fully-expressive random QNN with basis encoding produces no inductive bias, which makes it essentially a random learner. An inductive bias can be induced into the circuit by using a different encoding method. Our results have shown that a simplicity bias is possible by using amplitude encoding, but such an encoding method limits the expressivity of the QNN, which is a disadvantage compared to DNNs which are still highly expressive with their natural simplicity bias. If one is using amplitude encoding, then one would need to understand the dataset to understand whether the function could be expressed. This requires some manual intervention and is a limitation compared to DNNs that can

assume to express the functions in one's dataset.

Our results also show that it is possible to have encoding methods that do not limit the expressivity of the QNN, such as the ZZ feature map, but then the inductive bias is poor, as it has an inductive bias towards low-entropy functions. This inductive bias is poor because low-complexity, high-entropy functions are representative of patterns in classical data and therefore this shows a disadvantage on these types of classical boolean data compared to the DNN. These results imply a bias-expressivity tradeoff.

Furthermore, an inductive bias can also be induced into the circuit by restricting the expressivity of the circuit – however, this is an 'artificial' inductive bias and can negatively impact the performance of the QNN if a solution function can not be expressed by the QNN. In addition, we show that if an encoding method has a strong bias towards a particular function (as is the case with the ZZ feature map on the parity function), then it can have strong generalisation performance on that function.

This, however, requires manual intervention and is somewhat equivalent to

'hard-coding' the bias into the QNN. DNNs, however, generally don't need to be 'hard-coded' and can be used as they are. Moreover, constructing such encoding methods with a bias towards a particular function leads to a weak bias towards other functions which the DNN has a strong bias towards, even simple functions such as the '01' repeated boolean function.

Overall, we show that there is a fundamental difference between QNNs and DNNs: that DNNs have an inherent inductive bias towards simple functions, whereas the inductive bias of QNNs depends on the encoding method and whilst a simplicity bias can be obtained with amplitude encoding, the expressivity of the QNN becomes limited, implying a bias-expressivity tradeoff.

6.6.1 Limitations and Future Work

In our work, we studied common encoding methods, but there are other encoding methods, such as data re-uploading described in Section 3.4.1.5. The results we have shown depend on the encoding methods we have studied. It is possible that other encoding methods, such as data re-uploading, could introduce non-linearities and change the expressivity and/or inductive bias of

the QNNs. Future work could use our methods to study the effects of these other encoding methods, particularly data re-uploading, on the expressivity and/or inductive bias of QNNs.

In addition, we have made some assumptions on the variational circuit as well as the measurement operators we used for our analysis. We chose a variational circuit that can express all possible Boolean functions, so that the QNN has the *option* to choose all possible Boolean functions, but the functions it actually chooses inform us about the inductive bias of the QNN. We also chose a non-fully expressive circuit based on the variational circuit introduced in Reference [19], so that the QNN only has the option to choose a restricted set of Boolean functions, which then produces a restricted inductive bias. Therefore, the results based on the QNN are dependent on these assumptions, which limits the generality of the results on QNNs. To make our results more general, we investigated the inductive bias of the quantum kernels, which do not depend on a variational circuit. The results of the quantum kernel aligned well with those of the QNN.

Additionally, calculating the inductive bias can be expensive as we need to

generate all 2^n Boolean input data and there are 2^{2^n} possible Boolean functions (although we don't necessarily need to generate 2^{2^n} samples to observe the inductive bias as we can see the inductive bias from a smaller number of samples). Therefore, we can only generate the 2^n Boolean input data for a small n and our results are dependent on these small examples. It is possible that the behaviour of the QNNs changes as n grows larger and the inductive bias changes. Our results are dependent on small n and therefore may not apply to larger n. Future work could study how the size of n affects the inductive bias using our methods. Moreover, it may be possible to calculate the inductive bias using a smaller number of Boolean input data (and therefore not generating all 2^n Boolean input data); future work could adapt the techniques in this work to make the calculations more efficient. As well as that, future work could look at different measures of expressivity (for example, looking at only a subset of Boolean strings) that may make the calculations more efficient.

Furthermore, future work could explore the bias-expressivity tradeoff in the context of QNNs. With the limitations of inductive bias and expressivity in the current framework of QNNs, future work could explore alternative frameworks of QNNs.

6.7 Proofs

6.7.1 Proof of Expressivity of Amplitude Encoding

In this section, we prove that an arbitrary QNN with amplitude encoding can not express the parity function for $n = 3$.

Step 1: Calculate the initial statevector. The first step is to obtain the initial encoded statevector, which is the statevector of the QNN after the Boolean data has been encoded into the QNN, but before the variational quantum circuit has been applied; this statevector is $|\psi_i\rangle$.

Step 2: Calculate the final statevector. We wish to obtain $|\psi_f\rangle = U|\psi_i\rangle$, where $|\psi_f\rangle$ is the final statevector after applying the variational quantum circuit U to the initial encoded statevector. For the proof, we choose an arbitrary variational quantum circuit U.

Step 3: Calculate the expectation value. We want to find the expectation value, which is $\langle\sigma_z\rangle = \langle\psi_f|\sigma_z|\psi_f\rangle$, where $\sigma_z = I \otimes I \otimes Z$ for three qubits, as Z

is applied to the readout qubit q_r and the identity matrix I is applied to the data qubits.

Step 4: Constrain the expectation value to match the function. We now put a constraint on the expectation value, so that it matches the function we are checking that the QNN can express. Our constraint is $\langle \sigma_z \rangle_i = f_i$, where f is the function and i is the index of the function. If $\langle \sigma_z \rangle_i \geq 0$, then $f_i = 0$; if $\langle \sigma_z \rangle_i < 0$, then $f_i = 1$. We use the Z3 solver [202] to check the satisfiability of these formulas.

6.7.1.1 Example

This example shows the calculation of the above steps to prove that an arbitrary QNN with amplitude encoding can not express the parity function for $n = 3$.

Step 1: Calculate the initial statevector. For the $n = 3$ boolean dataset, we need two qubits to encode the Boolean data using amplitude encoding. Table 6.3 shows how we encode the $n = 3$ dataset into two qubits and the process for calculating $|\psi_i\rangle$.

Boolean Data	**Process of encoding the data**	**Normalised encoded state**	**State-vector of data qubits $\|\phi_i\rangle$**	**State-vector with read-out qubit $\|\psi_i\rangle$**
000	N/A	N/A	N/A	N/A
001	$0\cdot\|00\rangle+0\cdot\|01\rangle+1\cdot\|10\rangle+0\cdot\|11\rangle$	$\|10\rangle$	$\begin{pmatrix}0\\0\\1\\0\end{pmatrix}$	$\begin{pmatrix}0\\0\\0\\0\\1\\0\\0\\0\end{pmatrix}$
010	$0\cdot\|00\rangle+1\cdot\|01\rangle+0\cdot\|10\rangle+0\cdot\|11\rangle$	$\|01\rangle$	$\begin{pmatrix}0\\1\\0\\0\end{pmatrix}$	$\begin{pmatrix}0\\0\\1\\0\\0\\0\\0\\0\end{pmatrix}$
011	$0\cdot\|00\rangle+1\cdot\|01\rangle+1\cdot\|10\rangle+0\cdot\|11\rangle$	$\frac{1}{\sqrt{2}}\|01\rangle+\frac{1}{\sqrt{2}}\|10\rangle$	$\frac{1}{\sqrt{2}}\begin{pmatrix}0\\1\\1\\0\end{pmatrix}$	$\frac{1}{\sqrt{2}}\begin{pmatrix}0\\0\\1\\0\\1\\0\\0\\0\end{pmatrix}$
100	$1\cdot\|00\rangle+0\cdot\|01\rangle+0\cdot\|10\rangle+0\cdot\|11\rangle$	$\|00\rangle$	$\begin{pmatrix}1\\0\\0\\0\end{pmatrix}$	$\begin{pmatrix}1\\0\\0\\0\\0\\0\\0\\0\end{pmatrix}$
101	$1\cdot\|00\rangle+0\cdot\|01\rangle+1\cdot\|10\rangle+0\cdot\|11\rangle$	$\frac{1}{\sqrt{2}}\|00\rangle+\frac{1}{\sqrt{2}}\|10\rangle$	$\frac{1}{\sqrt{2}}\begin{pmatrix}1\\0\\1\\0\end{pmatrix}$	$\frac{1}{\sqrt{2}}\begin{pmatrix}1\\0\\0\\0\\1\\0\\0\\0\end{pmatrix}$
110	$1\cdot\|00\rangle+1\cdot\|01\rangle+0\cdot\|10\rangle+0\cdot\|11\rangle$	$\frac{1}{\sqrt{2}}\|00\rangle+\frac{1}{\sqrt{2}}\|01\rangle$	$\frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\\0\\0\end{pmatrix}$	$\frac{1}{\sqrt{2}}\begin{pmatrix}1\\0\\1\\0\\0\\0\\0\\0\end{pmatrix}$
111	$1\cdot\|00\rangle+1\cdot\|01\rangle+1\cdot\|10\rangle+0\cdot\|11\rangle$	$\frac{1}{\sqrt{3}}\|00\rangle+\frac{1}{\sqrt{3}}\|01\rangle+\frac{1}{\sqrt{3}}\|10\rangle$	$\frac{1}{\sqrt{3}}\begin{pmatrix}1\\1\\1\\0\end{pmatrix}$	$\frac{1}{\sqrt{3}}\begin{pmatrix}1\\0\\1\\0\\1\\0\\0\\0\end{pmatrix}$

Table 6.3: *(Caption on the next page.)*

Table 6.3: **Encoding the n = 3 boolean dataset into two qubits using amplitude encoding.** If $\vec{x}$ is the dataset, the first column shows the x_i data elements. $x_0 = 000$ can not be encoded using amplitude encoding so its following columns are Not Applicable (N/A). The second column shows the process for encoding the data into the two qubits. The data is encoded into the two-qubit state as follows: $x_{i_0}|00\rangle + x_{i_1}|01\rangle + x_{i_2}|10\rangle + 0|11\rangle$, where x_{i_j} is the jth index in the x_i data element. The $|11\rangle$ state is encoded with 0 as it is not needed for the data encoding. The third column shows the simplified, normalised, encoded state. The fourth column is showing the statevector representation of the third column. The fifth column adds the readout qubit q_r to the statevector, which is achieved by applying the tensor product of the statevector of the data qubits $|\phi_i\rangle$ with the readout qubit $|q_r\rangle = |0\rangle$ so that the encoded statevector of the QNN with the readout qubit is $|\psi_i\rangle = |\phi_i\rangle \otimes |q_r\rangle$.

Step 2: Calculate the final statevector. Equation 6.2 provides an example of obtaining $|\psi_f\rangle$, as described in Step 2, for the $x_1 = 001$ boolean data, which we indicate as $|\psi_{f_1}\rangle$.

$$|\psi_{f_1}\rangle = U|\psi_{i_1}\rangle = \begin{pmatrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & a_8 \\ b_1 & b_2 & b_3 & b_4 & b_5 & b_6 & b_7 & b_8 \\ c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 \\ d_1 & d_2 & d_3 & d_4 & d_5 & d_6 & d_7 & d_8 \\ e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 \\ f_1 & f_2 & f_3 & f_4 & f_5 & f_6 & f_7 & f_8 \\ g_1 & g_2 & g_3 & g_4 & g_5 & g_6 & g_7 & g_8 \\ h_1 & h_2 & h_3 & h_4 & h_5 & h_6 & h_7 & h_8 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} a_5 \\ b_5 \\ c_5 \\ d_5 \\ e_5 \\ f_5 \\ g_5 \\ h_5 \end{pmatrix} \quad (6.2)$$

Step 3: Calculate the expectation value. Equation 6.3 provides an example for calculating σ_z for three qubits and Equation 6.4 shows how to use σ_z to calculate the expectation value for the $x_1 = 001$ boolean data, which we indicate as $\langle \sigma_{z_1} \rangle$.

$$\sigma_z = I \otimes I \otimes Z = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{pmatrix} \tag{6.3}$$

$$\begin{aligned} \langle \sigma_{z_1} \rangle &= \langle \psi_{f_1} | \, \sigma_z \, | \psi_{f_1} \rangle \\ &= \begin{pmatrix} a_5^\dagger & b_5^\dagger & c_5^\dagger & d_5^\dagger & e_5^\dagger & f_5^\dagger & g_5^\dagger & h_5^\dagger \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} a_5 \\ b_5 \\ c_5 \\ d_5 \\ e_5 \\ f_5 \\ g_5 \\ h_5 \end{pmatrix} \\ &= a_5^\dagger a_5 - b_5^\dagger b_5 + c_5^\dagger c_5 - d_5^\dagger d_5 + e_5^\dagger e_5 - f_5^\dagger f_5 + g_5^\dagger g_5 - h_5^\dagger h_5 \end{aligned} \tag{6.4}$$

Step 4: Constrain the expectation value to match the function. We repeat the above steps for the whole Boolean dataset $\vec{x}$ and obtain the expectation value formulas $\langle \sigma_{z_1} \rangle, \langle \sigma_{z_2} \rangle \cdots \langle \sigma_{z_{n-1}} \rangle$ for the dataset. In order to simplify these formulas for the Z3 solver, we substitute some of the values in the formula for new variables. In particular, we substitute $\lambda_i^\dagger \lambda_j$ for λ_{ij}, where λ is the variable (in the case of three qubits, $\lambda \in \{a, b, c, d, e, f, g, h\}$), i is the index

of the variable with the dagger, and j is the index of the variable without the dagger (for three qubits, $i, j \in \{1, 2, 3, 4, 5, 6, 7, 8\}$). For example, the formula in Equation 6.4 becomes $a_{55} - b_{55} + c_{55} - d_{55} + e_{55} - f_{55} + g_{55} - h_{55}$. Table 6.4 shows these simplified formulas and constraints.

For $n = 3$, there are $2^{2^3} = 256$ possible Boolean functions f. Given that amplitude encoding can not encode the $x_0 = 000$ boolean function, there are $2^{2^3-1} = 128$ Boolean functions that we check that the QNN with amplitude encoding can express. We express the constraints for these 128 Boolean functions using the Z3 solver and find that the QNN with amplitude encoding can express 126 Boolean functions, except two, which are the parity function 1101001 and its reverse 0010110.

Boolean Data	**Expectation value formula $\langle\sigma_{z_i}\rangle$**	f_i	**Constraint**
000	N/A	0	N/A
001	$a_{55} - b_{55} + c_{55} - d_{55} + e_{55} - f_{55} + g_{55} - h_{55}$	1	< 0
010	$a_{33} - b_{33} + c_{33} - d_{33} + e_{33} - f_{33} + g_{33} - h_{33}$	1	< 0
011	$\frac{1}{2}(a_{33} + a_{35} + a_{53} + a_{55} - b_{33} - b_{35} - b_{53} - b_{55} + c_{33} + c_{35} + c_{53} + c_{55} - d_{33} - d_{35} - d_{53} - d_{55} + e_{33} + e_{35} + e_{53} + e_{55} - f_{33} - f_{35} - f_{53} - f_{55} + g_{33} + g_{35} + g_{53} + g_{55} - h_{33} - h_{35} - h_{53} - h_{55})$	0	≥ 0
100	$a_{11} - b_{11} + c_{11} - d_{11} + e_{11} - f_{11} + g_{11} - h_{11}$	1	< 0
101	$\frac{1}{2}(a_{11} + a_{15} + a_{51} + a_{55} - b_{11} - b_{15} - b_{51} - b_{55} + c_{11} + c_{15} + c_{51} + c_{55} - d_{11} - d_{15} - d_{51} - d + e_{11} + e_{15} + e_{51} + e_{55} - f_{11} - f_{15} - f_{51} - f_{55} + g_{11} + g_{15} + g_{51} + g_{55} - h_{11} - h_{15} - h_{51} - h_{55})$	0	≥ 0
110	$\frac{1}{2}(a_{11} + a_{13} + a_{31} + a_{33} - b_{11} - b_{13} - b_{31} - b_{33} + c_{11} + c_{13} + c_{31} + c_{33} - d_{11} - d_{13} - d_{31} - d + e_{11} + e_{13} + e_{31} + e_{33} - f_{11} - f_{13} - f_{31} - f_{33} + g_{11} + g_{13} + g_{31} + g_{33} - h_{11} - h_{13} - h_{31} - h_{33})$	0	≥ 0
111	$\frac{1}{3}(a_{11} + a_{13} + a_{15} + a_{31} + a_{33} + a_{35} + a_{51} + a_{53} + a_{55} - b_{11} - b_{13} - b_{15} - b_{31} - b_{33} - b_{35} - b_{51} - b_{53} - b_{55} + c_{11} + c_{13} + c_{15} + c_{31} + c_{33} + c_{35} + c_{51} + c_{53} + c_{55} - d_{11} - d_{13} - d_{15} - d_{31} - d_{33} - d_{35} - d_{51} - d_{53} - d_{55} + e_{11} + e_{13} + e_{15} + e_{31} + e_{33} + e_{35} + e_{51} + e_{53} + e_{55} - f_{11} - f_{13} - f_{15} - f_{31} - f_{33} - f_{35} - f_{51} - f_{53} - f_{55} + g_{11} + g_{13} + g_{15} + g_{31} + g_{33} + g_{35} + g_{51} + g_{53} + g_{55} - h_{11} - h_{13} - h_{15} - h_{31} - h_{33} - h_{35} - h_{51} - h_{53} - h_{55})$	1	< 0

Table 6.4: **The formulas and constraints used for the satisfiability tests.** The first column shows the Boolean dataset. The second column shows the simplified formula for the expectation value $\langle\sigma_{z_i}\rangle$ for the Boolean data x_i. The third column shows the parity function f and the fourth column shows the constraint that needs to be made on the formula in the second column in order to meet the value of the function in the third column. The $x_0 = 000$ data can not be encoded using amplitude encoding and therefore its columns are Not Applicable (N/A).

6.7.2 Proof That Fully-Expressive Random QNNs with Basis Encoding Have No Bias

We prove that a fully-expressive QNN of the form in Equation 6.8 (which we describe as a *random QNN*) with basis encoding has no inductive bias. We prove this by induction by decomposing a QNN with n qubits into layers that depend on previous layers.

For this proof, we use quantum channels, which describe the evolution of quantum states. The most general quantum channel can be described by Kraus' theorem which states that all quantum operations $\mathcal{E}$ (including quantum channels) on a system of Hilbert space dimension d can be generated by the sum shown in Equation 6.5 [203]

$$\mathcal{E}(\alpha) = \sum_{i=1}^{M} K_i \alpha K_i^\dagger, \tag{6.5}$$

where $1 \leq M \leq d^2$, meaning there are at most d^2 elements of the sum, K are the Kraus operators, which satisfy the condition $\sum_i K_i^\dagger K_i = I$, and α is a density matrix.

When the quantum channel is unitary, there is a single Kraus operator U and Equation 6.5 would be written as $\mathcal{E}(\alpha) = U\alpha U^\dagger$. We can combine these unitary channels probabilistically so that our quantum channel becomes the one shown in Equation 6.6 [204]

$$\mathcal{E}(\alpha) = \sum_i p_i U_i \alpha U_i^\dagger. \tag{6.6}$$

Each quantum unitary channel U_i is applied with probability p_i, so the resulting channel $\mathcal{E}$ is a probabilistic mixture of the original unitary channels. Equation 6.6 can be described as a convex combination of unitary channels, which is a linear combination of quantum unitary channels, where each quantum unitary channel is assigned a probability (therefore it is a convex combination, which requires the coefficients in the linear combination to be non-negative and summing to one).

The resulting quantum channel $\mathcal{E}$ from the convex combination of unitary channels has a Kraus representation (as shown in Equation 6.5) with the set of Kraus operators $\{\sqrt{p_i}U_i\}_i$ so that the channel becomes the one shown in

Equation 6.7

$$\mathcal{E}(\alpha) = \sum_i (\sqrt{p_i}U_i)\,\alpha\,(\sqrt{p_i}U_i)^\dagger = \sum_i p_i U_i \alpha U_i^\dagger. \tag{6.7}$$

For this proof, we restrict our attention to a convex combination of unitary channels. In other words, we use a channel D that transforms a density matrix α so that $\alpha \xrightarrow{D} \alpha' = \sum_i p_i U_i \alpha U_i^\dagger$, where p_i is the probability of picking U_i from some set. Our set of Us are infinite and therefore we integrate over all of the possible angles from 0 to 2π, so our channel makes the following transformation $\alpha \to \alpha' = \int U\alpha U^\dagger dU$, where $dU = d\theta_1 \cdots \theta_S$, where there are S parameters in U, and $\int dU = 1$. Therefore our proof is restricted to QNNs that have a variational circuit of the form in Equation 6.8

$$\alpha' = \int U\alpha U^\dagger dU. \tag{6.8}$$

where α is a density matrix, U is a unitary matrix, and dU is with respect to all of the parameters in U.

Step 1: Base Case for n = 1

We state that $U_i \in U_3$, where

$$U_3 = \begin{pmatrix} \cos(\theta/2) & -e^{i\lambda}\sin(\theta/2) \\ e^{i\phi}\sin(\theta/2) & e^{i(\phi+\lambda)}\cos(\theta/2) \end{pmatrix},$$

which implements the most general single-qubit rotation (with three parameters). We denote the U_3 gate as u_1 in this proof as the U_3 gate acts on one qubit and has three parameters as shown in Figure 6.3. Our channel makes the following transformation $\alpha \to \alpha' = \int u_1 \alpha u_1^\dagger du_1$, where $du_1 = d\theta d\varphi d\lambda$ and $\int du_1 = 1$.

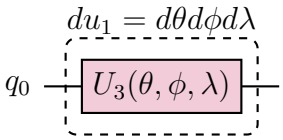


Figure 6.3: **u_1 gate.**

α is the initial density matrix and its general form can be written as $\alpha = \begin{pmatrix} \alpha_1 & \alpha_2 \\ \alpha_3 & \alpha_4 \end{pmatrix}$. Therefore we calculate

$$\begin{aligned} &\int_0^{2\pi}\int_0^{2\pi}\int_0^{2\pi} u_1 \alpha u_1^\dagger d\theta d\varphi d\lambda \\ =&\int_0^{2\pi}\int_0^{2\pi}\int_0^{2\pi} \begin{pmatrix} \cos(\theta/2) & -e^{i\lambda}\sin(\theta/2) \\ e^{i\phi}\sin(\theta/2) & e^{i(\phi+\lambda)}\cos(\theta/2) \end{pmatrix} \cdot \begin{pmatrix} \alpha_1 & \alpha_2 \\ \alpha_3 & \alpha_4 \end{pmatrix} \cdot \begin{pmatrix} \cos(\theta/2) & e^{-i\phi}\sin(\theta/2) \\ -e^{-i\lambda}\sin(\theta/2) & e^{-i(\phi+\lambda)}\cos(\theta/2) \end{pmatrix} d\theta d\varphi d\lambda \\ =&4\pi^3 \begin{pmatrix} \alpha_1 + \alpha_4 & 0 \\ 0 & \alpha_1 + \alpha_4 \end{pmatrix}. \end{aligned}$$

Because we are using basis encoding, the initial state can either be $\alpha = |0\rangle\langle 0| = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$ or $\alpha = |1\rangle\langle 1| = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$. Therefore, $4\pi^3 \begin{pmatrix} \alpha_1 + \alpha_4 & 0 \\ 0 & \alpha_1 + \alpha_4 \end{pmatrix} = 4\pi^3 \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$.

$\int_0^{2\pi}\int_0^{2\pi}\int_0^{2\pi} d\theta d\varphi d\lambda = 8\pi^3$, so we normalise the matrix to: $\frac{4\pi^3}{8\pi^3} = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{pmatrix}$.

The final density matrix is $\begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{pmatrix}$, which shows us that the probability of measuring 0 and 1 are the same and therefore there is no bias.

The base case for $n = 1$ is the following

$$\int_0^{2\pi}\int_0^{2\pi}\int_0^{2\pi} u_1 \alpha u_1^\dagger d\theta d\varphi d\lambda = 4\pi^3 \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = 4\pi^3 \cdot I = \frac{(2\pi)^{\sigma_1}}{2^1} \cdot I = r_1,$$

where σ_1 is the number of parameters for the circuit on one qubit, so $\sigma_1 = 3$. r_1 is the result of the evaluation of the integrals, which we normalise by $(2\pi)^{\sigma_1} = 8\pi^3$ so that our final density matrix is $\frac{r_1}{(2\pi)^{\sigma_1}} = \frac{(2\pi)^{\sigma_1}}{2^1(2\pi)^{\sigma_1}} \cdot I = \frac{4\pi^3}{8\pi^3} \cdot I = \frac{1}{2} \cdot I = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{pmatrix}$.

Step 2: Base case for n = 2

We state that $U_i \in U_{15}$, where U_{15} is a general two-qubit gate (implemented with 15 parameters). We use the general two-qubit gate proposed in Reference [205] that has 15 parameters. This circuit is shown in Figure 6.4.

We denote the U_{15} gate as u_2 in this proof as the U_{15} gate acts on two qubits as shown in Figure 6.4. Our channel makes the following transformation $\alpha \to \alpha' = \int u_2 \alpha u_2^\dagger du_2$, where $du_2 = d\theta_1 \cdots d\theta_{15}$ (as there are 15 parameters in the u_2 gate) and $\int du_2 = 1$.

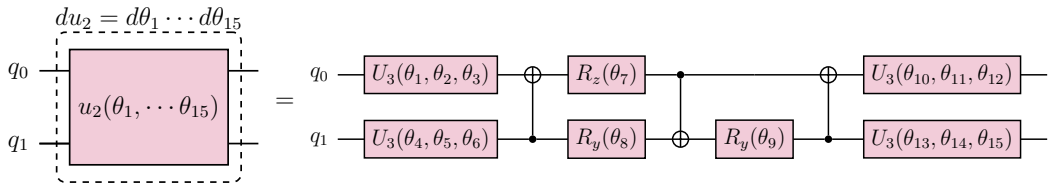


Figure 6.4: **u_2 gate.** The circuit for a general two-qubit rotation proposed in Reference [205].

α is the initial density matrix and its general form can be written as $\alpha = \begin{pmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 \\ \alpha_5 & \alpha_6 & \alpha_7 & \alpha_8 \\ \alpha_9 & \alpha_{10} & \alpha_{11} & \alpha_{12} \\ \alpha_{13} & \alpha_{14} & \alpha_{15} & \alpha_{16} \end{pmatrix}$. Therefore we calculate

$$\underbrace{\int_0^{2\pi}\cdots\int_0^{2\pi}}_{15} u_2\alpha u_2^\dagger d\theta_1\cdots d\theta_{15}$$

$$=\underbrace{\int_0^{2\pi}\cdots\int_0^{2\pi}}_{15}\underbrace{(U_3\otimes U_3)\cdot(CX_{1,0})\cdot(I\otimes R_y)\cdot(CX_{0,1})\cdot(R_z\otimes R_y)\cdot(CX_{1,0})\cdot(U_3\otimes U_3)}_{u_2}\cdot\alpha\cdot u_2^\dagger d\theta_1\cdots d\theta_{15}$$

$$=8192\cdot\pi^{15}\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix},$$

where $CX_{x,y}$ is the CX gate where the control qubit is x and the target qubit is y.

Because we are using basis encoding, the initial state can either be

$$|00\rangle\langle 00| = \begin{pmatrix}1&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\end{pmatrix}, |01\rangle\langle 01| = \begin{pmatrix}0&0&0&0\\0&1&0&0\\0&0&0&0\\0&0&0&0\end{pmatrix}$$

$$|10\rangle\langle 10| = \begin{pmatrix}0&0&0&0\\0&0&0&0\\0&0&1&0\\0&0&0&0\end{pmatrix}, |11\rangle\langle 11| = \begin{pmatrix}0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&1\end{pmatrix}.$$

As $\underbrace{\int_0^{2\pi}\cdots\int_0^{2\pi}}_{15} d\theta_1\cdots d\theta_{15} = (2^{15}\cdot\pi^{15})$,

we normalise the matrix to: $\frac{8192\pi^{15}}{2^{15}.\pi^{15}}\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix} = \begin{pmatrix}\frac{1}{4}&0&0&0\\0&\frac{1}{4}&0&0\\0&0&\frac{1}{4}&0\\0&0&0&\frac{1}{4}\end{pmatrix}$.

The final density matrix is $\begin{pmatrix}\frac{1}{4}&0&0&0\\0&\frac{1}{4}&0&0\\0&0&\frac{1}{4}&0\\0&0&0&\frac{1}{4}\end{pmatrix}$, which shows us that the probability of measuring the states $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ are equal and therefore there is no bias.

The base case for $n = 2$ is

$$\underbrace{\int_0^{2\pi}\cdots\int_0^{2\pi}}_{15} u_2\alpha u_2^\dagger d\theta_1\cdots d\theta_{15} = 8192\pi^{15}\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix} = 8192\pi^{15}\cdot I^{\otimes 2} = \frac{(2\pi)^{\sigma_2}}{2^2}\cdot I^{\otimes 2} = r_2,$$

where $I^{\otimes 2}$ is the identity matrix acting on two qubits and σ_2 is the number of parameters for the circuit on two qubits, so $\sigma_2 = 15$. r_2 is the result of the evaluation of the integrals, which we normalise by $(2\pi)^{\sigma_2} = 2^{15}\pi^{15}$ so that our final density matrix is $\frac{r_2}{(2\pi)^{\sigma_2}} = \frac{(2\pi)^{\sigma_2}}{2^2(2\pi)^{\sigma_2}}\cdot I^{\otimes 2} = \frac{8192\pi^{15}}{2^{15}\pi^{15}} = \frac{1}{4}\cdot I^{\otimes 2} = \begin{pmatrix}\frac{1}{4}&0&0&0\\0&\frac{1}{4}&0&0\\0&0&\frac{1}{4}&0\\0&0&0&\frac{1}{4}\end{pmatrix}$.

Step 3: Assume n = k holds

We decompose a QNN acting on N qubits into layers that act on $N-1$ qubits and 2 qubits [206, 207] using the u_2 gate as shown in Figure 6.5. We use this structure for the inductive step.

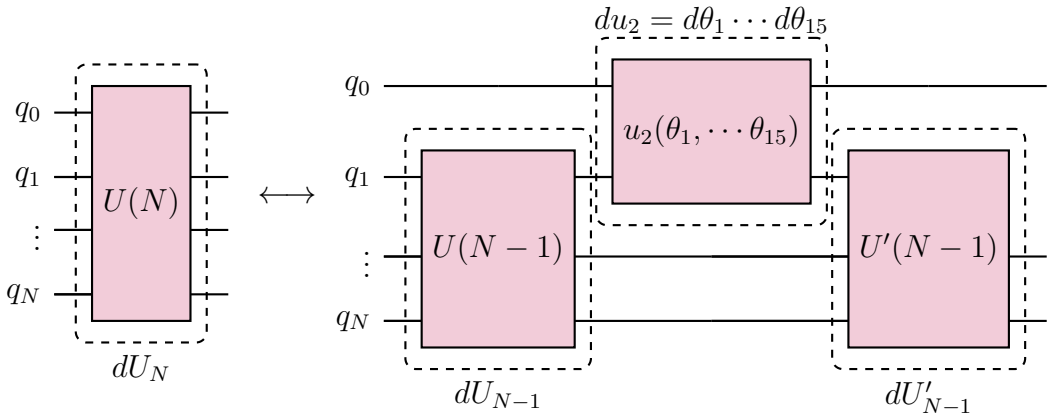


Figure 6.5: **U_N gate.** Inspired by Reference [206].

Assuming $n = k$ holds, then

$$\underbrace{\int_0^{2\pi} \cdots \int_0^{2\pi}}_{\sigma_k} \underbrace{\tilde{u}_{k-1}\tilde{u}_2\tilde{u}'_{k-1}}_{u_k} \alpha \underbrace{(\tilde{u}_{k-1}\tilde{u}_2\tilde{u}'_{k-1})^\dagger}_{u_k^\dagger} \underbrace{du_{k-1}du_2du'_{k-1}}_{du_k}$$
$$= \underbrace{\int_0^{2\pi} \cdots \int_0^{2\pi}}_{\sigma_k} u_k \alpha u_k^\dagger du_k = \frac{(2\pi)^{\sigma_k}}{2^k} \cdot I^{\otimes k} = r_k,$$

where $\sigma_k = 2\sigma_{k-1} + p$ is the number of parameters for a circuit on $n = k$ qubits,

where $p = 15$ as p is the number of parameters in a general two-qubit gate as shown in Figure 6.4. du is with respect to the parameters in gate u, for example, $du_2 = d\theta_1 \cdots d\theta_{15}$, and $\tilde{u}$ takes into account the structure of the gate u placed in Figure 6.5; for example, $\tilde{u}_2 = u_2 \otimes I^{\otimes N-2}$.

r_k is the result of the evaluation of the integrals, which we normalise by $(2\pi)^{\sigma_k}$ so that our final density matrix is $\frac{r_k}{(2\pi)^{\sigma_k}} = \frac{(2\pi)^{\sigma_k}}{2^k(2\pi)^{\sigma_k}} \cdot I^{\otimes k} = \frac{1}{2^k} \cdot I^{\otimes k}$.

Step 4: Show $\mathbf{n = k + 1}$ holds

$$\text{We want to show } \underbrace{\int_0^{2\pi} \cdots \int_0^{2\pi}}_{\sigma_{k+1}} u_{k+1} \alpha u_{k+1}^\dagger du_{k+1} = \frac{(2\pi)^{\sigma_{k+1}}}{2^{k+1}} \cdot I^{\otimes k+1},$$

where $\sigma_{k+1} = 2\sigma_k + p$.

$$\underbrace{\int_0^{2\pi} \cdots \int_0^{2\pi}}_{\sigma_{k+1}} u_{k+1} \alpha u_{k+1}^\dagger du_{k+1} = \underbrace{\int_0^{2\pi} \cdots \int_0^{2\pi}}_{\sigma_{k+1}} \underbrace{\tilde{u}_k \tilde{u}_2 \tilde{u}_k'}_{u_{k+1}} \alpha \underbrace{(\tilde{u}_k \tilde{u}_2 \tilde{u}_k')^\dagger}_{u_{k+1}^\dagger} \underbrace{du_k du_2 du_k'}_{du_{k+1}}$$

$$= \underbrace{\int_0^{2\pi} \cdots \int_0^{2\pi}}_{\sigma_{k+1}} (I \otimes u_k) \left(u_2 \otimes I^{\otimes k-1}\right) (I \otimes u_k') \alpha \Big((I \otimes u_k) \left(u_2 \otimes I^{\otimes k-1}\right) (I \otimes u_k') \Big)^\dagger du_k du_2 du_k'$$

$$= \frac{(2\pi)^{\sigma_k} \cdot (2\pi)^p \cdot (2\pi)^{\sigma_k}}{2^{k+1}} I^{\otimes k+1} = \frac{(2\pi)^{2\sigma_k + p}}{2^{k+1}} I^{\otimes k+1} = \frac{(2\pi)^{\sigma_{k+1}}}{2^{k+1}} I^{\otimes k+1} = r_{k+1}.$$

By using the assumption that $n = k$ holds, one can use the result r_k and the

base case result r_2 to substitute terms in this step, allowing one to evaluate the integrals and arrive at r_{k+1}.

This concludes the proof that the density matrix of an arbitrary circuit with parameters sampled uniformly is equivalent to

$$\underbrace{\int_0^{2\pi} \cdots \int_0^{2\pi}}_{\sigma_n} u_n \alpha u_n^\dagger du_n = \frac{(2\pi)^{\sigma_n}}{2^n} \cdot I^{\otimes n} = r_n,$$

where σ_n is the number of parameters for the circuit on n qubits. r_n is the result of the evaluation of the integrals, which we normalise by $(2\pi)^{\sigma_n}$ so that our final density matrix is

$$\frac{(2\pi)^{\sigma_n}}{2^n(2\pi)^{\sigma_n}} \cdot I^{\otimes n} = 2^{-n} \cdot I^{\otimes n},$$

which shows that the outputs have equal probability and therefore no bias. This result can also be derived in simpler ways based on the literature [208].

6.8 Additional Experiments and Results

6.8.1 Inductive Bias in QNNs

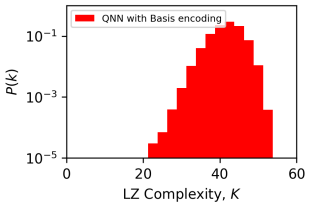
6.8.1.1 Probability of Complexity vs Complexity

We look at how the probability of functions with a certain complexity K changes with the complexity K of those functions. There are exponentially more complex functions than simple functions, so for a random learner with no bias, we'd expect that functions with higher complexity have a higher probability, as shown in Figure 6.6f. This Figure 6.6f is generated by generating random boolean functions and measuring their complexity.

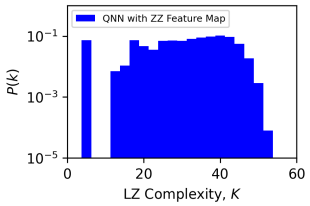
When there is a bias towards simple functions as the DNN exhibits as shown in Figure 6.6e, the probability of the complexity flattens across complexities as the DNN's exponential bias towards simple functions counteracts the exponentially more complex functions.

The QNN with basis encoding has a $P(K)$ vs K plot approximately like the random learner, showing that such QNNs have no bias. This corresponds with

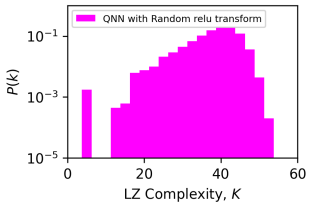
the results in Figure 6.1a. The QNNs with other encoding methods look similar to the DNN and flatten out the $P(K)$ probabilities.



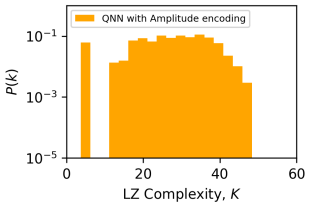
(a) Prior $P(K)$ for QNN with basis encoding



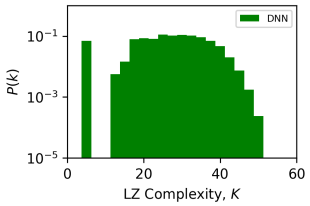
(b) Prior $P(K)$ for QNN with ZZ Feature Map



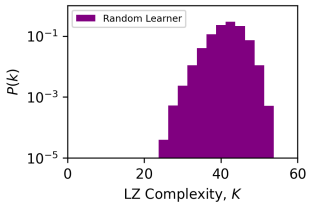
(c) Prior $P(K)$ for QNN with relu transform



(d) Prior $P(K)$ for QNN with amplitude encoding



(e) Prior $P(K)$ for classical DNN



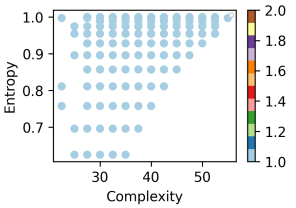
(f) Prior $P(K)$ for random learner

Figure 6.6: **Probability to obtain a function of Lempel-Ziv complexity K versus its complexity for five data qubits and 10^5 samples.** $P(f)$ versus Lempel-Ziv complexity, K for a QNN with (a) basis encoding (red), (b) ZZ feature map (blue), (c) random relu transform (pink), (d) amplitude encoding (orange), (e) a DNN (green), and (f) a random learner (purple). $P(K)$ is calculated by generating 10^5 samples of functions from the QNN by using random samples of parameters Θ over a uniform distribution. 10^7 samples were generated from the DNN and 10^5 of the samples were randomly selected to match the scale of the QNN.

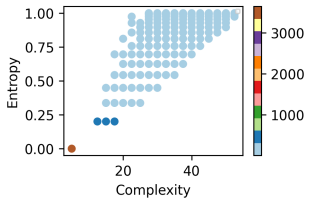
6.8.1.2 Entropy vs Complexity

To see more clearly the simplicity bias and the entropy bias for the different encoding methods, we plot the entropy vs complexity of the functions produced

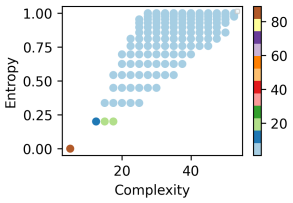
from the QNN in Figure 6.7.



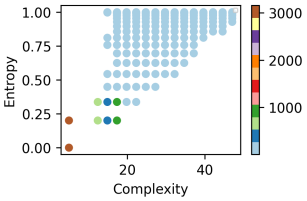
(a) Entropy vs complexity for QNN with basis encoding



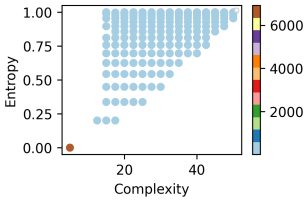
(b) Entropy vs complexity for QNN with ZZ feature map



(c) Entropy vs complexity for QNN with random relu transform



(d) Entropy vs complexity for QNN with amplitude encoding



(e) Entropy vs complexity for classical DNN

Figure 6.7: **Entropy of a boolean function (f) versus its complexity for five data qubits and 10^5 samples.** The colour of the data point corresponds to the maximum count of a function that has that complexity and entropy. Therefore, a data point with a colour that has a higher count indicates that the QNN has a stronger bias towards functions with that entropy and complexity. The boolean functions produced by the QNN were grouped and for each entropy-complexity pair, there were multiple functions, so the maximum count of a function is displayed.

The plots show an entropy or simplicity bias in several ways: first, we look at low-complexity, high-entropy functions. We would expect an encoding method with a simplicity bias to have a strong bias towards these types of functions and therefore produce many of them. On the other hand, we would expect an

encoding method with an entropy bias to not have a bias towards these types of functions and therefore not produce them. As $n = 5$, there are 4.3×10^9 possible functions. As we produce 10^5 samples, we would expect that the functions that are not produced by the QNN (and therefore have no data points on the plots) are functions that the QNN does not have a bias towards.

Figure 6.7d shows that the QNN with amplitude encoding produces functions with low-complexity and high-entropy. For example, at complexity at around 20, the data points produce a vertical line from entropy 0.25 up until 1. This pattern, consisting of a vertical line of data points at the lowest complexity across most values of the entropy, forming a right-angle trianglular shape, is similar to the DNN figure showed in Figure 6.7e. For the QNN with the ZZ feature map and the random relu transform (Figure 6.7b and Figure 6.7c respectively), however, we see that such QNNs are only able to produce functions with low-entropy when there is low-complexity. Data points for low-complexity, high-entropy functions are missing; this is because these QNNs have a bias towards low-entropy functions. Therefore, the right-angle triangular shape 'disappears' as the vertical line of data points at low complexity across most values of the entropy disappear.

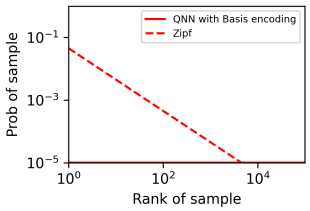
Furthermore, we can look at the colours of the data points. The data points with higher-count colours for the QNNs with the ZZ feature map and random relu transform are below entropy of 0.25 and complexity of 20 (the three dark blue data points and the brown data point). For the amplitude encoding, however, in addition to those data points, it also produces data points with higher-count colours for functions with entropy above 0.25 and complexity below 20 as well as an additional brown data point with entropy higher than the brown data point shared with the encoding methods. This further demonstrates the simplicity bias of amplitude encoding and the entropy bias of the other encoding methods. The figure shows that for basis encoding, there is neither entropy nor complexity bias as all of the data points remain at a similar count, therefore indicating no bias.

6.8.1.3 Probability of Boolean Function vs Rank

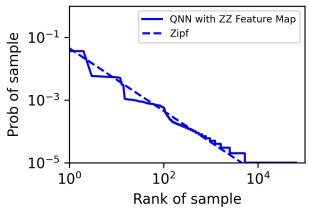
We plot the probability of a boolean function versus the rank of the sample in Figure 6.8. We compare the QNN data with the normalised Zipf law $P(r) = \frac{1}{ln(N_O)r}$, where r is the rank and $N_O = 2^{2^n}$, where n is the number of data qubits. Zipf's law tells us that the frequency of a sample is inversely

proportional to its rank. In Figure 6.8 we plot the data for $n = 5$ so $N_O = 2^{32}$.

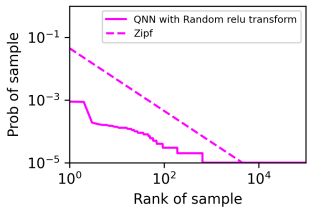
Figure 6.8a shows the rank plot for basis encoding - its probability versus rank is uniform at 10^{-s} where s is the number of samples. Figures 6.8b, 6.8d, and 6.8e fit Zipf's law.



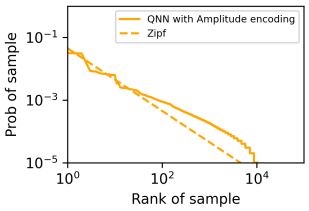
(a) $P(f)$ vs Rank for QNN with basis encoding



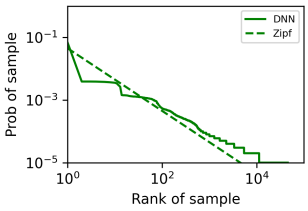
(b) $P(f)$ vs Rank for QNN with ZZ feature map



(c) $P(f)$ vs Rank for QNN with random relu transform



(d) $P(f)$ vs Rank for QNN with amplitude encoding



(e) $P(f)$ vs Rank for classical DNN

Figure 6.8: **Probability of a boolean function versus its rank for five data qubits and 10^5 samples.** $P(f)$ ranked by probability of individual functions, generated from 10^5 samples of parameters Θ over a uniform distribution. The dotted line is Zipf's law $P(f) = \frac{1}{(32ln2)\text{Rank}(f)}$. 10^7 samples were generated from the DNN and 10^5 of the samples were randomly selected to match the scale of the QNN.

6.8.2 Inductive Bias in Quantum Kernels

In this section, we explain quantum kernels and how they relate to QNNs. We show additional experiments that demonstrate that the inductive bias in QNNs corresponds to the inductive bias in quantum kernels.

6.8.2.1 The Kernel Matrix

A kernel function k maps the input data into a higher-dimensional space $k(\vec{x}_i, \vec{x}_j) = \langle f(\vec{x}_i), f(\vec{x}_j) \rangle$, where $\vec{x}_i, \vec{x}_j$ are n dimensional inputs, f is a map from an n-dimension space to an m-dimension space, and $\langle a, b \rangle$ denotes the inner product between a and b. A kernel matrix K can be constructed from all the input data $K_{ij} = k(\vec{x}_i, \vec{x}_j)$.

A quantum feature map $\phi(\vec{x})$ maps classical data, represented as a vector $\vec{x}$, to a quantum Hilbert space and the kernel matrix is $K_{ij} = \left| \langle \phi^\dagger(\vec{x}_j) | \phi(\vec{x}_i) \rangle \right|^2$.

To obtain the kernel matrix from a QNN, we encode the x_i data into the QNN and obtain the final statevector $|\phi\rangle$. We can then compute $\left| \langle \phi^\dagger(\vec{x}_j) | \phi(\vec{x}_i) \rangle \right|^2$ for all pairs of i, j. Further details on how to do this are in Section 3.5.

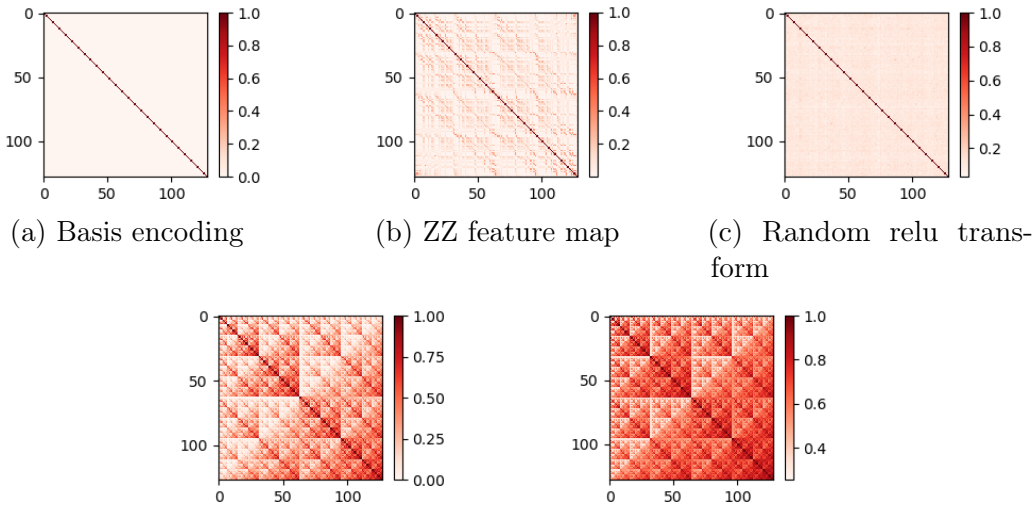
6.8.2.2 Kernel Matrix Plots

We plot the kernel matrices for different encoding methods in Figure 6.9. One can see in Figure 6.9a that for the basis encoding, there is no correlation between data points. The kernel matrix with basis encoding is equivalent to the Identity matrix as shown in Figure 3.10, which confirms that basis encoding produces no inductive bias. The ZZ feature map (Figure 6.9b) and random relu transform (Figure 6.9c) show some correlation, and the amplitude encoding (Figure 6.9d) shows stronger correlation. The DNN kernel (Figure 6.9e) is included for comparison.

6.8.2.3 Kernel Eigenvalues

The eigenvalues of the kernel matrices are plotted in Figure 6.10. Looking at the eigenvalues also provides insight into the type of bias.

The plot shows that for the kernel with basis encoding, its eigenvalues are flat, which is not the case for the eigenvalues for the other encoding methods. Kubler et al. provide a detailed analysis of the eigenvalues of kernel matrices and how it relates to inductive bias [37].



(a) Basis encoding (b) ZZ feature map (c) Random relu transform

(d) Amplitude encoding (e) Classical FCN (fully convolutional network)

Figure 6.9: **Kernel matrices for different encodings for seven qubits (n = 7) resulting in 128 × 128 dimension matrices.**

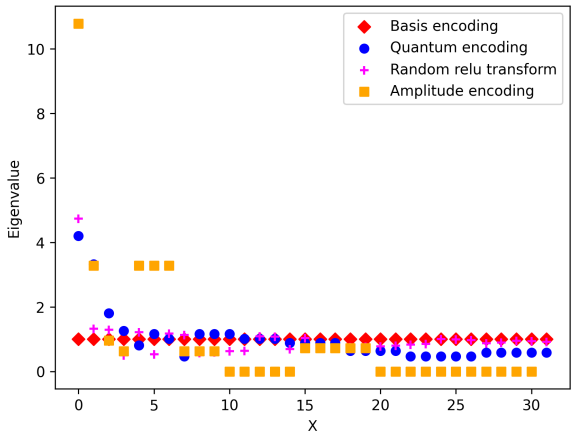
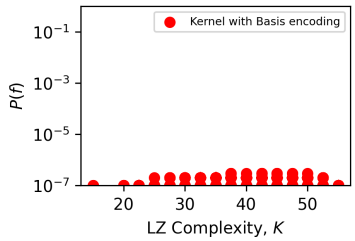


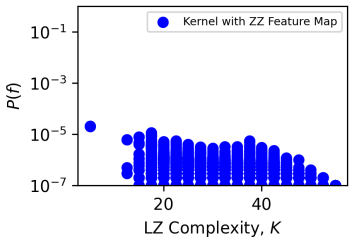
Figure 6.10: **Kernel eigenvalues for different encoding methods for five qubits (n = 5) where 'x' is the index of the eigenvalue.**

6.8.2.4 Probability of Boolean Function vs Complexity

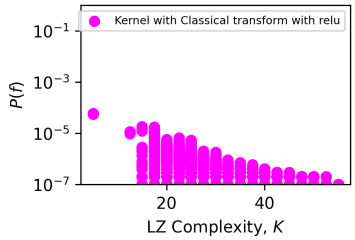
We plot the probability of the boolean function vs complexity as described in Section 6.4.3. The trends in Figure 6.11 are similar to those of the QNN shown in Section 6.4.3, demonstrating the alignment between the QNN and quantum kernel with regards to how the encoding method changes their inductive bias.



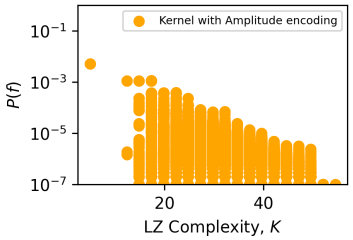
(a) Prior $P(f)$ vs complexity for the quantum kernel with basis encoding



(b) Prior $P(f)$ vs complexity for the quantum kernel with ZZ feature map



(c) Prior $P(f)$ vs complexity for the quantum kernel with random relu transform

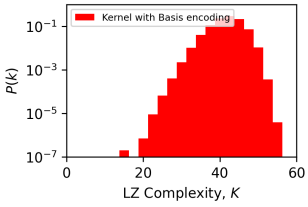


(d) Prior $P(f)$ vs complexity for the quantum kernel with amplitude encoding

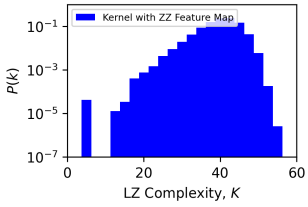
Figure 6.11: **Probability of a boolean function (f) versus its complexity for five data qubits and 10^7 samples.** $P(f)$ versus Lempel-Ziv complexity, K for a quantum kernel with (a) basis encoding (red), (b) ZZ feature map (blue), (c) random relu transform (pink), and (d) amplitude encoding (orange). $P(f)$ is calculated by generating 10^7 samples of functions from the quantum kernel by using multivariate normal in numpy.

6.8.2.5 Probability of Complexity vs Complexity

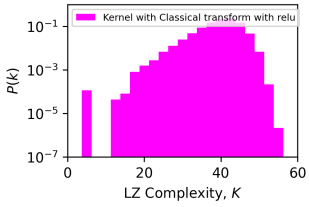
We also plot the probability of obtaining boolean functions with complexity K versus the complexity K in Figure 6.12 as described in Section 6.8.1.1. The trends in Figure 6.12 are also similar to those of the QNN shown in Section 6.8.1.1.



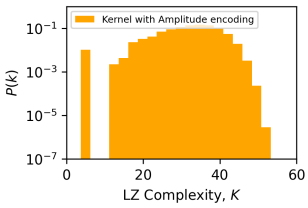
(a) Prior $P(K)$ for the quantum kernel with basis encoding



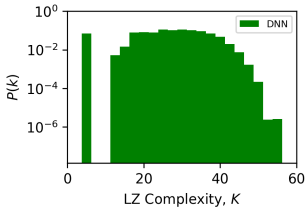
(b) Prior $P(K)$ for the quantum kernel with ZZ feature map



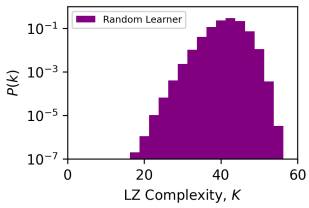
(c) Prior $P(K)$ for the quantum kernel with random relu transform



(d) Prior $P(K)$ for the quantum kernel with amplitude encoding



(e) Prior $P(K)$ for the classical DNN

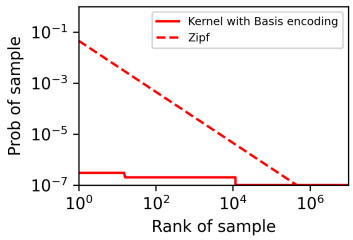


(f) Prior $P(K)$ for a random learner

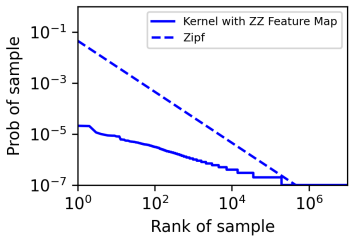
Figure 6.12: **Probability to obtain a function of Lempel-Ziv complexity K versus its complexity for five data qubits and 10^7 samples.** $P(f)$ versus Lempel-Ziv complexity, K for a quantum kernel with (a) basis encoding (red), (b) ZZ feature map (blue), (c) random relu transform (pink), (d) amplitude encoding (orange), (e) a DNN (green), and (f) a random learner (purple). $P(f)$ is calculated by generating 10^7 samples of functions from the quantum kernel by using multivariate normal in numpy.

6.8.2.6 Probability of Boolean Function vs Rank

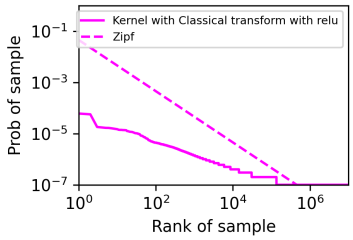
We plot the probability of a boolean function versus the rank of the sample in Figure 6.13. We compare the kernel data with the normalised Zipf law $P(r) = \frac{1}{ln(N_O)r}$, where r is the rank and $N_O = 2^{2^n}$, where n is the number of data qubits. Zipf's law tells us that the frequency of a sample is inversely proportional to its rank. In Figure 6.13 we plot the data for $n = 5$ so $N_O = 2^{32}$.



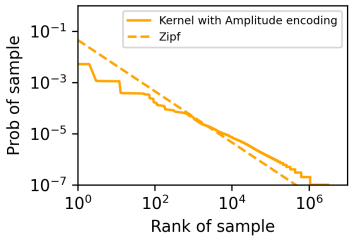
(a) $P(f)$ vs Rank for the quantum kernel with basis encoding



(b) $P(f)$ vs Rank for the quantum kernel with ZZ feature map



(c) $P(f)$ vs Rank for the quantum kernel with random relu transform



(d) $P(f)$ vs Rank for the quantum kernel with amplitude encoding

Figure 6.13: **Probability of a boolean function versus its rank for five data qubits and 10^7 samples.** $P(f)$ ranked by probability of individual functions, generated from 10^7 samples from the kernel. The dotted line is Zipf's law $P(f) = \frac{1}{(32ln2)\text{Rank}(f)}$.

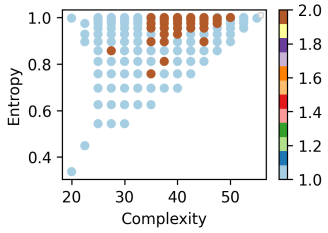
6.8.2.7 Entropy vs Complexity Plots

Another way to visualise whether the quantum kernels have a simplicity or entropy bias is to plot the entropy vs complexity and look at the count of each data point. Figure 6.14 shows the entropy vs complexity where the colour of the data point corresponds to the maximum count of a function that has that complexity and entropy.

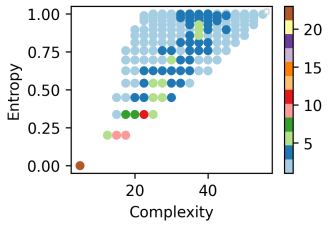
Figure 6.14a for basis encoding shows that there is neither entropy nor complexity bias as the blue and brown data points correspond to one and two counts of the boolean functions respectively. The difference between the counts is small and does not indicate any bias.

6.8.3 Training and Test Datasets for Boolean Data

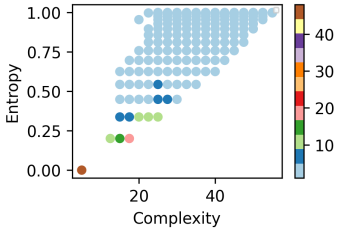
We describe the general process for generating the training and test datasets in Section 6.5.4. In this section, we provide an example. If the target function $t = 10100011$, then $t_0 = 1, t_1 = 0, t_2 = 1$, etc... We randomly select half of these indices to be the training set, for example $0, 2, 4, 5$ and the other half is the test set, which is $1, 3, 6, 7$.



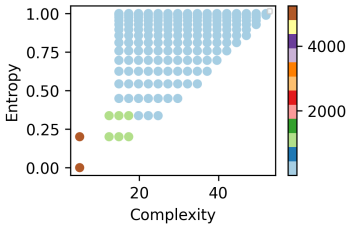
(a) Entropy vs complexity for the quantum kernel with basis encoding



(b) Entropy vs complexity for the quantum kernel with ZZ feature map



(c) Entropy vs complexity for the quantum kernel with random relu transform



(d) Entropy vs complexity for the quantum kernel with amplitude encoding

Figure 6.14: **Entropy of a boolean function (f) versus its complexity for five data qubits and 10^6 samples.** The colour of the data point corresponds to the maximum count of a function that has that complexity and entropy. The boolean functions produced by the quantum kernel were grouped and for each entropy-complexity pair, there were multiple functions, so the maximum count of a function is displayed.

Table 6.5 shows the Boolean data inputs and their indices for $n = 3$.

The training inputs then are the Boolean data inputs X at indices $0, 2, 4, 5$.

This corresponds to training inputs of $X_0 = (0, 0, 0), X_2 = (0, 1, 0), X_4 = (1, 0, 0), X_5 = (1, 0, 1)$. The test inputs are at indices $1, 3, 6, 7$ corresponding to

test inputs of $X_1 = (0, 0, 1), X_3 = (0, 1, 1), X_6 = (1, 1, 0), X_7 = (1, 1, 1)$.

The training labels are the values of the bits at the training indices of the target function. For example, for the target function $t = 10100011$, the training labels are $t_0 = 1, t_2 = 1, t_4 = 0, t_5 = 0$. Similarly, the test labels are the values of the bits at the test indices of the target function, so the test labels are $t_1 = 0, t_3 = 0, t_6 = 1, t_7 = 1$.

Input	**Index**
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Table 6.5: **Boolean input data for n = 3.**

6.8.4 Generalisation Error of QNNs

6.8.4.1 Target Functions

For training the QNNs to calculate the generalisation error, we use constructed and random target functions. A list of the target functions we use are shown

in Table 6.6. Table 6.7 includes further details about these target functions.

No.	Sym	Function	Description
1		00000000000000000000000000000000	all 0s
2		01001110100010110111100001100010	random
3		11111111111111111111111111111111	all 1s
4		00000000000000001000000000000000	one 1
5		11110111111111111111111111011111	two 0s
6		10000001110101011000101111010000	ZZ sample
7		01101001100101101001011001101001	parity
8		01010101010101010101010101010101	01 repeat
9		10101010101010101010101010101010	10 repeat
10		11111111111111110000000000000000	halfs 1 half 0s
11		00000000000000001111111111111111	half 0s half 1s
12		11101110111011101110110111011110	random length 4 function repeated 8 times
13		01011111010111110101111101011111	random length 8 function repeated 4 times

Table 6.6: **Target functions for five qubits**. The first column is the number of the target function for identifying it. The second column is the symbol used for the target function. The third column is the target function itself and the fourth column is a description of the target function.

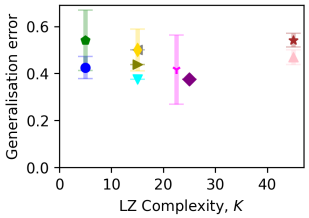
6.8.4.2 Generalisation Error vs Complexity, Entropy, and Probability of Target Function

In Figure 6.15, we chose constructed and random target functions (as shown in Table 6.6 and Table 6.7) and train the QNN with different encoding methods on

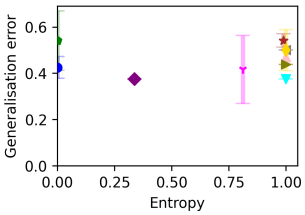
		Value		Category		QNN $P(f)$			
No.	**Sym**	**K**	**E**	**K**	**E**	**Basis**	**ZZ**	**Relu**	**Amp**
1		5	0	Low	Low	0	$3\cdot10^{-2}$	$8\cdot10^{-4}$	$3\cdot10^{-2}$
2		55	1	High	High	0	0	0	0
3		5	0	Low	Low	0	$3\cdot10^{-2}$	$8\cdot10^{-4}$	$3\cdot10^{-2}$
4		15	0.2	Low	Low	0	$5\cdot10^{-3}$	$9\cdot10^{-5}$	$3\cdot10^{-3}$
5		25	0.33	Med.	Low	0	$7\cdot10^{-4}$	$2\cdot10^{-5}$	0
6		45	0.99	High	High	0	0	0	0
7		45	1	High	High	0	0	0	0
8		15	1	Low	High	0	0	0	$2\cdot10^{-4}$
9		15	1	Low	High	0	0	0	$2\cdot10^{-4}$
10		15	1	Low	High	0	0	0	$1\cdot10^{-4}$
11		15	1	Low	High	0	0	0	$1\cdot10^{-4}$
12		17.5	0.81	Low	High	0	0	0	0
13		22.5	0.81	Low	High	0	0	0	$2\cdot10^{-4}$

Table 6.7: **Target functions for five qubits.** We show the LZ complexity (K) and the entropy (E) of the target function in the third and fourth columns respectively. We categorise the complexity and entropy in the fifth and sixth columns. For the following columns, we show the probability of obtaining the target function f from the QNN with the different encoding methods.

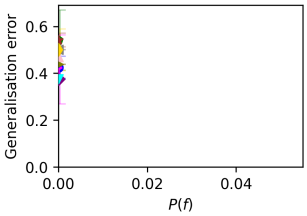
these target functions and calculate the generalisation error. We then plot the generalisation error versus the complexity of the target function, the entropy of the target function, and the probability of obtaining the target function when randomly sampling the QNN. Other details about how the QNNs were trained are in Section 6.4.4.



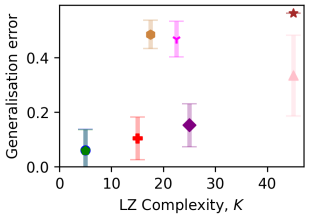
(a) Generalisation error vs complexity for QNN with basis encoding



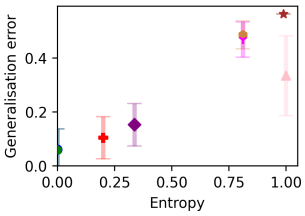
(b) Generalisation error vs entropy for QNN with basis encoding



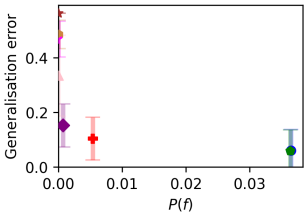
(c) Generalisation error vs probability for QNN with basis encoding



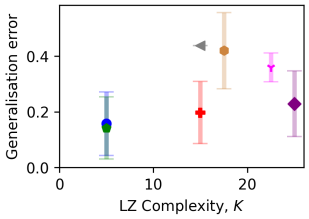
(d) Generalisation error vs complexity for QNN with ZZ feature map



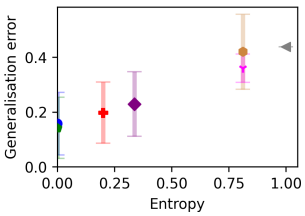
(e) Generalisation error vs entropy for QNN with ZZ feature map



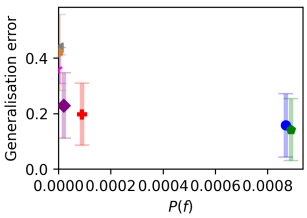
(f) Generalisation error vs probability for QNN with ZZ feature map



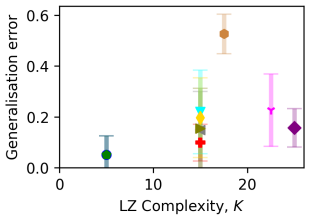
(g) Generalisation error vs complexity for QNN with random relu transform



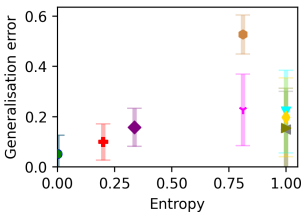
(h) Generalisation error vs entropy for QNN with random relu transform



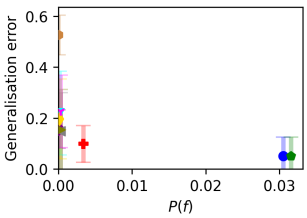
(i) Generalisation error vs probability for QNN with random relu transform



(j) Generalisation error vs complexity for QNN with amplitude encoding



(k) Generalisation error vs entropy for QNN with amplitude encoding



(l) Generalisation error vs probability for QNN with amplitude encoding

Figure 6.15: *(Caption on the next page.)*

Figure 6.15: **Generalisation error of a fully expressive QNN with different encoding methods for five data qubits. (a)** Generalisation error versus Lempel-Ziv complexity of the target function for a QNN with different encoding methods. The QNN is trained to zero training error on a training set S of size $m = 16$ and the generalisation error is calculated on the remaining 16 functions. The QNNs are trained by randomly sampling parameters 10^5 times and finding the generalisation error on the set of parameters that obtain zero training error. Error bars are one standard deviation. **(b)** Generalisation error versus entropy of the target function with the same specifications listed in (a). **(c)** Generalisation error vs probability of a target function for a QNN with different encoding methods. 10^5 samples of functions are generated from the QNN by using random samples of parameters. The probability of each function is calculated and its generalisation error is calculated.

These results show that the QNN with basis encoding has the equivalent performance of a random learner as across all target functions it has a 0.5 generalisation error, regardless of the complexity or entropy of the target function.

The QNN with the ZZ feature map generalises well on low-complexity low-entropy functions, such as the all 0s function. However, we can see two target functions that break the trend of low-complexity low-generalisation error. Those are the target functions 12 and 13 which have low complexity but high entropy. The QNN with the ZZ feature map generalises poorly on these, with around a 0.5 error.

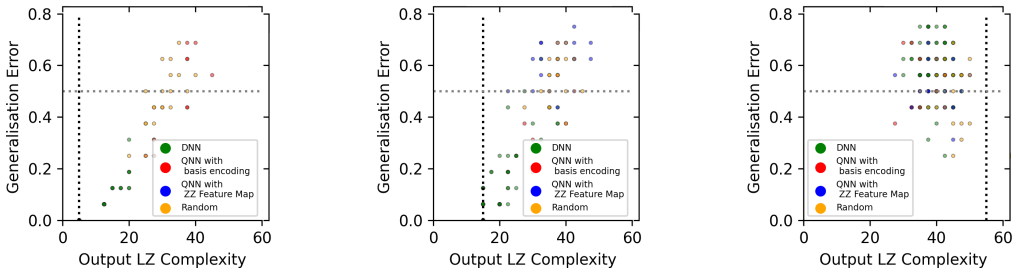
If we look at the entropy plot for the QNN with the ZZ feature map in Figure 6.15e, we see that this method has a bias towards low entropy functions.

This pattern is similar for the random relu transform, which is a random unitary matrix applied to the input data and then put through a ReLU layer. Target functions 8 and 12 have low complexity but high entropy and the random relu transform generalises poorly on these target functions. The entropy plot, Figure 6.15h, shows that this transform also has a bias towards low entropy functions.

The QNN with amplitude encoding shows a different story. For target functions with low complexity and high entropy, target functions 8 - 13, we see that the generalisation error is generally low, showing that amplitude encoding has a bias towards low complexity functions, similar to the DNN.

6.8.4.3 Generalisation Error vs Complexity of Output Function

Each figure in Figure 6.16 shows the generalisation error vs output LZ complexity for a specified target function.



(a) Target function LZ = 5

(b) Target function LZ = 15

(c) Target function LZ = 55

Figure 6.16: **Generalisation error versus output function LZ complexity scatterplots for n = 5 and for 20 random initialisations for three target functions.** The output LZ complexity is the complexity of the target function that the neural network produces after it has been trained to zero training error. A neural network that produces an output target function that is equivalent to the specified target function will have zero generalisation error and the same output LZ complexity as the specified target function. Each data point corresponds to one iteration. The dashed vertical line denotes the target function complexity.

Figure 6.16a shows the generalisation error vs output LZ complexity for a simple target function with Lempel-Ziv complexity $LZ = 5$. The plot shows that the DNN data points remain in the lower left of the plot as it produces target functions with low complexity and low generalisation error. The random learner produces random target functions (i.e. random boolean functions), and given that there are exponentially more complex functions than simple functions, the output target functions it produces are more likely to be complex and therefore have high generalisation error. The QNN with basis encoding is

similar to the random learner and also produces high complexity output target functions even when trained on a low complexity target function.

Figures 6.16b and 6.16c show the results for target functions with higher complexities. We see that the data points for the DNN shift towards the upper right of the graph as the DNN produces target functions with higher complexity and higher generalisation error. The random learner and QNN with basis encoding continue to produce high complexity output functions, which have high generalisation error.

6.8.5 Quantum Generalisation Performance on Parity Function

Target function 7 in Figure 6.17 is the parity function (whose value is one if and only if the input vector has an odd number of ones). The classical perceptron can not express the XOR function (the parity function on two inputs). In Figure 6.17, the QNN with the ZZ feature map has a lower generalisation error for the parity function compared to the DNN. The parity function has high complexity (LZ complexity $= 45$) and therefore we'd expect a high generalisation error for the DNN as the DNN has a bias towards simple functions. One might also

expect the QNN to have a high generalisation error on the parity function as it has a high complexity. We see the opposite however – the QNN with the ZZ feature map has a low generalisation error on the parity function.

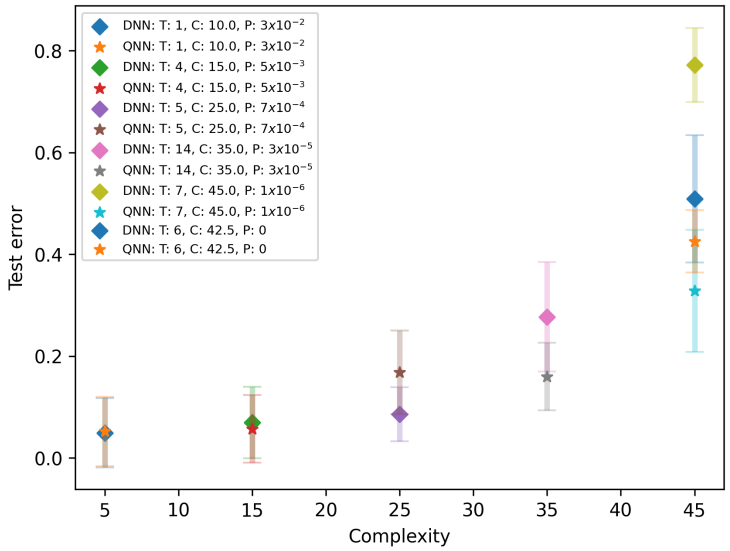


Figure 6.17: **Generalisation error versus Lempel-Ziv complexity of the QNN with ZZ Feature Map vs DNN for $n = 5$ for different target functions T training to zero error on a training set S of size $m = 16$, with the error calculated on the remaining $|T| = 16$ functions.** The target functions correspond to those in Table 6.6 and Table 6.7. An additional target function was added in Table 6.8. The networks were trained with the COBYLA optimiser on MSE loss for 20 random initialisations. Error bars are one standard deviation. The stars are the data points for the QNN with the ZZ feature map and the diamonds are the data points for the DNN. The legend indicates the LZ complexity (C) of the target function and the probability (P) of obtaining the target function in 10^6 samples.

This is because the QNN with the ZZ feature map has a stronger bias towards the parity function (it was engineered to do so) [127]. The QNN with the

No.	Function	Description
14	01000001000000101000000000100000	ZZ sample

Table 6.8: **Additional target function for five qubits.** This target function with LZ complexity = 35 is used in Figure 6.17.

ZZ feature map has a low generalisation error (around 0.25) for the parity function. This can also be seen in comparison with the other target functions with complexity 45, which have generalisation errors around 0.5.

It should be noted, however, that another DNN could be constructed to have a strong bias towards the parity function. Although the standard DNN has a poor generalisation performance on the parity function and the QNN has a strong generalisation performance on the parity function, which could seem like a quantum advantage, another type of DNN could be constructed to also have a strong generalisation performance on the parity function.

6.8.6 Training QNNs

In this section, we show the plots of the training process of the QNNs using an optimiser. It was difficult to train via this method because it was hard to obtain zero training error for certain encoding methods. Therefore, we trained

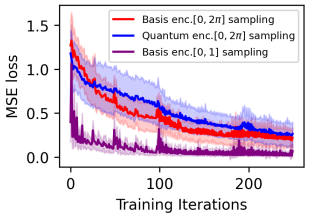
via random sampling to obtain the results in the main text of the paper (see Section 6.4.4). In this section, we show the results of the training process using optimisers.

We want to train the QNNs to learn a particular boolean function, which is the target function. Further details about the training method can be read in Section 6.5.4. We used Qiskit's EstimatorQNN to train the QNNs using the COBYLA and SPSA optimisers. The training score is obtained by running the training inputs through the QNN with the training parameters and obtaining the predicted y labels. The accuracy score s is calculated by the fraction of positions where the predicted label y_{predict} and the training label y_{train} match. The error is $1 - s$. We ensure that the QNN trains to zero training error (i.e. that the predicted label y_{predict} exactly matches the training label y_{train}).

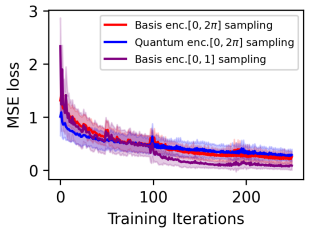
With zero training error, we can then compute the test error (also called the generalisation error). The test score g is obtained by running the test inputs through the QNN with the finalised set of parameters (that obtained zero training error) and obtaining the predicted y_{predict} labels. The test accuracy is the fraction of positions where the predicted label y_{predict} and the test label

y_{test} match. The generalisation error is $1 - g$.

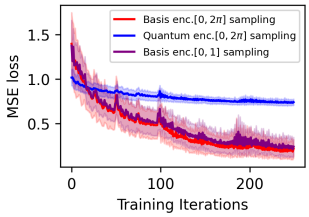
In Figures 6.18a - 6.18c we show the mean squared error (MSE) loss vs training iterations during the training process for different target functions.



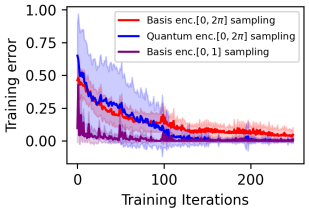
(a) MSE Loss vs iterations for target function 1 (which is all 0s) with complexity 5.0



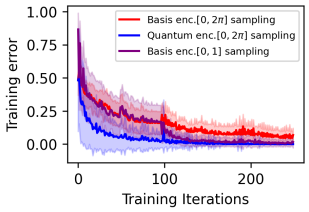
(b) MSE Loss vs iterations for target function 3 (which is all 1s) with complexity 5.0



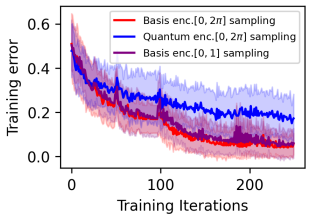
(c) MSE Loss vs iterations for target function 2 with complexity 55.0



(d) Training error vs iterations for target function 1 (which is all 0s) with complexity 5.0



(e) Training error vs iterations for target function 3 (which is all 1s) with complexity 5.0



(f) Training error vs iterations for target function 2 with complexity 55.0

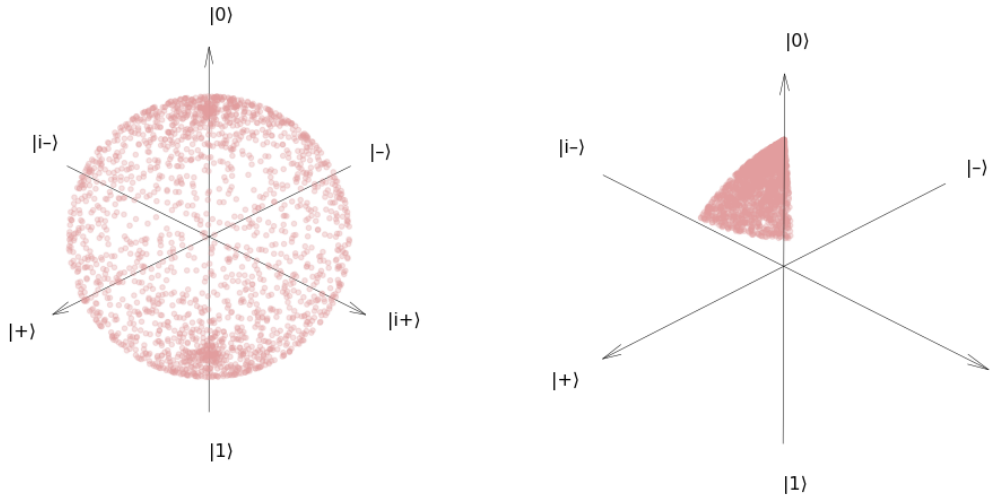
Figure 6.18: **MSE loss and training error vs iterations for different target functions and different encoding methods for n = 5.** The target function numbers correspond to those in Table 6.6 and Table 6.7.

We show the training process for two encoding methods: basis encoding and the ZZ feature map. In addition, we show the plot for basis encoding with

$[0, 1]$ sampling, which means that we sample the initial parameters for the QNN from the range $[0, 1]$ instead of the range $[0, 2\pi]$. As of the date of these experiments, IBM's Qiskit software includes, by default, the initial parameters for the QNN from the range $[0, 1]$. We overrided this default to sample the initial parameters for the QNN from the range $[0, 2\pi]$. We include both of these parameter initialisations to show how restricting the range of the parameters affects the training.

Figure 6.18a shows the MSE loss versus training iterations for target function $f_t = 0$ which is the all 0s function (e.g. for $n = 3$, $f_t = 00000000$), which has complexity $LZ = 5$. Note that the plots are for $n = 5$. We see that the basis encoding and ZZ feature map with $[0, 2\pi]$ sampling perform similarly well, whereas the basis encoding with $[0, 1]$ sampling achieves lower MSE loss quicker. One hypothesis for this is because restricting the initial parameter range restricts the states that the QNN can express. Sampling from $[0, 1]$ restricts the range of states for the U_3 gate to states where its classification will be zero, which is the function we are learning – therefore, the basis encoding with $[0, 1]$ sampling is initialised with the solution.

Figure 6.19a shows a Bloch sphere with the range of states that the U_3 gate can express with $[0, 2\pi]$ sampling and Figure 6.19b shows a Bloch sphere with the range of states that the U_3 gate can express with $[0, 1]$ sampling – as one can see, the states are restricted to the top hemisphere of the Bloch sphere and the top hemisphere is the zero classification (whereas the bottom hemisphere is the one classification). We believe it is important to be aware of this as IBM's Qiskit software starts with this default.



(a) Bloch sphere for U_3 gate applied to a qubit with parameters sampled from $[0, 2\pi]$

(b) Bloch sphere for U_3 gate applied to a qubit with parameters sampled from $[0, 1]$

Figure 6.19: **Bloch sphere for the U_3 gate applied to one qubit with parameters sampled from different distributions.** Figures generated with Pennylane software [78] and inspired by Reference [206].

Figure 6.18b shows the MSE loss versus training iterations for target function

$f_t = 6$ which is the all 1s function (e.g. for $n = 3$, $f_t = 11111111$), which has complexity $LZ = 5$. We see that all methods perform similarly. This highlights the difference that the $[0, 1]$ sampling made for training the QNN as when we train on the all 1s function, the QNN with $[0, 1]$ sampling does not perform any better than with $[0, 2\pi]$ sampling, whereas when we train on the all 0s function, the QNN with $[0, 1]$ sampling does perform better than $[0, 2\pi]$ sampling.

Figure 6.18c shows the MSE loss versus training iterations for target function $f_t = 5$ which has complexity $LZ = 55$. The ZZ feature map performs worse than the basis encoding. Intuitively, this is because we have skewed the data using the feature map, and therefore it has become harder to learn. Therefore, introducing certain inductive biases into a QNN via an encoding method can make it harder for the QNN to train and learn. The reason that we didn't see this for Figures 6.18a and 6.18c is because the QNN with the ZZ feature map has a bias towards the all 0s and all 1s function. So the QNN with the ZZ feature map will train well on functions it has a bias towards. When it encounters functions it has a weak bias towards, however, the training suffers.

Figures 6.18d - 6.18f show the same trends but for training error instead of

MSE loss.

6.8.7 Inductive Bias of Quantum Convolutional Neural Networks (QCNNS)

In this section we present some experiments on the inductive bias of QCNNs.

6.8.7.1 Definition of QCNNs

The Quantum Convolutional Neural Network (QCNN) was first proposed by Cong et al. [209]. The QCNN still has the general structure of the QNN explained in Section 3.4 and the variational structure is more specific in the QCNN. In particular, alternating convolutional and pooling layers are applied for the variational circuit, which reduce the dimension of the circuit until one qubit is left – the readout qubit q_r. In Figure 6.20 we show an example of a QCNN for $n = 4$. The first pooling layer reduces the dimension of the QCNN from four to two qubits by disregarding the first two qubits. The second pooling layer reduces the dimension of the QCNN from two qubits to one qubit q_r.

We wish to use a general two-qubit rotation gate in the layers. Vatan and

Williams [205] proposed an optimal quantum circuit to construct a general two-qubit rotation gate by using 15 parameters. This circuit is shown in Figure 6.21.

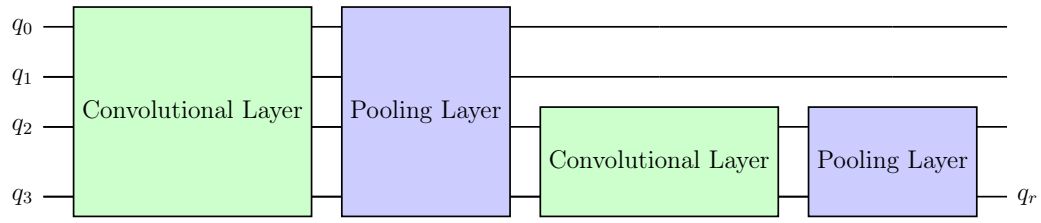


Figure 6.20: **General circuit diagram for a QCNN with four qubits.** Figure inspired by Figure 2 in Reference [210].

For the convolutional layer, the general two-qubit unitary is applied to neighbouring qubits: first to all even pairs, then to all odd pairs (the first and final qubits are also a pair). For the pooling layer, the general two-qubit unitary is applied to all pairs of qubits. Then one qubit from each pair is ignored for the rest of the QCNN.

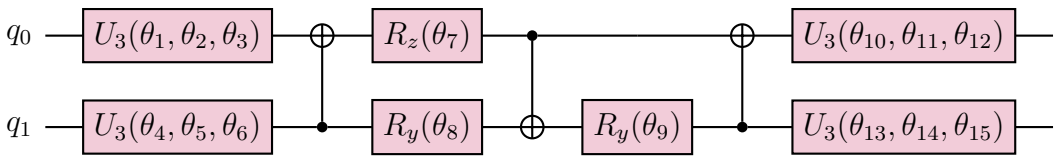


Figure 6.21: **Circuit for general two-qubit rotation.** Proposed in Reference [205].

6.8.7.2 Experiment Details

We use Qiskit's QCNN framework [210]. In their example, they use a restricted version of the general two-qubit rotation unitary (as shown in Figure 6.22) which means that it is not able to express all states in the Hilbert space. Our work has demonstrated that the expressivity of the QNN can affect its bias so we make plots for both the restricted two-qubit rotation unitary (Figures 6.23a - 6.23c) and the fully expressive two-qubit rotation unitary (Figures 6.23d - 6.23f).

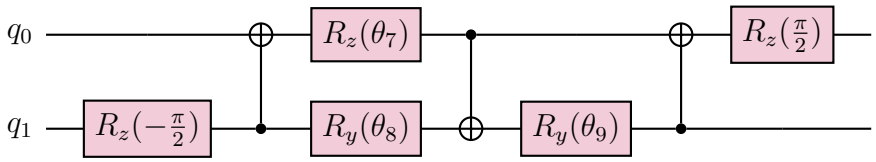


Figure 6.22: **Restricted two-qubit rotation.**

We sample from the QCNN in the same manner as the QNN and plot the probability of a boolean function versus its LZ complexity for three encoding methods: basis encoding, ZZ feature map, Z feature map. The Z feature map and ZZ feature map are derived from the Pauli Expansion Circuit, which is a data encoding circuit that transforms input data $\vec{x} \in \mathbb{R}^n$ as defined in Equation

6.9 from Reference [211]

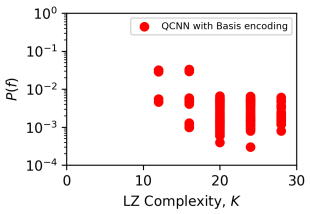
$$U_{\Phi(\vec{x})} = \exp\left(i \sum_{S \in \mathcal{I}} \phi_S(\vec{x}) \prod_{i \in S} P_i\right), \tag{6.9}$$

where $P_i \in \{I, X, Y, Z\}$ are the Pauli matrices, S is a set of qubit indices that describes the connections in the feature map, $\mathcal{I}$ is a set containing all these index sets, and the data-mapping $\phi_S(\vec{x})$ is defined in Equation 6.10

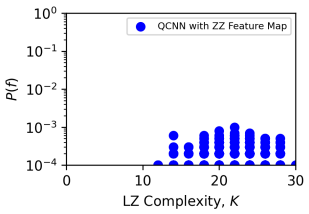
$$\phi_S(\vec{x}) = \begin{cases} x_i & \text{if } S = \{i\}, \\ \prod_{j \in S}(\pi - x_j) & \text{if } |S| > 1. \end{cases} \tag{6.10}$$

6.8.7.3 Probability of Boolean Function vs Complexity

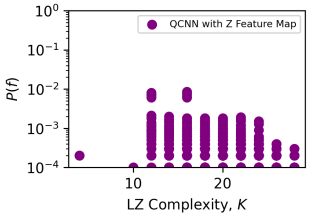
Figure 6.23 shows the probability of the boolean function vs the function's complexity as described in Section 6.4.3. Any bias seems to be further reduced in the fully expressive case, which aligns with our results that non-fully expressive QNNs can exhibit an artificial inductive bias.



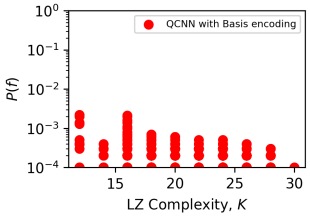
(a) $P(f)$ versus LZ complexity for non-fully expressive unitary in QCNN with basis encoding



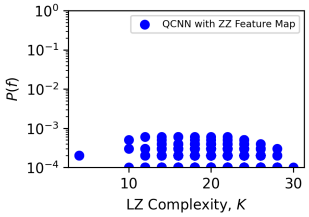
(b) $P(f)$ versus LZ complexity for non-fully expressive unitary in QCNN with ZZ Feature Map



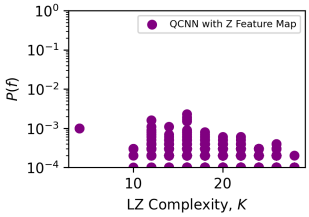
(c) $P(f)$ versus LZ complexity for non-fully expressive unitary in QCNN with Z Feature Map



(d) $P(f)$ versus LZ complexity for fully expressive unitary in QCNN with basis encoding



(e) $P(f)$ versus LZ complexity for fully expressive unitary in QCNN with ZZ Feature Map



(f) $P(f)$ versus LZ complexity for fully expressive unitary in QCNN with Z Feature Map

Figure 6.23: **The prior probability P(f) versus LZ complexity K for QCNNs with different encoding methods and expressivity for n = 4 and 10^4 samples.**

6.9 Review

As highlighted at the beginning of this chapter, Chapter 6, our aim in this chapter was to answer the question of whether QNNs have simplicity bias, which was motivated by the theoretical advances in classical deep learning and the unanswered questions in quantum machine learning.

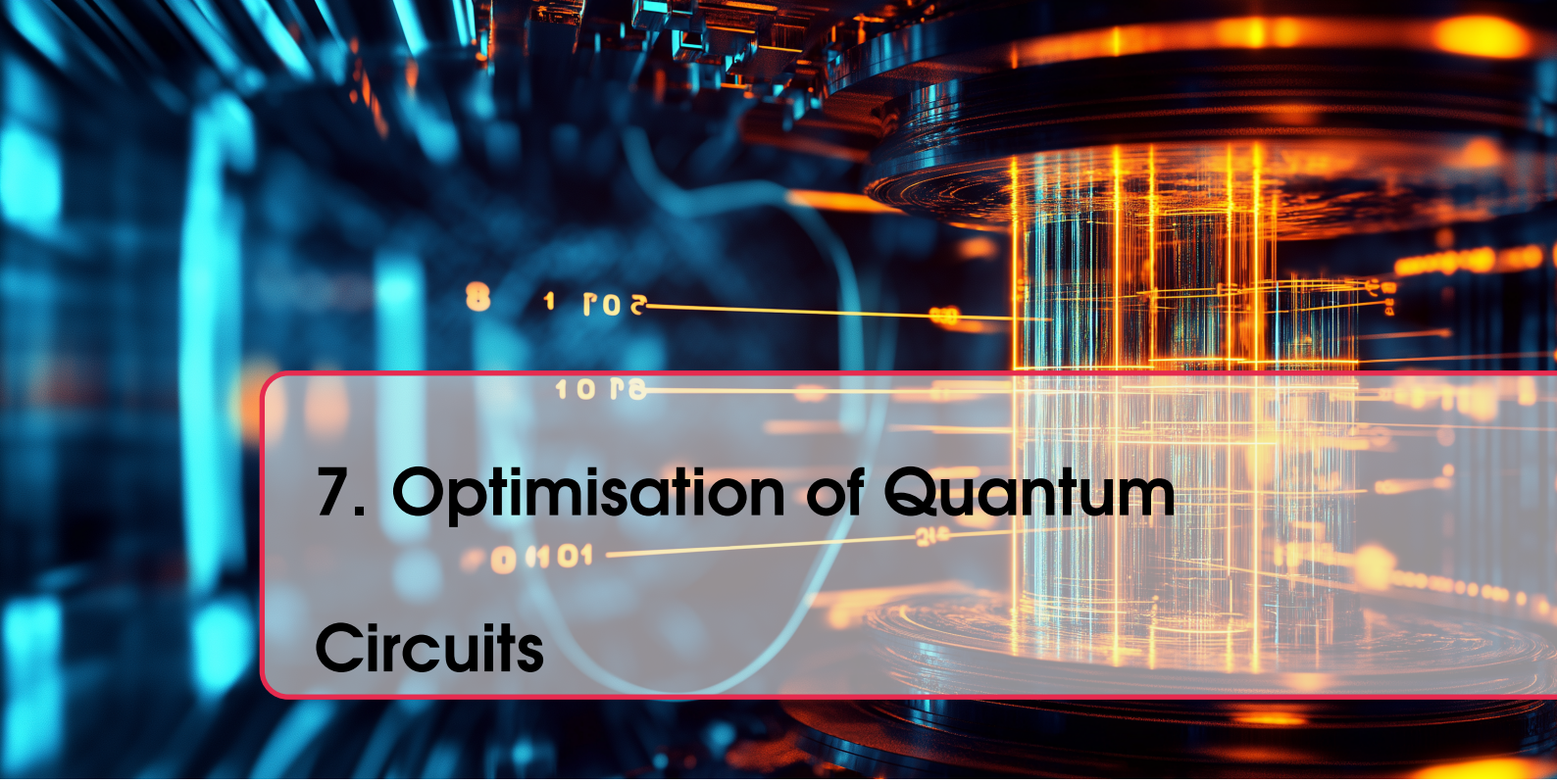
We answered this research question by constructing fully-expressive and non-fully expressive QNNs with different common encoding methods, including basis encoding, amplitude encoding, ZZ feature map, and random relu transform. We randomly sampled parameters for these QNNs and calculated the probability of obtaining particular boolean functions when given the boolean dataset. We measured the complexity of these boolean functions using two measures, LZ complexity and entropy, and plotted how the probability of boolean functions varied as the complexity of the boolean function varied. To distinguish encoding methods with a bias towards LZ complexity versus entropy, and to also understand how the encoding method affects the generalisation performance of the QNNs, we trained the QNNs on a subset of a target function and calculated its test error (i.e. generalisation error) by measuring its accuracy on another subset of the target function. We also studied the inductive bias of quantum kernels and quantum convolutional neural networks.

These results showed that basis encoding has no inductive bias which we also proved for random QNNs by induction, ZZ feature map and random relu transform have an inductive bias towards low-entropy functions, and amplitude encoding has a simplicity bias. We showed, however, that the expressivity

of the QNN was also affected by the encoding method, most notably that amplitude encoding limited the expressivity of the QNN, which we proved by contradiction. We also showed that intentionally restricting the expressivity of the QNN could introduce a bias, even if there is no bias in the fully-expressive version of the QNN.

Our results suggested a bias-expressivity tradeoff for the encoding method used. Given the inherent simplicity bias in deep neural networks and their high expressivity, our results suggest that QNNs can not have an advantage over deep neural networks as general-purpose learning tools on classical data. This chapter provides a better understanding of the performance of QNNs on classical data and can help motivate alternative frameworks for QNNs that could have both high expressivity and a good inductive bias.

In the next chapter, Chapter 7, we turn to how we can optimise the performance of QNNs, by creating a quantum compiler to optimise quantum circuits. At the end of Chapter 7, we review how the inductive bias and expressivity of QNNs explored in this chapter, Chapter 6, affect this quantum compiler and vice versa.



7. Optimisation of Quantum Circuits

7.1 Preview 385

7.2 Summary 386

7.3 Introduction 387

7.4 Results 397

7.5 Discussion 414

7.6 Methods 419

7.7 Review 444

7.1 Preview

In the previous chapter, Chapter 6, we studied the inductive bias and expressivity of quantum neural networks, which are quantum circuits with parameters that can be varied via some feedback loop. In this chapter, Chapter 7, we are interested in optimising quantum circuits themselves, so that quantum circuits, and hence quantum neural networks, can be run with quicker execution times on classical simulators and hardware and with reduced noise. Reducing the runtime and errors in quantum neural networks could make them more likely to be practically used for applications, as runtime and errors are some of the bottlenecks for practical quantum applications. This chapter presents original research on a new technique for quantum compiling that is based on the trends we discussed in the *Quantum Computing* chapter (Chapter 2). Particularly, we discussed that quantum tools, such as quantum compilers, were developed as the quantum hardware and software matured. The early quantum compilers were designed for specific quantum hardware, but hardware-agnostic compilers later emerged; however, these compilers focused on mapping logical quantum circuits to hardware and only used a simple set of rules for optimising the quantum circuits. The motivation for this chapter is to contribute to a further

development of quantum compilers, in which complex logical quantum circuit optimisations are performed, without regards to mapping to the hardware; as a result we can go beyond a simple set of rules and find more complex circuit equivalences to use for optimisation. Our aim in this chapter is to design a quantum compiler using automatic generation of circuit identities, inspired by classical compilation techniques, so that more optimal circuit optimisations can be found. Our aim is also to design a technique to use smaller-scale complex circuit identities for larger-scale quantum circuits. This chapter aims to contribute to the *optimisation* of quantum algorithms, the second conceptual layer of the framework presented in Chapter 1, Section 1.2. This chapter is based on the paper titled 'Quanto: optimizing quantum circuits with automatic generation of circuit identities' [40].

7.2 Summary

Existing quantum compilers focus on mapping a logical quantum circuit to a quantum device and its native quantum gates. Only simple circuit identities are used to optimise the quantum circuit during the compilation process. This approach misses more complex circuit identities, which could be used to optimise

the quantum circuit further. We propose Quanto, the first quantum optimiser that automatically generates circuit identities. Quanto takes as input a gate set and generates provably correct circuit identities for the gate set. Quanto's automatic generation of circuit identities includes single-qubit and two-qubit gates, which leads to a new database of circuit identities, some of which are novel to the best of our knowledge. In addition to the generation of new circuit identities, Quanto's optimiser applies such circuit identities to quantum circuits and finds optimised quantum circuits that have not been discovered by other quantum compilers, including IBM Qiskit and Cambridge Quantum Computing Tket. Quanto uses a customisable cost function to optimise the quantum circuit, meaning that Quanto can take into account a variety of quantum circuit metrics for optimisation unlike previous compilers. Quanto's database of circuit identities could be applied to improve existing quantum compilers and Quanto can be used to generate identity databases for new gate sets.

7.3 Introduction

Quantum programs are expressed as circuits of primitive gates; the specific *gate set* available varies depending on the quantum computer. Compilers are

used to optimise the circuit in an effort to reduce the execution time, the noise that naturally arises in quantum computations [14], or both.

A quantum algorithm is implemented on a quantum computer with a quantum circuit, which consists of a sequence of quantum gates. A quantum gate applies a transformation to a qubit and can be represented as a matrix. Generally, the larger the circuit depth (length of the longest path from the input to the output), the noisier the computation, leading to more errors in the results. Because of the possibility of incorrect results and the inherent probabilistic nature of quantum computation, quantum programs must be executed many times to obtain an accurate picture of their results. Therefore, reducing the depth of a quantum circuit not only makes a quantum program faster, but also reduces the potential for noise and reduces the number of program evaluations required. Thus, finding equivalent but smaller depth circuits is a core problem in the implementation of quantum programs.

Quantum hardware imposes many constraints on quantum compilers, including the layout and connectivity of the qubits. Such aspects are addressed by the

compilers built by the hardware vendor (such as IBM or Google). Less attention has been given to optimising the logical circuit itself – finding circuit identities for a given gate set.

Circuit identities have been found manually [212] and automatically for single-qubit gates [213], but as far as we know there is no system today that automatically finds identities for full gate sets and no compiler that optimises quantum circuits based on such circuit identities.

We present Quanto, the first quantum circuit optimiser that *automatically generates* circuit identities. Figure 7.1 shows an overview of Quanto. In a first, offline phase, Quanto generates a database of circuit identities. The Quanto compiler takes a quantum circuit and identity database as inputs and uses a cost-based search algorithm to replace the quantum circuit with an optimised quantum circuit.

Figure 7.2 shows an example of a quantum circuit with two qubits and a depth of five. The circuit contains the two-qubit gate CX and the single-qubit gates H and Z. The single-qubit gates are represented by 2×2 unitary complex

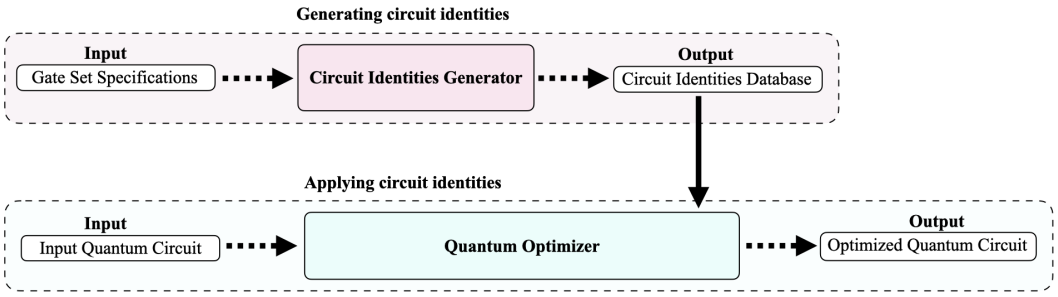


Figure 7.1: **Overview of Quanto.**

matrices and the two-qubit gates are represented by 4×4 unitary complex matrices. This quantum circuit is a *logical quantum circuit*, which is a quantum circuit defined in terms of unitary matrices and is agnostic to the quantum hardware.

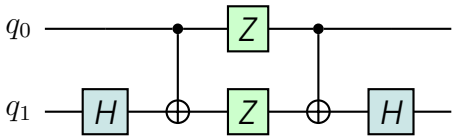


Figure 7.2: **Example of a quantum circuit that can be optimised with Quanto.**

If we wish to run this logical quantum circuit on the quantum hardware, we would have to compile the quantum circuit into the hardware's *native quantum gates*. These are the set of quantum gates that have been implemented on particular quantum hardware. For example, IBM's native quantum gate set

was {U1,U2,U3,CX} [1], where

$$U1(\lambda) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\lambda} \end{pmatrix},$$
$$U2(\psi, \lambda) = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -e^{i\lambda} \\ e^{i\psi} & e^{i(\lambda+\psi)} \end{pmatrix},$$
$$U3(\theta, \psi, \lambda) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -e^{i\lambda}\sin\left(\frac{\theta}{2}\right) \\ e^{i\psi}\sin\left(\frac{\theta}{2}\right) & e^{i(\lambda+\psi)}\cos\left(\frac{\theta}{2}\right) \end{pmatrix},$$
$$CX = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

In this example, the *source gates* are the gates present in the logical quantum circuit {X, H, CX} and the *target gates* are the native quantum gates {U1, U2, U3, CX}. Therefore, in order to run this logical quantum circuit on the quantum hardware, we must find circuit identities for the source gates in terms of the target gates. In this case $Z = U1(\pi)$ and $H = U2(\pi, 0)$.

Another step for running a logical quantum circuit on the hardware is to map the qubit and gate connections to those of the physical constraints of the hardware. Most compilers focus on these two aspects: (1) mapping source

[1] IBM later changed their native gate set to {CX, ID, RZ, SX, X}

gates to target gates and (2) mapping the quantum circuit to the hardware constraints. Then, generally, a set of simple manual rules are used to optimise the logical quantum circuit itself. For example, VOQC (Verified Optimizer for Quantum Circuits) [214] includes five rules for optimisations – (0) not propagation, (1) Hadamard reduction, (2) single-qubit gate cancellation, (3) two-qubit gate cancellation, and (4) rotation merging. The VOQC optimiser applies these rules in the following order in order to optimise a circuit: 0, 1, 3, 2, 3, 1, 2, 4, 3, 2.

Applying a limited number of optimisation rules limits the opportunity to find other optimisations. Instead of defining a set of optimisation rules, our compiler is the first to automatically generate circuit identities. Quanto finds an optimised logical quantum circuit that is equivalent to the input logical quantum circuit.

The first phase in the compiler is generating *circuit identities*. Circuit identities are quantum circuits that are mathematically equivalent. A quantum circuit can be represented as a matrix and two quantum circuits are equivalent when their matrices are equal (for a certain floating point precision).

Quanto's circuit identity generator enumerates all possible quantum circuits over a given set of gates up to a fixed depth and number of qubits. In the example in Figure 7.2, the circuit identity generator would find all possible circuits with two qubits and a depth of five containing the gates {X, H, CX} (or another gate set specified). To efficiently find all such circuit identities, Quanto constructs a hash table where quantum circuits are stored based on the hash of their *unitary matrix*. The operation of a quantum gate can be represented using a unitary complex matrix. The operation of a quantum circuit can be represented by a unitary matrix obtained from the matrices of individual gates using matrix multiplication and tensor products. In the example in Figure 7.2, the unitary matrix is $U = (I \otimes H) \cdot CX \cdot (Z \otimes Z) \cdot CX \cdot (I \otimes H)$.

The second phase is the optimiser. When given an input quantum circuit, our optimiser searches for quantum circuits equivalent to the input quantum circuit that minimise the cost function. In this paper we use circuit depth as the cost; that is, a circuit will be replaced by the lowest-depth equivalent circuit known. In general, however, the cost function can be chosen to optimise any quantifiable criteria of a circuit. For example, a cost function could include the total number of gates, total number of specific gates, such as T gates or

two-qubit gates, and/or gate errors. Because we generate all possible circuits, Quanto returns the optimal quantum circuit with respect to the cost function. In the example in Figure 7.2, our optimiser returns an equivalent quantum circuit shown in Figure 7.3. Quanto has reduced the depth of the circuit from five to one. This circuit has the minimum depth possible for that given gate set.

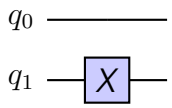


Figure 7.3: **The optimal circuit of the quantum circuit in Figure 7.2 for the given set.**

The circuit identities are obtained by simulating the quantum circuits. One problem, however, is that simulating a quantum circuit is expensive if the quantum circuit has a large number of qubits and circuit depth. The promise of quantum computing is dependent on the fact that simulating quantum circuits is classically inefficient. Therefore, the larger the quantum circuit becomes (i.e. more qubits and greater depth), and the more gates that are added, the more configurations there are to generate, which eventually can become infeasible. In order to solve this issue in our optimiser, we introduce *tiles*. A *tile* is a subset

of a quantum circuit with a specified height and width. A tile is the source quantum sub-circuit that is matched to equivalent quantum circuits. In the example in Figure 7.2, if our tile width is equal to three and our tile height is equal to two, then the tiles in Figure 7.4 would be generated.

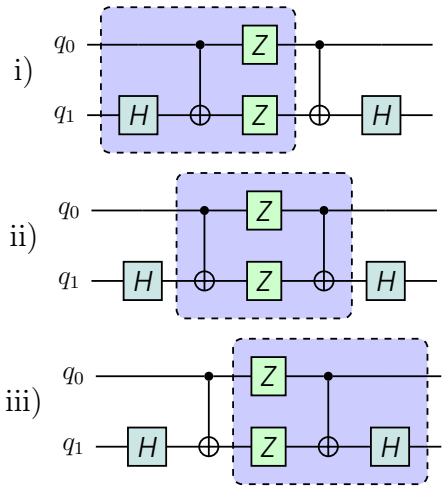


Figure 7.4: **Splitting the circuit into tiles of width 3 and height 2.**

Quanto searches for quantum circuits equivalent to the tiles in the circuit identities database. A cost-based search algorithm is used to apply the substitution, which is a quantum circuit in the circuit identities database that replaces a tile in the quantum circuit. A substitution is the target quantum circuit that defines an equivalent new quantum circuit to replace the matched quantum circuit (i.e. the tile). Figure 7.5 shows how the circuit gets optimised when it

is split into tiles.

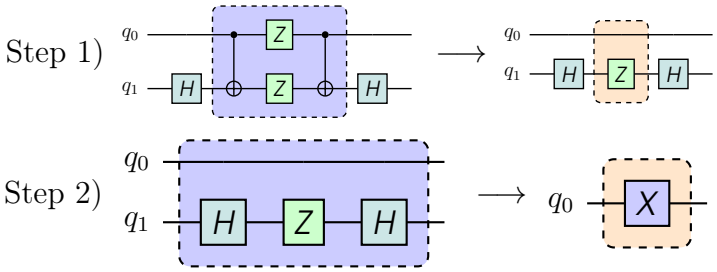


Figure 7.5: **Steps to optimise the circuit when the circuit is split into tiles.** The boxes shaded in blue are the tiles. The boxes shaded in orange are the substitutions.

Our main contribution is presenting Quanto, which is the first quantum optimiser to automatically generate quantum circuit identities. Our automatic generator finds identities used by existing compilers, and in addition finds novel identities (to the best of our knowledge). Quanto can generate novel identities for any given gate set, which is especially useful for quantum hardware with native gate sets that are not standard (for example, the two-qubit iSwap gate is less common than the two-qubit CX gate). In our evaluation, we demonstrate that the IBM Qiskit compiler [45] can not find these novel identities. Quanto also includes a tile-based method to optimise a large quantum circuit using the generated identities. We demonstrate that Quanto's optimiser finds equivalent circuits with smaller depth compared to IBM's Qiskit and Cambridge Quantum

Computing's Tket [13] compilers.

7.4 Results

7.4.1 Generating Existing Circuit Identities

Quanto can automatically discover identities that are known from previous work [212, 215]. We ran the compilers (Quanto, IBM's Qiskit, and Cambridge Quantum Computing's Tket) on a set of quantum circuits with known optimisations. The results show that Quanto is able to find the known optimisations for all of the input quantum circuits, whereas other compilers fail to find some of the identities. The other compilers could find simple identities, such as a circuit with the same gate in a row, but they failed to find more complicated identities.

These results are shown in Figures 7.6, 7.7, 7.8, 7.9, 7.10. References [212, 215] give a list of equivalence rules, which lead to equivalent quantum circuits (i.e. circuit identities). The circuits that are shown in our results are chosen from References [212, 215] because these are known circuit identities.

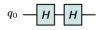
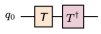
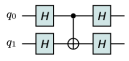
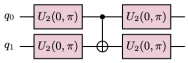
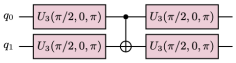
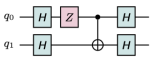
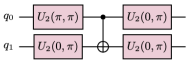
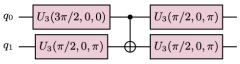
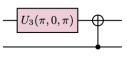
Circuit	Uncompiled circuit	IBM compiled circuit	Tket compiled circuit	Quanto compiled circuit	Quanto-IBM compiled circuit
A					
Depth	2	0	0	0	0
B					
Depth	2	0	0	0	0
C					
Depth	3	3	3	1	1
D					
Depth	4	3	3	2	2

Figure 7.6: **Comparing Quanto to other compilers for optimising quantum circuits with known optimisations.** Some of these circuits are chosen from Reference [212]. IBM and Tket backend is *ibmq essex*.

The second columns of the tables in the Figures show the input (uncompiled) circuit and the other columns show the compiled circuit after running on the compiler specified in that column. The last column in Figure 7.6 and the penultimate column in Figure 7.9 show the circuits that are generated when the uncompiled circuits are run through the Quanto compiler and then the IBM compiler. The reason we show these columns is because our optimiser is at the logical level and does not map the qubits to the hardware, a task we

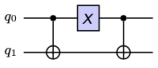
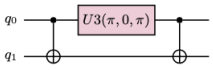
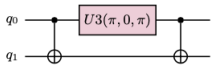
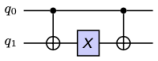
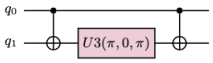
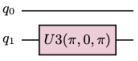
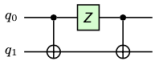
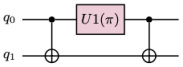
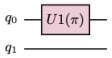
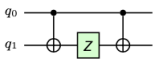
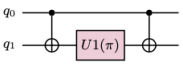
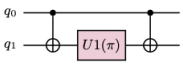
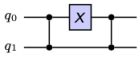
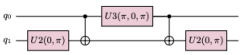
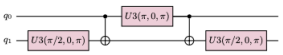
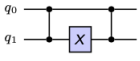
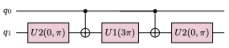
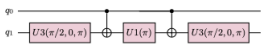
Circuit	Uncompiled circuit	IBM compiled circuit	Tket compiled circuit	Quanto compiled circuit
A				
Depth	3	3	3	1
B				
Depth	3	3	1	1
C				
Depth	3	3	1	1
D				
Depth	3	3	3	1
E				
Depth	3	5	5	1
F				
Depth	3	5	5	1

Figure 7.7: **Comparing Quanto to other compilers for optimising quantum circuits with known optimisations.** IBM and Tket backend is *Aerbackend*.

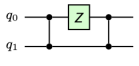
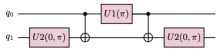
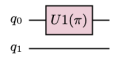
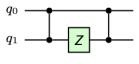
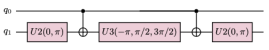
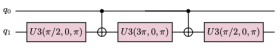
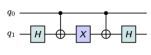
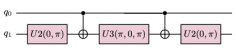
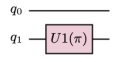
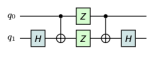
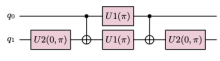
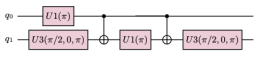
Circuit	Uncompiled circuit	IBM compiled circuit	Tket compiled circuit	Quanto compiled circuit
A				
Depth	3	3	1	1
B				
Depth	3	5	5	1
C				
Depth	5	5	1	1
D				
Depth	5	5	5	1

Figure 7.8: **Comparing Quanto to other compilers for optimising quantum circuits with known optimisations.** IBM and Tket backend is *Aerbackend*.

leave to the IBM compiler. We show that for the circuits in Figure 7.6 and some of the circuits in Figure 7.9 the circuits are unchanged by the mapping to the hardware. In particular, it is interesting to look at Circuit A and Circuit C in Figure 7.9. One might say that the IBM compiler circuit increased the depth of the uncompiled circuit from depth three to six in Circuit A and from depth three to seven in Circuit C because it needs to map to the hardware,

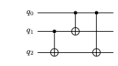
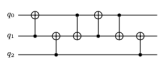
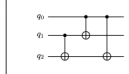
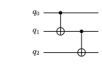
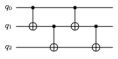
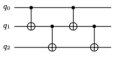
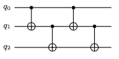
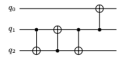
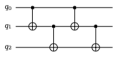
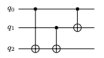
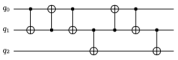
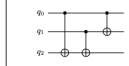
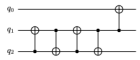
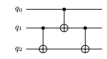
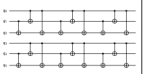
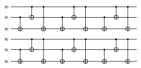
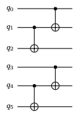
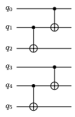
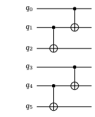
Circuit	Uncompiled circuit	IBM compiled circuit	Tket compiled circuit	Quanto compiled circuit	Quanto-IBM compiled circuit	Quanto compiled circuit (with constraint)
A						
Depth	3	6	3	2	2	2
B						
Depth	4	4	4	1	4	4
C						
Depth	3	7	3	2	2	2
D						
Depth	3	3	3	2	5	3
E		Circuit too large for displaying image				
Depth	9	31	9	2	2	2

Figure 7.9: **Comparing Quanto to other compilers for optimising quantum circuits with known optimisations.** Some of these circuits are chosen from Reference [212]. IBM backend is *ibmq lima* and Tket backend is *Aerbackend*.

but when we run our compiled circuit (which reduced the depth of the circuit to two in both circuits), the IBM compiler left the optimisation unchanged. These results demonstrate that optimising at the logical level is a useful task that can be done before mapping to the specific quantum hardware.

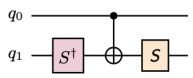
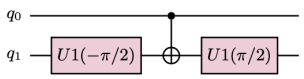
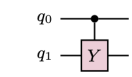
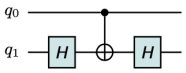
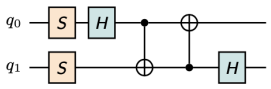
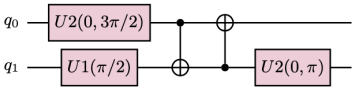
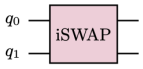
Circuit	Uncompiled circuit	IBM compiled circuit	Quanto compiled circuit
A			
Depth	3	3	1
B			
Depth	3	1	1
C			
Depth	5	4	1

Figure 7.10: **Comparing Quanto to other compilers for optimising quantum circuits for unique gate sets.** Some of these circuits are chosen from Reference [215].

The Quanto compiled circuits of Circuits B and D in Figure 7.9 are changed when they are mapped to the hardware. In particular, two-qubit gates that are not applied to neighbouring qubits are changed. In order to address this, we implemented a two-qubit gate neighbouring constraint in Quanto that can be turned on or off. If this constraint is on, it only returns optimised circuits that include two-qubit gates on neighbouring qubits. This is a simple hardware

constraint. Quanto could be improved by implementing more specific hardware constraints. The last column in Figure 7.9 shows the compiled circuits when the constraint is on.

7.4.1.1 Example of Known Circuit Identities

Known circuit identities are presented in the quantum computation literature, including References [212, 215], which focus on circuit identities for specific quantum gates (e.g. the Hadamard gate), and Reference [216], which presents circuit identities for any unitary gate or gates with a specific form (e.g. a gate which is composed of a sequence of rotation and other gates). Our results present known circuit identities for specific quantum gates. In this section, we want to show that some of these known circuit identities with specific gates are also equivalent to the generic circuit identities presented in Reference [216]. One of the circuit identities presented in Reference [216] is shown in Figure 7.11.

This two-qubit circuit identity is $CV = (I \otimes A) \cdot CX \cdot (I \otimes B)$, where CV is a controlled-V gate. A and B are unitary if and only if V is of the form

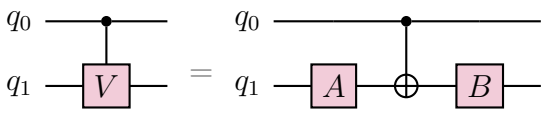


Figure 7.11: **Known circuit identity from Reference [216] where A and B are unitary if and only if V is of the form $\mathbf{V} = \mathbf{R_z}(\alpha) \cdot \mathbf{R_y}(\theta) \cdot \mathbf{R_z}(\alpha) \cdot \mathbf{X}$.**

$$V = R_z(\alpha) \cdot R_y(\theta) \cdot R_z(\alpha) \cdot X = \begin{pmatrix} \sin(\theta/2) & e^{i\alpha}\cos(\theta/2) \\ e^{-i\alpha}\cos(\theta/2) & -\sin(\theta/2) \end{pmatrix} \text{ and where}$$

$$R_y(\theta) = \begin{pmatrix} \cos(\theta/2) & \sin(\theta/2) \\ -\sin(\theta/2) & \cos(\theta/2) \end{pmatrix},$$

$$R_z(\alpha) = \begin{pmatrix} e^{i\alpha/2} & 0 \\ 0 & e^{-i\alpha/2} \end{pmatrix}.$$

Quanto is able to find this known circuit identity as shown in circuits A and B in Figure 7.10. For circuit A, the circuit identity is $CY = (I \otimes S^\dagger) \cdot CX \cdot (I \otimes S)$. Therefore, this is equivalent to the circuit identity from Reference [216] as shown in Figure 7.11 because $V = Y, A = S^\dagger, B = S$ where Y is the pauli-Y gate and has the form of V as

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} = R_z(\pi/2) \cdot R_y(2\pi) \cdot R_z(\pi/2) \cdot X.$$

For circuit B, the circuit identity is $CZ = (I \otimes H) \cdot CX \cdot (I \otimes H)$. Therefore, this is equivalent to the circuit identity from Reference [216] as shown in Figure 7.11 because $V = Z, A = H, B = H$, where Z is the pauli-Z gate and has the form of V as

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = R_z(0) \cdot R_y(\pi) \cdot Rz(0) \cdot X.$$

7.4.2 Generating Novel Circuit Identities

Quanto can also generate new identities, especially when given custom gates, as it is more likely that identities for custom gates have not been previously discovered. Existing compilers have a fixed repertoire of identities using common gates. The advantage of Quanto is that it can find identities for any custom gate, leading to optimisations that may be missed by other compilers. In Figure 7.12 we show some of these novel identities (these identities were found from our identity database and we are not aware of any previous reports of these identities). We searched through our identity database to find circuit identities we believe have not been identified based on our review of the literature. We compare our novel identity optimisations to the IBM Qiskit compiler, as it allows us to specify basis gates for compiling. IBM's compiler is unable to find

these circuit identities.

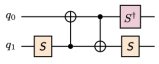
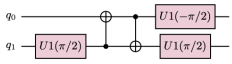
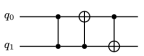
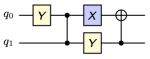
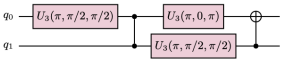
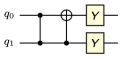
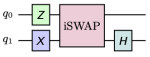
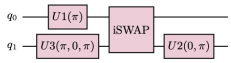
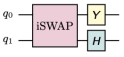
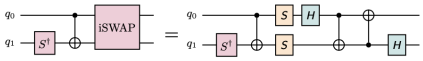
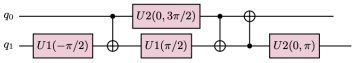
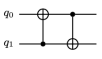
Circuit	Uncompiled circuit	IBM compiled circuit	Quanto compiled circuit
A			
Depth	4	4	3
B			
Depth	4	4	3
C			
Depth	3	3	2
D			
Depth	7	6	2

Figure 7.12: **Comparing Quanto to other compilers for optimising quantum circuits with unknown optimisations.**

7.4.3 Optimising Quantum Circuits

Other than generating existing and novel identities, Quanto can also optimise a quantum circuit. For quantum circuits where we are able to produce tiles with the same dimension as the quantum circuit, our optimiser will always find the most optimal equivalent quantum circuit (given a particular gate set). Therefore our optimiser will perform the same or better than other optimisers; there would be no instances where other optimisers outperform.

For quantum circuits where we can only produce tiles with dimensions smaller than the quantum circuit, our optimiser is not guaranteed to find the most optimal equivalent quantum circuit. Therefore, there may be some instances where the other optimisers outperform our optimiser.

7.4.4 Benefits of Optimisation

Our results show that Quanto can reduce the depth of quantum circuits. Two notable benefits of reducing the depth include (1) faster quantum program execution times, both for simulations and hardware and (2) noise reduction in the quantum program. In this section, we show why reducing the depth leads to these benefits and we show a practical example of the benefits on a quantum circuit.

7.4.4.1 Execution Times

Each quantum gate takes a certain duration to execute both on quantum simulators and hardware. A quantum simulator executes a quantum gate by performing matrix multiplication. A quantum gate, which we denote as U, operating on n qubits has dimension $2^n \times 2^n$. The quantum circuit is a sequence

of these unitary gates, so that the unitary matrix of the quantum circuit is equivalent to $U = U_{d-1} \cdot U_{d-2} \cdots U_0$, where d is the number of layers in the circuit (i.e. the depth).

Therefore, the number of $2^n \times 2^n$ matrix multiplications in a quantum circuit simulation is equivalent to d (there are other methods for simulating quantum circuits that may be more efficient - for example by taking advantage of Clifford gates as described in Reference [217], but we are using the default method for our analysis). If our quantum circuit optimisation algorithm reduces the depth of a quantum circuit by k, the number of $2^n \times 2^n$ matrix multiplications in a quantum circuit simulation is now equivalent to $d - k$. If each matrix multiplication takes time t_s, the execution time of the quantum program has been reduced from $d \cdot t_s$ to $(d - k) \cdot t_s$. Figure 7.13 shows an example of this reduction in execution time in practice.

For quantum hardware, each quantum gate also has an execution time t_h, which depends on the specific properties of the hardware. For example, the ECR gate, which is a two-qubit gate, on the IBM Kyoto device has a gate time of 660 nanoseconds [218]. Reducing the depth of the input quantum circuit also

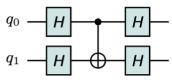
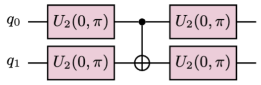
Circuit	Number of qubits	Uncompiled circuit	IBM compiled circuit	Quanto compiled circuit
A	2			
Depth	-	3	3	1
Compilation Time (seconds)	-	-	0.3017	0.0140
Execution Time (seconds)	-	-	0.0120	0.0054

Figure 7.13: **Comparing the compilation and execution time of the circuit compiled by Quanto versus by IBM.**

reduces the execution time of the quantum program from $d \cdot t_h$ to $(d - k) \cdot t_h$.

7.4.4.2 Noise Reduction

Another benefit of reducing the depth is reducing the noise for the quantum program. For our analysis, we use a simple noise model to show the effect of our optimiser on noise. We also show an example of this reduction in noise in practice in Figure 7.14.

Each quantum gate implemented on quantum hardware has an error rate e. The error rate is different depending on the quantum gate, but we can take

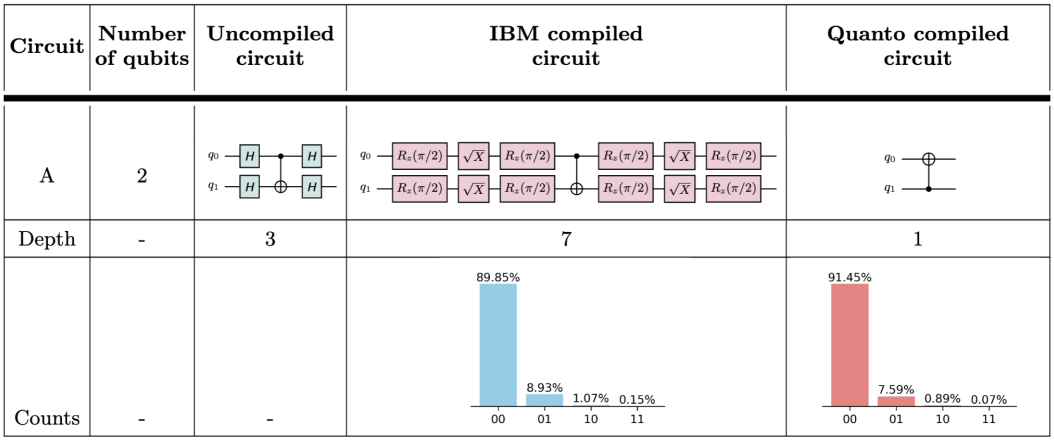


Figure 7.14: **Comparing the noise of the circuit compiled by Quanto versus compiled by IBM, using a noisy simulation of the IBM Fake-Vigo device.**

the average and state that the average gate error is $\tilde{e}$. If a quantum program consists of x gates, then the probability of no error in our quantum program is $P_s = (1 - \tilde{e})^x$. If an optimiser reduces the number of gates by k, then the probability of no error in the quantum program becomes $P_s = (1 - \tilde{e})^{(x-k)}$ [219].

We can use this simple noise model to estimate the error rates for a quantum program on a particular quantum device. In this example, we are going to use the quantum program in Figure 7.14. There are four Hadamard gates and one CX gate, which the IBM compiler compiles to eight R_z gates, four $\sqrt{X}$ gates,

and one CX gate. At a particular snapshot in time, the gate error rates for the IBM Vigo device are shown in Table 7.1.

Gate	Qubit(s)	Error rate
CX	[0,1]	$1.2012 \cdot 10^{-2}$
$\sqrt{X}$	0	$4.1352 \cdot 10^{-4}$
$\sqrt{X}$	1	$5.0203 \cdot 10^{-4}$
R_z	0	0
R_z	1	0

Table 7.1: **Gate error rates for the IBM Vigo device.**

Using the simple noise model and the error rates in Table 7.1, we can calculate that the probability of no error for the circuit compiled by IBM's compiler is $P_{s_{\text{IBM}}} = (1 - e_0^{R_z})^4 \cdot (1 - e_1^{R_z})^4 \cdot (1 - e_0^{\sqrt{X}})^2 \cdot (1 - e_1^{\sqrt{X}})^2 \cdot (1 - e_{0,1}^{CX}) = (1 - 4.1352 \cdot 10^{-4})^2 \cdot (1 - 5.0203 \cdot 10^{-4}) \cdot (1 - 1.2012 \cdot 10^{-2}) = 98.668\%$ where e_q^g is the error rate for the gate g on the qubit(s) q.

Quanto, however, compiles this circuit to just one CX gate. We can estimate that the probability of no error is $P_{s_{\text{Quanto}}} = (1 - e_{0,1}^{CX}) = (1 - 1.2012 \cdot 10^{-2}) = 98.799\%$. Using this simple noise model, Quanto has reduced the error by 0.131%. This simple noise model, however, underestimates the true amount of

noise and in practice Quanto can provide a greater noise reduction.

We can perform a noisy simulation to see the noise reduction for this circuit in practice. To perform noisy simulations of quantum circuits, we can use the IBM Qiskit Aer device noise model [220], which generates a simplified noise model for a real device, by using the calibration information for a device, taking into account the gate error, time to execute the gate, relaxation times of the qubits, and readout errors.

To obtain the probability of error in the noisy simulation, we execute the circuit 10^6 times and calculate the probability of obtaining the correct output.

Using our simulation on the IBM FakeVigo device (a noisy simulation of the real IBM Vigo device), as shown in Figure 7.14, the IBM-compiled circuit has $P_{s_{\text{IBM}}} = 89.85\%$ and $P_{s_{\text{Quanto}}} = 91.45\%$. In this case, Quanto has reduced the error by 1.6%.

The noise in the experiment is greater than our simple noise model because our simple noise model only takes into account the gate error, whereas there are

other sources of noise in practice such as the decoherence time of the qubits – the longer the execution of the quantum program (which depends on the depth), the more the qubits decohere and the more noise in the quantum program.

The magnitude of the noise reduction depends on the properties of the device. For example, the IBM London device has a different set of basis gates and therefore the IBM compiler compiles the circuit to a different set of gates (as shown in Figure 7.13). This compiled circuit has a shorter depth compared to the compiled circuit for the IBM Vigo device and therefore we can expect the noise reduction to be less significant for the Quanto-compiled circuit. An experiment on this device demonstrates $P_{s_{\text{IBM}}} = 96.19\%$ and $P_{s_{\text{Quanto}}} = 96.59\%$ leading to a noise reduction of 0.4%.

7.5 Discussion

7.5.1 Related Work

7.5.1.1 Automatically Generating Identities

To the best of our knowledge, previous work on automatically discovering quantum circuit identities is limited to single-qubit gates [213]. In addition, these automatically generated identities have not been applied to optimising quantum programs.

7.5.1.2 Superoptimisation

Superoptimisation is a compiler technique that was originally designed to find the optimal machine code for a sequence of instructions [221] [222] [223]. Our approach is inspired by TASO [39]. TASO applies superoptimisation to deep learning by optimising tensor computation graphs. For a given set of tensor operators (e.g. addition, matrix convolution) TASO automatically finds identities and uses these as substitutions to optimise computation graphs of deep learning applications. There are some differences: in the TASO paper, they generate a graph and then go through the list of substitutions and try to

see whether the substitution is a sub-graph in the original graph. They have a limited number of substitutions, in the thousands, so they can do that fairly efficiently. In our case, however, we have can have hundreds of thousands or millions of substitutions and the time complexity of that technique would grow large. Instead we split our circuit into sub-circuits (the tiles) and then look for the hash of that tile in the hash table, which is a constant lookup. Therefore, our technique is more efficient for our case and only depends on the number of tiles in the circuit, which grows modestly.

7.5.1.3 Quantum Optimisation

There are other compilers [45, 13, 214], but their focus is mapping the logical circuit to the quantum device. Therefore, they do not pay much attention to the potential logical transformations. This can be shown in our results – other optimisers fail to recognise certain circuit identities. These optimisers have manual rules for logical substitutions and therefore are not finding the full range of optimisation.

7.5.2 Limitations and Future Work

7.5.2.1 Scalability

One limitation of Quanto is the scalability of generating a large number of circuit identities, which limits the size (depth and number of qubits) of the circuit identities. This issue could be mitigated by applying pruning techniques during the generation of substitutions to remove redundant substitutions or by using heuristics to produce substitutions.

7.5.2.2 Missed Substitutions and Larger Circuits

Another limitation is missing out on potential substitutions, due to invalid tiles in the quantum circuit as shown in Figure 7.21. In addition, substitutions between distant pairs of qubits could be missed if the tiles only consider its nearest neighbour qubits. This could be mitigated by reordering the qubits (as we describe in the *Methods* section) at the cost of increasing the time complexity of the algorithm. Due to these limitations, Quanto's performance on larger quantum circuits is not as effective. We believe that further work to reduce these limitations could improve the effectiveness of Quanto when applied to larger quantum circuits.

7.5.2.3 Parametrised Gates

To use gates with parameters in our optimiser, we can set the parameter to a small set of specific values when generating identities. Since many parameters are used for qubit rotations, and certain angles of rotation are very common in quantum algorithms, this approach gives some reasonable coverage of potential optimisations involving parameters. However, our optimiser would be improved if it included symbolic identities, where substitutions for parametrised gates are generated regardless of the value of the parameter.

7.5.2.4 Noise Effects

It could also be interesting to do an in-depth analysis of the effects on noise in quantum hardware after using the methods presented here, thereby providing further insight into how these methods could be used as an error-mitigation technique.

7.5.2.5 Customised cost functions

One strength of our work is that the cost function can easily be changed. In the implementation in Python, one can define a Python function that is given a

quantum circuit as an input and returns a value as an output; this value is the cost of the quantum circuit and is used as the cost for evaluating substitutions in the optimiser. Therefore, a customised cost function could be supplied to the optimiser, which would use the customised cost for optimisation. Future work could explore the effectiveness of other cost functions. This compiler could be used as a more explicit error mitigiation technique by using quantum gate errors in the cost function. Quantum hardware providers sometimes provide the gate errors for their device as shown in Table 7.1, allowing one to define a customised cost function using the quantum gate errors for a particular quantum device. From the individual gate errors, one could calculate the 'error' of the quantum circuit. The quantum circuit with the smallest quantum circuit error would then be used as a substitution in the optimisation.

7.5.3 Conclusion

Quanto is the first quantum optimiser that automatically generates general circuit identities. Quanto finds novel optimisations and finds better optimisations for certain quantum circuits, compared to other compilers. This work contributes to reducing the circuit depth of quantum circuits, which could lead

to less noise and more accurate execution of quantum algorithms.

7.6 Methods

7.6.1 Circuit Identity Generator

This section describes the Quanto circuit identity generator, which automatically generates circuit identities given a gate set, as well as the number of qubits and depth bounds. This generation algorithm finds all circuit identities given the number of qubits, circuit depth, and gate set.

7.6.1.1 Generation Algorithm

Table 7.2 shows the inputs to the generation algorithm.

Step 1: Represent the quantum circuit as a grid. We represent a quantum circuit as a grid, where the number of rows m is the number of layers in the quantum circuit, also known as the circuit depth. The number of columns n is the number of qubits in the quantum circuit. If there is no gate present at a particular row for a particular column, we place the Identity gate

Symbol	Definition	Description
n	number of qubits	maximum number of qubits of the circuit identities generated
d	depth of the circuit	the maximum depth of the circuit identities generated
dp	decimal point accuracy	the decimal points of the floating-point numbers in the matrices of the gates
gs	gate set	a list of gates to use in the circuit identities, including the identity gate

Table 7.2: **Inputs of the generation algorithm.**

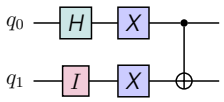
in that position. Figure 7.15 shows this quantum circuit representation. In the notation, CXC denotes applying the CX gate to the *control* qubit and CXT denotes applying the CX gate to the *target* qubit.

Step 2: For a single layer (depth = 1), generate all the possible circuits of quantum circuits using the gate set.

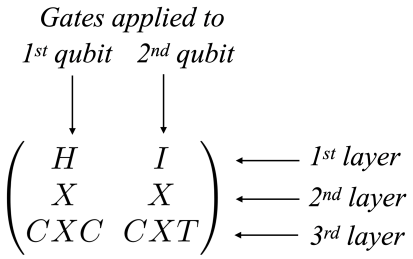
a) Single-qubit gates only

If there are only single-qubit gates in the gate set, then each gate is applied to

(a)



(b)



(c)

$$[[H, I], [X, X], [CXC, CXT]]$$

Figure 7.15: **(a) Quantum circuit which can be represented as (b) a matrix, which can be represented computationally as (c).**

a single qubit. If n is the number of qubits and g is the number of single-qubit gates, then there are g^n possible circuits.

Example: Gate Set $= [I,\ X,\ H]$ and $n = 2$, where

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, H = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}.$$

Table 7.3 shows all the possible circuits that would be generated for the gate set $[I, X, H]$. In the case of the possible circuit $[H, X]$ the H gate is applied to

the first qubit and the X gate is applied to the second qubit as shown in Table 7.3.

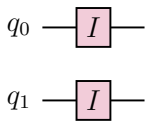
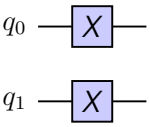
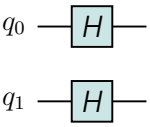
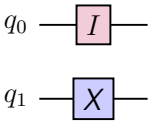
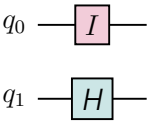
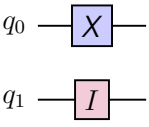
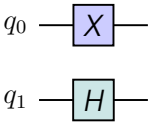
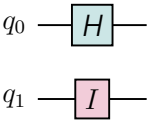
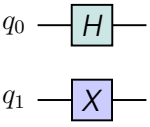
		
$[I,\ I]$	$[X,\ X]$	$[H,\ H]$
		
$[I,\ X]$	$[I,\ H]$	$[X,\ I]$
		
$[X,\ H]$	$[H,\ I]$	$[H,\ X]$

Table 7.3: **Possible circuits for single-qubit gates given gate set $[\mathbf{I}, \mathbf{X}, \mathbf{H}]$.**

b) Two-qubit and single-qubit gates

If there are two-qubit gates, then we split the two-qubit gate into the first and second qubits it operates on.

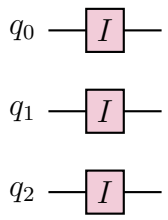
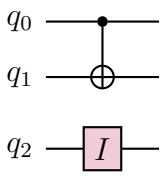
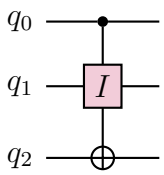
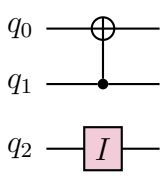
Example: Gate Set $= [I,\ CX]$ and $n = 3$, where

$$CX = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

Table 7.4 shows all the possible circuits that would be generated for the gate set $[I, CX]$. In the case of the possible circuit $[CXC, CXT, I]$ the CX gate is applied to the first and second qubit where the first qubit is the *control* qubit and the second qubit is the *target* qubit. The I gate is applied to the third qubit.

Verifying the validity of the two-qubit gates

CXC and CXT must both be present in the possible circuits; it is not possible to have one present without the other. For example, $[CXC, I, I]$ would be an invalid configuration. Quanto verifies that CXC and CXT are present in each possible circuit.

			
$[I,\ I,\ I]$	$[CXC,\ CXT,\ I]$	$[CXC,\ I,\ CXT]$	$[CXT,\ CXC,\ I]$

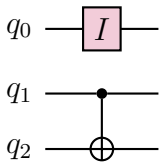
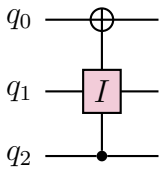
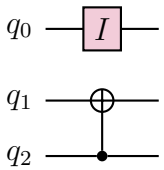
		
$[I,\ CXC,\ CXT]$	$[CXT,\ I,\ CXC]$	$[I,\ CXT,\ CXC]$

Table 7.4: **Possible circuits for two-qubit gates given gate set $[\mathbf{I}, \mathbf{CX}]$.**

Step 3: For multiple layers (depth > 1), generate all the possible circuits of the quantum circuits using the gate set. For an additional layer in the circuit, we add all the possible circuits for a single layer onto the previous layer.

Example: For the gate set presented in Table 7.3, we take each possible circuit and append it to the end of each possible circuit. Table 7.5 illustrates appending the possible circuit $[X, X]$ to each possible circuit. This is repeated

for $[H, H], [I, X], etc...$

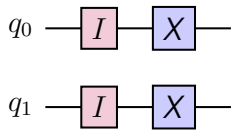
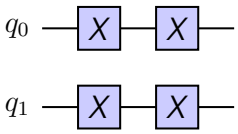
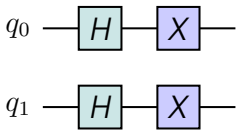
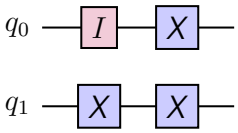
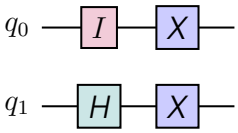
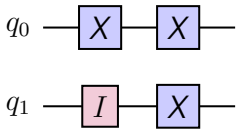
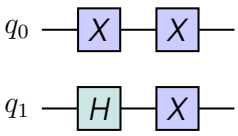
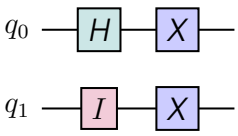
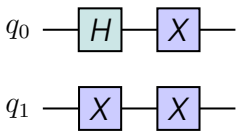
		
$[I, I], [X, X]$	$[X, X], [X, X]$	$[H, H], [X, X]$
		
$[I, X], [X, X]$	$[I, H], [X, X]$	$[X, I], [X, X]$
		
$[X, H], [X, X]$	$[H, I], [X, X]$	$[H, X], [X, X]$

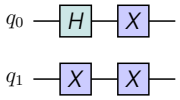
Table 7.5: **Possible circuits for multiple layers.**

Step 4: Calculate the unitary matrices and generate their fingerprints.

For each possible circuit, we find the *unitary matrix* of the possible circuit. We adopt an idea from compiler superoptimisation [223] and compute a *fingerprint* for each possible circuit, which is a hash of the unitary matrix. Figure 7.16 shows a possible circuit and how to calculate the unitary matrix. The matrices

in part (c) correspond to the gates in part (b) and (a) of Figure 7.16.

(a)



(b)

$$(H \otimes X) \times (X \otimes X)$$

(c)

$$\left(\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} \otimes \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\right) \times \left(\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix} \otimes \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\right) = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 0 & 1 & 0\\ 0 & 1 & 0 & 1\\ -1 & 0 & 1 & 0\\ 0 & -1 & 0 & 1\end{pmatrix}$$

Figure 7.16: **(a) A possible circuit generated from a gate set. (b) The gate representation of the circuit. (c) The matrix calculation of the circuit to obtain the unitary matrix.**

Step 5: Create hash table of fingerprints. The hash table of fingerprints includes the quantum circuit as the key and the fingerprint of the quantum circuit as the value. This allows us to look up a quantum circuit's fingerprint in constant time during the optimiser stage, without having to recalculate the unitary matrix and the fingerprint of the quantum circuit. Figure 7.17 shows the sequence of steps to generate a hash table of fingerprints for each possible circuit. Table 7.6 is an example hash table with one entry which shows the quantum circuit representation as the key and its corresponding fingerprint as the value.

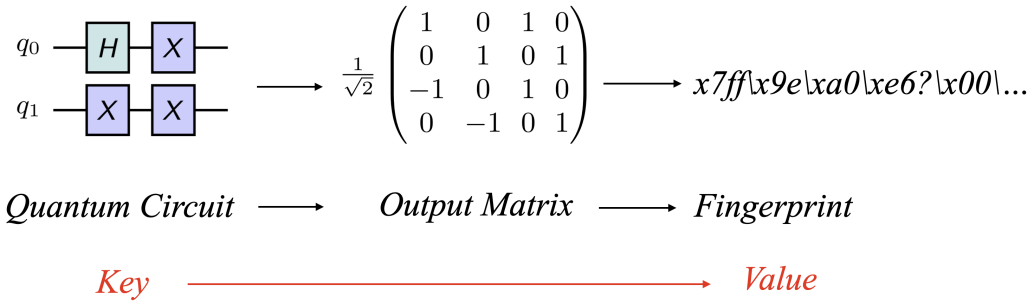


Figure 7.17: **A quantum circuit can be represented by a unitary matrix. The hash of this unitary matrix is its fingerprint.**

Key	Value
$[[H, X], [X, X]]$	$x7ffx9exa0xe6?x00\ldots$

Table 7.6: **Example of a hash table with one entry demonstrating the quantum circuit representation as the key and the fingerprint as the value.** Note that the fingerprint in this table is only for illustrative purposes (as the actual fingerprint is too long to be displayed).

Step 6: Create hash table of circuit identities. The hash table of substitutions includes the fingerprints as keys and an array of matching circuit identities as values. For each possible circuit we generate, we check if its fingerprint is in the hash table of circuit identities. If it is, we add the possible circuit to the matching fingerprint. If it is not, we add the fingerprint to the hash table and add the possible circuit to the array of matching circuit identities. The identities are stored implicitly in the hashtable – any set of matrices associated with the same key are equal and can be substituted for

each other. The identities are equal up to some (configurable) floating point precision. Table 7.7 is an example hash table with one entry which shows the fingerprint as the key and all the circuit identities that match to the fingerprint as the value.

Key	Value
$x7ffx9exa0xe6?x00\ldots$	$[[X,X],[X,X]]$ $[[H,H],[H,H]]$

Table 7.7: **Hash table with one entry demonstrating the fingerprint as the key and the array of substitutions as the value.**

7.6.2 Scaling

To estimate the computational resources to generate circuit identities, we derived a formula for the total number of possible circuits to be generated, as shown in Equation 7.2 based on Equation 7.1. The symbols are defined in Table 7.8.

$$S_l = \sum_{r=0}^{\lfloor n/2 \rfloor} \frac{n!}{r!(n-2r)!} g^{n-2r} t^r \tag{7.1}$$

$$S = S_l^d \tag{7.2}$$

Symbol	Definition
S	total number of possible circuits
S_l	number of possible circuits per depth
g	number of single-qubit gates
n	number of qubits
t	number of two-qubit gates
r	number of two-qubit gates in a single layer of an n-qubit circuit
d	depth

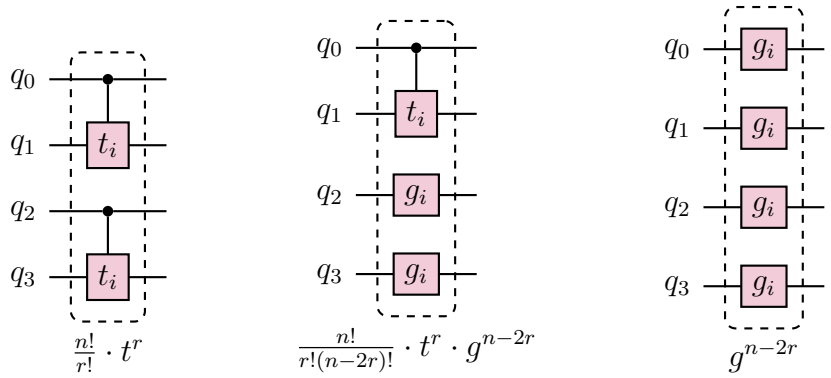
Table 7.8: **Symbols for scaling formula.**

As one can see, the number of possible circuits scales quickly with n which is to be expected. When using the optimiser, if the size of the tile is equivalent to the size of the input circuit, there is only one tile, meaning we enumerated this circuit during generation.

We derived the scaling formula by calculating the permutations of two-qubit and single-qubit gates in a quantum circuit. In particular, we can fix the number of two-qubit gates in a single layer and calculate the permutations of two-qubit and single-qubit gates in that layer. The formula is then derived by calculating the summation of the permutations across all possible number of two-qubit gates in a single layer. The maximum number of two-qubit gates in

a single layer of an n-qubit circuit is $\lfloor n/2 \rfloor$.

If $r = \lfloor n/2 \rfloor$ then there are $\frac{n!}{r!}$ ways to arrange the two-qubit gates in a single layer. For example, for $n = 4$, the maximum number of two-qubit gates in a single layer is $\lfloor 4/2 \rfloor = 2$ as shown in Figure 7.18a and there are $\frac{4!}{2!} = 12$ ways to arrange those two two-qubit gates.



(a) $r = 2$ (b) $r = 1$ (c) $r = 0$

Figure 7.18: **Permutations of single-qubit and two-qubit gates in $\mathbf{n = 4}$ circuit.**

To account for multiple types of two-qubit gates, we multiply $\frac{n!}{r!}$ by t^r where t is the number of types of two-qubit gates. For example, if our quantum circuit contains both CX and CZ gates, $t = 2$ and the number of ways to arrange two two-qubit gates in a single layer from the set of CX and CZ becomes $\frac{4!}{2!} \cdot 2^2$. For this example, where we have CX and CZ gates, $t_i \in [X, Z]$ for the gates

in Figure 7.18a.

If $r = 0$, this means the number of two-qubit gates in a single layer is zero and therefore the layer only contains single-qubit gates, which can include the identity gate. If g is the number of types of single-qubit gates, then g^n is the number of possible ways to arrange n gates in a single layer from a set of g types of single-qubit gates. For example, for $n = 4$, and $g = 3$, where the set of single-qubit gates is $g_i \in [I, X, H]$, there are 3^4 possible ways as shown in Figure 7.18c. If two-qubit gates are present in the n-qubit circuit, then there are g^{n-2r} possible ways to arrange the $n - 2r$ single-qubit gates in a layer, because the two-qubit gates occupy $2r$ positions in the n-qubit circuit.

If $r = 1$, then there is one two-qubit gate and two single-qubit gates in a circuit where $n = 4$ as shown in Figure 7.18b. We now have two positions to place the two-qubit gate, so the number of possible ways to arrange them is $\frac{n!}{r!(n-2r)!}$ and we multiply by t^r to account for multiple types of two-qubit gates. We also have two positions to place the two single-qubit gates, so the number of possible ways to arrange them is g^{n-2r}.

The scaling formula is the sum of permutations over all possible r's.

7.6.3 Optimiser

This section describes the Quanto optimiser, which applies the automatically generated circuit identities to optimise a quantum circuit.

7.6.3.1 Algorithm

Table 7.9 shows the inputs.

Symbol	Definition
quantum circuit	quantum circuit to optimise
i	the maximum number of qubits of the tiles
j	the maximum depth of the tiles

Table 7.9: **Inputs to the optimiser algorithm.**

Step 1: Generate tiles of the input quantum circuit. If the input circuit, which has depth (or width) m and number of qubits (or height) n, is larger than the size of the circuit identities generated, we split a circuit into sub-circuits, called *tiles*. The tile width, j, is the maximum depth of the circuit identities and the tile height, i, is the maximum number of qubits of the circuit identities. We iterate through the rows and columns in the quantum circuit matrix and

generate the indices of all possible tiles. The number of possible $i \times j$ tiles for an $n \times m$ circuit is $(n - i + 1) \cdot (m - j + 1)$.

Figure 7.19 shows the dimensions of the tiles and quantum circuit. Figure 7.20 shows all the possible tiles generated if we chose a tile with $j = 2$ and $i = 2$ in a quantum circuit with $m = 3$ and $n = 2$. Using the formula for the number of possible tiles, we see that there are four possible tiles, which corresponds with Figure 7.20.

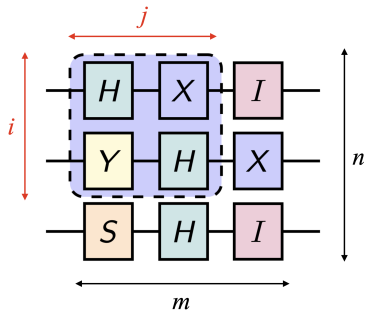


Figure 7.19: **Dimensions of a tile compared to the dimensions of a quantum circuit.**

Step 2: Check for valid tiles. Not all tiles generated from the circuit will be valid. An *invalid tile* is a tile which has a two-qubit gate that is cut off by the tile and the two-qubit gate is not placed at the boundaries of the tile. We discard invalid tiles. We can keep a tile if the two-qubit gate is placed at the

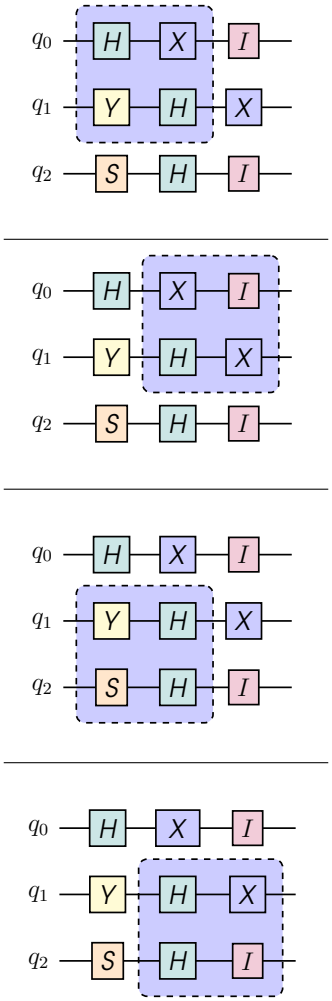


Figure 7.20: **All the possible tiles (shaded in blue) generated for a tile width and height = 2 for a quantum circuit with width and height = 3.**

boundary of the tile even it has been cut off by the tile (we call these tiles *valid tiles*). Figure 7.21 shows an example of an invalid tile, whereas Figure 7.22 shows an example of a valid tile.

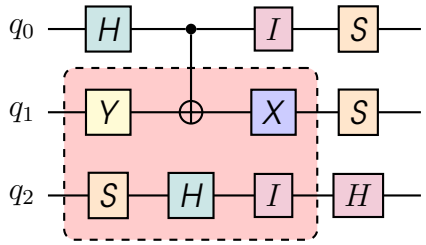


Figure 7.21: **An example of an invalid tile (shaded in red), because the CX gate is cut off and is not first or last on its wire.**

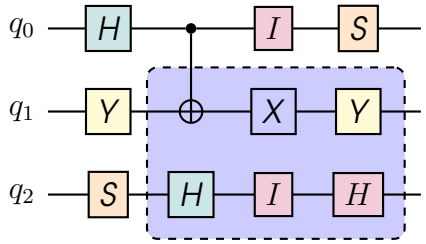


Figure 7.22: **An example of a valid tile (shaded in blue), because even though the CX gate is cut off, it is first or last on its wire.**

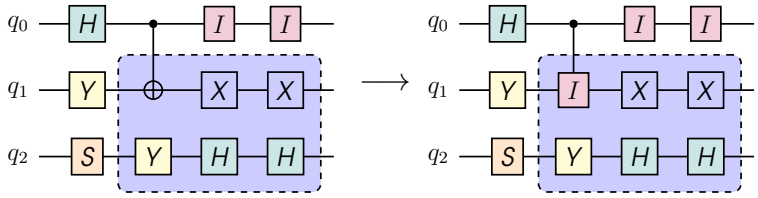
Step 3: Search for tile in hash table of circuit identities. For each tile we have generated, we first search in the hash table of fingerprints. If a tile contains a two-qubit gate that has been cut off *only at the boundary* of the tile (i.e. the valid tiles), we essentially 'push' the two-qubit gate outside of the tile (keeping the two-qubit gate in the overall quantum circuit) and add an Identity gate at the position of the cut-off two-qubit gate in the tile, so that the correct shape of the tile is recovered (as the tile shape was incorrect when we pushed the two-qubit gate outside of the tile). We need the correct shape

of the tile so that we can check for the tile in the hash table of fingerprints (which is used to find the fingerprint for the hash table of circuit identities). To implement this in practice, we first have to 'replace' the two-qubit gate with an Identity gate in order to make the tile shape correct and searchable in the hash table of fingerprints. Then the two-qubit gate is later replaced back (in Step 6). Part (a) in Figure 7.23 shows this transformation. The now-corrected tile is represented as a matrix, as shown in Part (b) in Figure 7.23 and we can search for this matrix in the hash table of fingerprints. This gives us the fingerprint of the tile, without needing to calculate the unitary matrix of the tile if the tile is already in the hash table. After we have obtained the fingerprint, we can search for circuit identities in the hash table of circuit identities as shown in Part (c) in Figure 7.23. Part (d) in Figure 7.23 shows circuit identities for the tile.

Once we have the fingerprint, we then look up the possible substitutions by its fingerprint in the hash table of circuit identities.

Step 4: Apply substitutions via cost-based search. We then iterate through the circuit identities for each tile and evaluate the cost of each circuit

a)



b)

$$\begin{pmatrix} I & Y \\ X & H \\ X & H \end{pmatrix} \longrightarrow$$

Key	Value
$[[I,Y],[X,H],[X,H]]$	$x7ffx9exa0xe6?x00\ldots$

c)

Key	Value
$x7ffx9exa0xe6?x00\ldots$	$[[I,Y],[I,H],[I,H]]$ $[[I,Y],[X,I],[X,I]]$ $[[I,Y],[I,I],[I,I]]$

d)

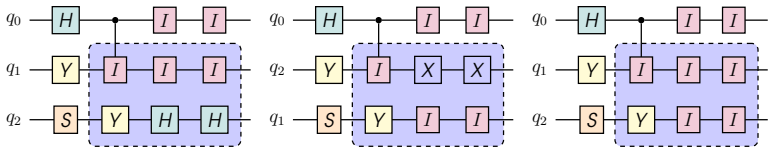


Figure 7.23: **(a) The cut-off CX gate in the tile is 'replaced' with an Identity gate to make the tile searchable in the hash table of fingerprints (the CX is replaced back at a later stage). (b) The matrix representation of the tile is searched for in the hash table of fingerprints, which returns its fingerprint. (c) The fingerprint is used to find the circuit identities for the tile in the hash table of circuit identities. (d) The circuit identities for the tile.**

identity. In this paper, we simply use the depth of the circuit (ignoring identity gates) as the circuit's cost; the cost function can be customised if desired. Table

7.10 shows the cost for the circuit identities presented in Figure 7.23. In this case, circuit (iii) in Table 7.10 has the lowest cost and will be used to substitute the tile in the quantum circuit.

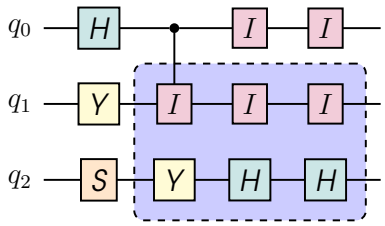
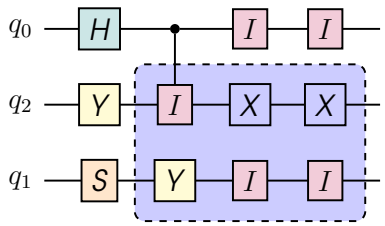
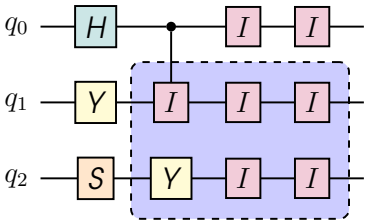
	Circuit Identity	Cost
i		3
ii		3
iii		1

Table 7.10: **Circuit identities and their cost.**

Step 5: Check for valid substitutions. Adding a substitution might make the entire quantum circuit invalid. For example, two-qubit gates may be cut

off or not aligned properly. Additional steps, explained below, are needed to ensure the final circuit has valid two-qubit gates.

Step 6: Apply the substitution to the circuit. We apply the substitution to the quantum circuit. If the original tile contained a cut-off CX, then we have to do a couple more transformations before applying the substitution. Figure 7.24 shows the steps. Step (a) is equivalent to Step (a) in Figure 7.23. In Step (b), we apply the substitution with the lowest cost, which was the circuit identity (iii) in Table 7.10. We now need to transform this new circuit so that it is consistent with the gates in the original circuit. In particular, this new circuit has the CX target gate replaced with an Identity gate. Step (b) replaces the Identity gate with the original CX target gate. Step (c) shows the total transformation from the original circuit to an optimised circuit, after only one tile substitution. The Identity gates have been omitted to show that the depth of the circuit has decreased and therefore the circuit has been improved. We then continue to the next tile with the newly replaced substitution.

Step 7: Repeat process. Steps 1 - 6 are repeated for a fixed number of times. In our implementation, ten times were sufficient.

a)

b)

c)

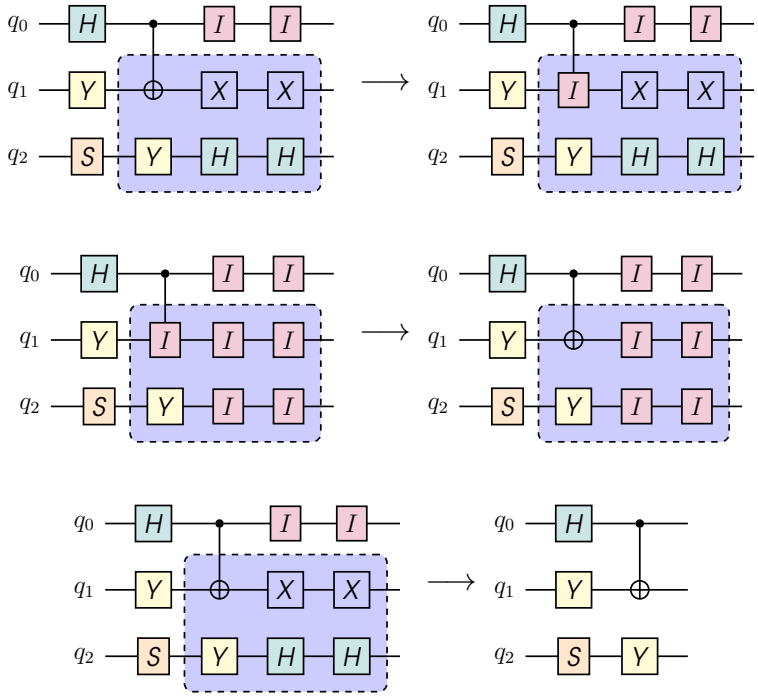


Figure 7.24: **Applying the substitution.**

7.6.3.2 Reordering Qubits

In order to find more potential optimisations in the circuit, we can also reorder the qubits. Figure 7.25 shows an example of a circuit with width 5 and height 3 (this is a modification of the example circuit in Figure 7.4). In this circuit, we have a couple of long-range two-qubit gates, which are CX gates between qubit q_0 and qubit q_2. If the tiles have width 5 and height 2, which is shown in Figure 7.26, then optimisations between distant pairs of qubits, in this case q_0

and q_2 would be missed.

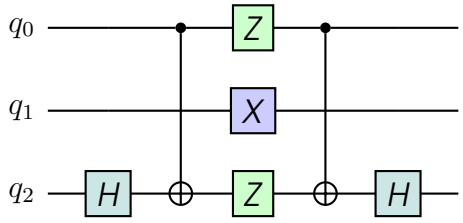


Figure 7.25: **Example of a quantum circuit with a long-range two-qubit gate.**

One way to address this issue is to reorder the qubits. Figure 7.27 shows the quantum circuit where the qubits have been reordered. Now, with a tile of width 5 and height 2, we can find an optimisation for the circuit and reduce its width from 5 to 1, which is shown in Figure 7.28.

7.6.4 Runtime

The optimisation process involves finding an optimum sub-circuit for all of the possible tiles in the input circuit. For each possible tile, the optimum circuit is found in $O(1)$ because of our construction of the hash table of circuit identities. The number of possible $i \times j$ tiles for an $n \times m$ circuit is $(n-i+1) \cdot (m-j+1)$. Therefore, the runtime of the optimisation algorithm is $O(m \cdot n)$. If qubits are reordered to improve the effectiveness of the optimisation, then this adds

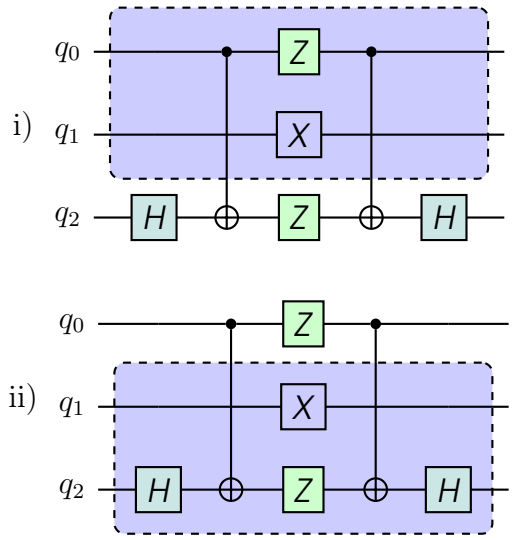


Figure 7.26: **Splitting the circuit into tiles of width 5 and height 2.**

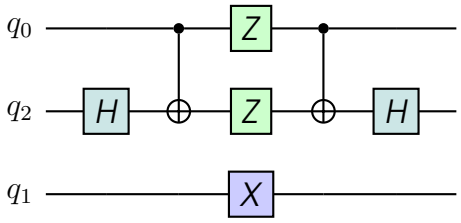


Figure 7.27: **Example of a quantum circuit with qubits reordered.**

$O(n!)$ to the algorithm and therefore the algorithm with qubit reordering, takes time $O(m \cdot n!)$. Figure 7.13 shows an example of the runtime of optimising a quantum circuit in practice.

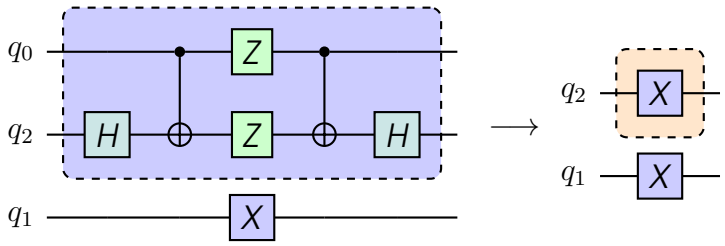


Figure 7.28: **Steps to optimise the circuit when the circuit is split into tiles.** The boxes shaded in blue are the tiles. The boxes shaded in orange are the substitutions.

7.6.5 Implementation

Quanto is implemented in Python and can be used as an API in principle. In particular, Quanto has the following modules:

7.6.5.1 Circuit Module

Quanto can take *qasm* (quantum assembly language) files [224] as input [225, 226, 227, 228, 229], which is a popular way to construct circuits and makes our optimiser compatible with other quantum circuit frameworks, such as Qiskit [45] and Cirq [230], which can input/output *qasm* files. Our code converts the *qasm* file into our object representation of a circuit as shown in Figure 7.15. We represent a circuit as a grid, where the rows represent the layers and the columns represent the gates applied to the qubits.

7.6.5.2 Subs Module

The user can specify a gate set to generate circuit identities. Quanto also has several pre-set gate set options apart from customised gate sets. If no gate set is specified, then our code automatically detects all the gates used in the input circuit and uses that as the gate set. This module generates the circuit identities and builds the circuit identities database, which is used for the optimiser module.

7.6.5.3 Optimise Module

The optimise module takes the generated circuit identities database and optimises the input circuit. It outputs an optimised circuit, along with metrics, such as the final circuit depth. The circuit can be saved as a *qasm* file or a qiskit circuit (applicable for certain gates).

7.7 Review

Our aim in this chapter was to design a new type of quantum compiler that uses more complex circuit identities for optimisation, compared to the simple set of

rules used in previous work, by automatically generating circuit identities and applying them to larger circuits via a unique circuit-splitting method. We were able to achieve this goal by creating a program that generated circuit identities given a gate set, number of qubits, and circuit depth. We then designed a method to split the larger circuit into sub-circuits so that each sub-circuit could be optimised with the generated circuit identities.

Our results showed that on certain quantum circuits, our optimiser was able to outperform existing quantum compilers, by finding equivalent quantum circuits with a lower depth. Due to several limitations as outlined in the *Limitations and Future Work* section (Section 7.5.2), our optimiser's performance on larger quantum circuits is not as effective, though future work that we detailed could address this. We also demonstrated that reducing the quantum circuit depth via our quantum optimiser led to a quantum circuit solution with less noise and a reduced runtime of the quantum circuit. By generating circuit identities, we could also find novel circuit identities, compared to previous manual circuit identities, and also extend the reach of quantum compilers by providing circuit identities for custom gates, such as the iSWAP gate on Google's quantum hardware. By using a customisable cost function, our compiler could take into

account a variety of more real-time quantum circuit metrics (such as quantum gate errors on devices) for optimisation, unlike previous compilers.

The work in this chapter is the first step to helping make quantum neural networks more efficient and accurate by reducing the runtime and noise respectively. Although the work in this chapter focuses on the optimisation of quantum circuits, as quantum neural networks are types of quantum circuits, the work in this chapter can be applied to quantum neural networks, though further work may be needed to adapt our quantum compiler for a quantum neural network context. For example, one limitation of the work in this chapter is working with parametrised gates, which are used in quantum neural networks, and therefore future work would need to take this into account to make it directly applicable to quantum neural networks. A follow-up paper adapted the technique in this chapter to apply the optimiser to parametrised quantum circuits specifically [231].

One limitation of our work is that the compiler is not as effective on larger quantum circuits, which we believe is due to a large number of invalid tiles that are generated when we break down the larger quantum circuit into tiles,

especially if the quantum circuit has many two-qubit gates. We believe that the fixed-size circuit identities that we generate are effective for optimisation, but the bottleneck is whether we can generate enough valid fixed-size circuit identities. This issue was addressed in the follow-up paper [231]; they used a different method to optimise the quantum circuit with fixed-size circuit identities. Their results provide evidence for our suggestion that the fixed-size circuit identities are effective for optimising larger quantum circuits, as they built a compiler using the technique of generating fixed-size circuit identities as we proposed. They applied these circuit identities on larger quantum circuits and showed that these fixed-size circuit identities were effective in optimising the quantum circuits, which they measured by the number of quantum gates. This suggests that these fixed-size identities are still useful as the size of the quantum circuit increases.

Another question is how the work in the previous chapter, Chapter 6, affects the work in this chapter. Given that our quantum compiler only finds mathematically equivalent quantum circuits for optimisation, the inductive bias and expressivity of the quantum neural network would not change as a result of the optimisation. Therefore the quantum compiler does not affect the bias and

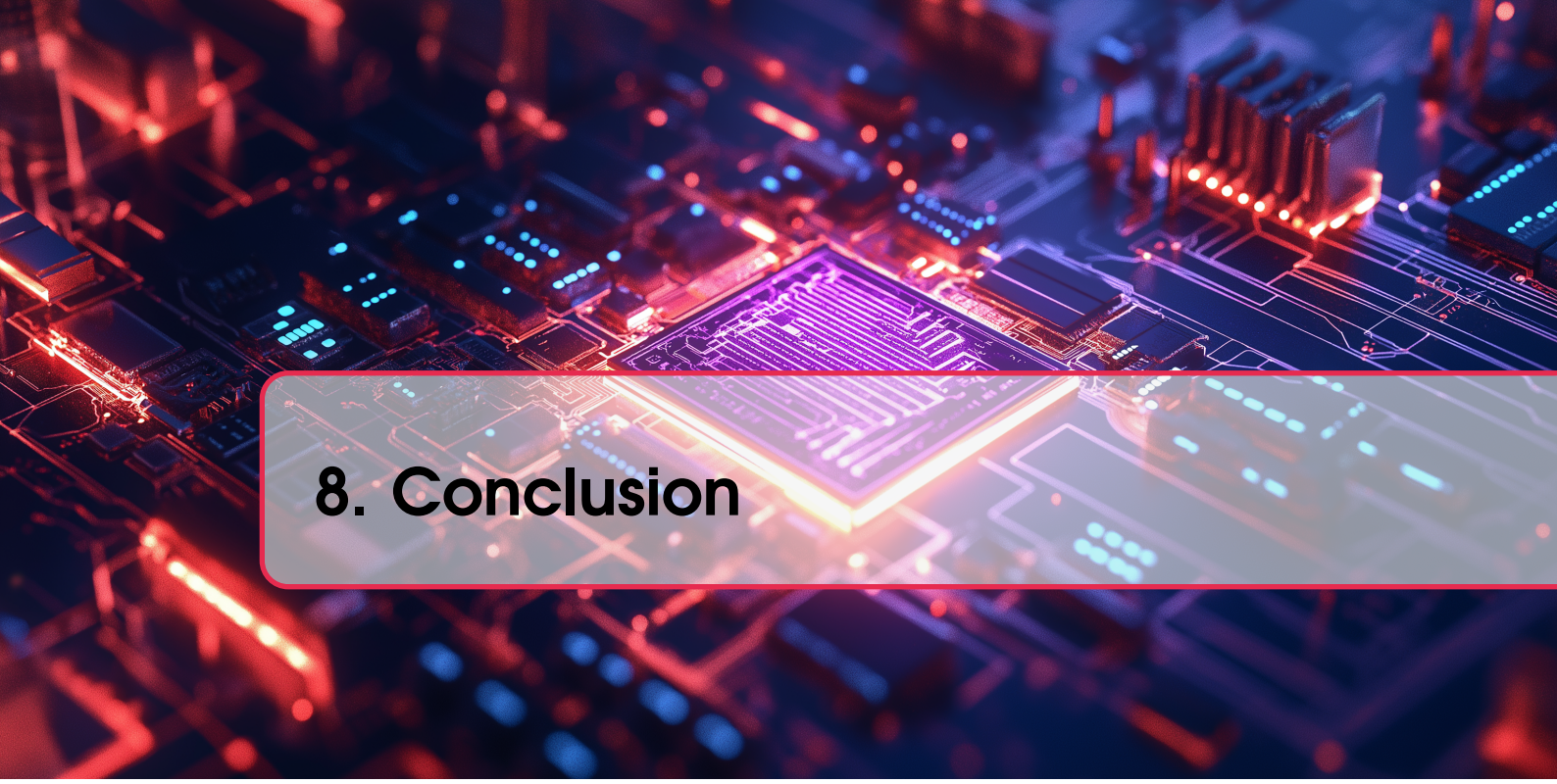
expressivity of a quantum neural network.

It would be interesting to see if the reverse holds true, that is whether the bias and expressivity of a quantum neural network would affect the performance of the quantum compiler. Quantum neural networks with a higher expressivity typically have a longer depth; in fact in Chapter 3, we showed that in order to have full boolean expressivity, the quantum circuit needed to contain 2^n gates where n is the number of qubits (Section 3.4.2.2). As a more expressive quantum neural network would have a higher number of gates and depth, it is likely that the quantum compiler would perform worse, as the quantum circuit would need more splitting, which could lead to sub-optimal optimisation. A similar argument could be said regarding the inductive bias; certain encoding methods require a higher number of quantum gates and could therefore lead to a worse performance for the quantum compiler. For example, amplitude encoding could require a lot of gates for preparation of a particular quantum state and therefore could not be as well optimised by the quantum compiler. Future work could explore this relationship of how the inductive bias and expressivity of circuits affects the performance of quantum compilers.

Another interesting area of future work includes the design of heuristic quantum compilers, that is quantum compilers that do not find exact quantum circuit equivalences but rather approximations. One question then is how to measure approximately equivalent quantum circuits. The inductive bias and expressivity could be interesting measures to use for finding approximately equivalent quantum circuits. Therefore a suggestion for a new type of quantum compiler could be one that finds another quantum circuit with an equivalent inductive bias and/or expressivity, although the quantum circuits may not be mathematically equivalent. Such approximately equivalent quantum circuits could be used for the optimisation of quantum neural networks that need to keep the inductive bias and/or expressivity the same during optimisation.

Furthermore, it could be interesting to create quantum compilers that change the inductive bias and/or expressivity of a quantum neural network. Similar to how a transpiler produces code in a different programming language when given code in another programming language; a *quantum bias transpiler* could produce a quantum circuit with a different inductive bias/expressivity when given a quantum circuit with another inductive bias/expressivity.

In the next chapter, Chapter 8, I provide a summary of the thesis and outline future work. In Appendix A, we look at applying variational quantum algorithms to a real-world problem. At the end of Appendix A, we review how the inductive bias and expressivity and optimisation of quantum neural networks explored in Chapter 6 and this chapter, Chapter 7, affect this quantum application and vice versa.



8. Conclusion

8.1 Overview 451

8.2 Thesis Summary 452

8.3 Research Summary and Future Research 463

8.4 Concluding Remarks 477

8.1 Overview

This thesis has contributed to an understanding of the performance of quantum neural networks as general-purpose learning tools, how quantum circuits can be optimised, and how variational quantum algorithms can be used to solve useful real-world problems. We have investigated these research questions

through three lenses: the foundation, optimisation and application of quantum algorithms. This thesis has integrated domain expertise in different fields including quantum computing, machine learning, classical compilation, and mining engineering. In this conclusion, I summarise the thesis, including our main contributions that can help advance the field of quantum computing. Finally, I suggest future work that can be built upon the findings of this thesis, so further progress in the field can be made. The goal to understand the foundation, optimisation, and application of quantum algorithms has been achieved through three main projects I did during my DPhil.

8.2 Thesis Summary

8.2.1 Thesis Introduction

Quantum computing has fascinating roots in computing and quantum physics. Impressive milestones have been reached as the field matures and progresses towards useful applications. Despite the growing infrastructure and enthusiasm in the field, significant challenges remain, as researchers aim to understand their theory and practicality better. At the beginning of this thesis, I set the context

for this thesis, briefly journeying from the early days of quantum computing to the current context, as I touched on the areas of traditional quantum algorithms, quantum hardware, quantum tools, NISQ and fault-tolerant devices, hybrid quantum-classical algorithms, quantum machine learning, and quantum neural networks. To tie all of the research in this thesis together, I created a framework of how to work with quantum algorithms, consisting of three conceptual layers: foundation, optimisation, and application, and discussed how each layer builds on top of each other. I then showed how the original research my coauthors and I performed in this thesis fits into this framework. With this context, I presented the contributions of this thesis, as I summarised the context, research done, the results, and contributions to the field for each chapter of original research. I summarised the chapters of this thesis and the publications that have been made. Finally, I discussed the computational resources used in order to make the research possible.

8.2.2 Quantum Computing Chapter

After introducing the thesis, I provided additional background information for the reader to set the context of this thesis. I explored the origins of quantum

computing from its roots in computing and quantum physics and how the field progressed in its early days through the development of traditional quantum algorithms and proposals for quantum hardware that were built as proof-of-concepts. As quantum hardware matured, quantum cloud platforms emerged to make the quantum hardware accessible and quantum software emerged to make quantum algorithms accessible. In 2018, John Preskill coined the term NISQ (Noisy Intermediate-Scale Quantum) to describe the regime of quantum hardware. In 2013, variational quantum algorithms were developed that were designed to be run on NISQ devices. As the hardware and software developed, quantum tools materialised, such as quantum compilers to optimise the mapping from software to hardware and to run more efficient quantum algorithms. The milestone of quantum supremacy had been reached in 2019, which sparked further hope and interest in the field. As a result of these trends, researchers applied quantum computers to various applications with the hopes of finding quantum advantage. One popular application is machine learning, which is widely applicable to many industries. I reviewed the roots of quantum machine learning in the traditional quantum algorithms and how that shifted to variational quantum algorithms with the rise of the NISQ devices

and hybrid quantum-classical algorithm framework. Subsequently, quantum neural networks became more popular and researchers began to use them to perform machine learning tasks on small-scale datasets.

8.2.3 Quantum Neural Networks Chapter

I then explored the origins of quantum neural networks, as early as the first quantum algorithms, and the early proposals for a building block of a quantum neural network. I showed that as variational quantum algorithms were developed, the two concepts merged as quantum neural networks became known as a subclass of variational quantum algorithms for machine learning tasks. I then showed how quantum neural networks work, including its three main components for encoding, training, and measuring. I presented the mathematics for common encoding methods and how a variational circuit works. I explained the expressivity of quantum neural networks and distinguished boolean expressivity from Hilbert space expressivity. I presented fully expressive and non-fully expressive quantum neural networks that would later be used in Chapter 6. To conclude how quantum neural networks work, I described the measurement operators used and the learning process, including analytic gradients.

After this, I explored quantum kernels and the link between them and quantum neural networks. I reviewed the advantages and challenges of using quantum kernels versus quantum neural networks. I then described the technicalities of quantum kernels and in particular how to construct a kernel matrix from the common encoding methods I reviewed earlier, which were used for study in Chapter 6. In addition, I described the properties of quantum neural networks that have been found based on the literature, including barren plateaus, model capacity, role of data, and bias in quantum kernels. Finally, I reviewed the idea of quantum advantage and what is known about quantum advantage for supervised machine learning. I pointed out the challenge of finding quantum advantage for machine learning problems and summarised the current state of the field.

8.2.4 Classical Neural Networks Chapter

I then moved into classical neural networks and summarised its history, including significant milestones that helped move the field forward, including the invention of backpropagation and the AlexNet deep neural network that classified images successfully. I reviewed the theory of neural networks, including its base in

statistics which assumed a bias-variance tradeoff. I highlighted how several papers noted that empirically the performance of deep neural networks did not match the bias-variance tradeoff as these networks had both a low bias error and low variance error. I showed that a couple of theories emerged to reconcile this discrepancy, including that the deep neural networks have simplicity bias, which allows them to generalise well on real-world data as most real-world data is structured (i.e. simple). I explained further about inductive bias and simplicity bias in deep neural networks, as well as the no free lunch theorem. I reviewed the key findings in algorithmic information theory, including the complexity of a function, which provide the justification for simplicity bias in deep neural networks. I reviewed the technicalities of classical neural networks and how they work. I reviewed key concepts that would be relevant for Chapter 6 including expressivity and generalisation error. Finally, I discussed the boolean system, which is used as a dataset for investigating inductive bias.

8.2.5 Categories of Quantum Advantage and Algorithms Chapter

After reviewing background information relevant for this thesis, I move onto contributions made by this thesis. I first clarify the various types of quantum algorithms and show how quantum neural networks are a subclass of variational quantum algorithms, which are a subclass of hybrid-quantum classical algorithms, which are a subclass of quantum algorithms. I distinguish the subtle differences between variational quantum algorithms and quantum neural networks and mention that quantum neural networks could evolve into a new class, apart from variational quantum algorithms. I then connect these types of quantum algorithms to the research presented in this thesis. Next, I move onto quantum advantage and present a framework for different measures of quantum advantage, including the performance and resources used. I then present a flowchart I created to identify definitions, key questions, and goals regarding quantum advantage. I explain the different branches of the flowchart and clarify the goals that arise from the flowchart. Finally, I show examples of quantum advantage in the literature and categorise them according to this framework.

8.2.6 Foundation of Quantum Neural Networks Chapter

With all of this background information, I then presented the research we performed on investigating whether quantum neural networks have a simplicity bias. In this chapter, we introduced the motivation for the research, starting from the success of deep neural networks due to their simplicity bias and high expressivity and how that could give insights into the potential success of quantum neural networks. We presented the results, by summarising how the encoding method affected the inductive bias and expressivity of the quantum neural network, which suggested a bias-expressivity tradeoff as encoding methods with a better bias had a worse expressivity and vice versa. We presented the data showing the inductive bias of quantum neural networks with different encoding methods and also showed how the generalisation error is affected by the encoding method. We then presented more detailed results on the boolean expressivity of quantum neural networks. Finally, we presented the methods we used to find the inductive bias in quantum neural networks and quantum kernels, as well as how we trained the quantum neural networks to obtain the generalisation error results. We summarised the findings and implications in the discussion, demonstrating that our results suggest quantum neural networks

will not have an advantage as a general-purpose learning tool on classical data. After the discussion, we presented a proof by contradiction and proof by induction we derived for the expressivity and inductive bias of certain encoding methods respectively. We then presented additional experiments and results that supported the claims made in the paper, which included other measures and ways to visualise the data. We also presented the results on the inductive bias of quantum kernels, which correspond with those of the quantum neural networks we studied. Furthermore, we presented how we set up the training and test datasets for the boolean system to obtain the generalisation error results and reviewed the generalisation error on a particular function. In addition, we discussed the various methods we used for training the quantum neural networks and presented the data from them. Finally, we used the methods to display the inductive bias for the convolutional quantum neural network, another type of quantum neural network.

8.2.7 Optimisation of Quantum Circuits Chapter

After a study of quantum neural networks and their fundamental properties, I move on to the optimisation of quantum neural networks, in particular,

how to make quantum circuits more efficient. Improving the efficiency of quantum circuits would also potentially improve the efficiency for quantum neural networks, as quantum neural networks are built upon quantum circuits. In Chapter 7, we introduce the previous work on optimising quantum circuits and the gaps in their performance, which motivates our research. We introduce how our quantum compiler works and why it fills in those gaps and leads to better performance on certain quantum circuits. We then present the results of the research, which show that our quantum compiler can generate existing circuit identities as well as find novel circuit identities. We then show results on the benefits of optimisation as our results show that our optimiser can reduce the noise and execution times on quantum hardware. We discuss related work including automatically generating identities and superoptimisation, the classical research that inspired the methods we used in this research. We finally discuss the challenges of the research including the scalability and sub-optimal optimisations as well as parametrised gates and future work. After the discussion, we delve in-depth on the methods we use, including how we generate the circuit identities and split the quantum circuits into tiles which we optimise. We also derive a scaling formula and runtime formulas. Finally,

we discuss the practical implementation of the optimiser and the modules that we programmed. At the end of the chapter, I connect this work to the work in the previous chapter by discussing how the work affects each other and suggest future work that could look at these synergies, including a compiler that uses inductive bias and/or expressivity as a measure for finding approximately equivalent quantum circuits, and the idea of a quantum bias transpiler.

8.2.8 Application of Quantum Computers Chapter

In the Appendix I include research on a quantum application of variational quantum algorithms. We introduce the concept of combinatorial optimisation problems and quantum algorithms for approximately solving them. We introduce the ultimate pit problem as a real-world application that we approximately solve on a small-scale on quantum computers. We briefly explain the classical algorithms, such as LG and pseudoflow, that have been used to solve this problem and their commercialisation for real-world open-pit mining. Next, we present the methods we used for approximately solving small-scale instances of this problem on a quantum computer, including mapping the problem onto a Hamiltonian and constructing a quantum circuit ansatz to mimic the problem.

We then present my coauthors' idea of domain decomposition, to solve larger-scale instances of the problem on the small-scale hardware. We present the results, which include numerical simulations and runs on quantum hardware, which show that we are able to minimise the expectation value of the Hamiltonian and find a good solution for the ultimate pit problem, which we compare to the classical algorithm solutions. For the hardware runs, we show the effect of error mitigation to reduce the hardware errors. Finally, we conclude that our research was a first step towards addressing quantum computing for mining applications. At the end of the chapter, I suggest future work that could use the findings of the previous chapters to this work.

8.3 Research Summary and Future Research

8.3.1 Foundation

8.3.1.1 Summary of Results and Contributions

The first project had the aim of understanding the foundation of quantum neural networks, which was done through studying the inductive bias and expressivity of quantum neural networks. As the inductive bias is essential to

how a learning algorithm performs, our work shed light on the learning and performance of quantum neural networks. We looked at common encoding methods for quantum neural networks and measured their inductive bias; we came up with a proof to show that the most basic form of encoding data, basis encoding, is not useful as it does not encode a bias in a quantum neural network and therefore is no better than a random learner. We showed that it is possible to introduce an 'artificial' inductive bias into a quantum neural network with no bias, by intentionally restricting the expressivity of the quantum neural network. We showed that various other encoding methods, including the ZZ feature map and random relu transform, have an inductive bias that is not towards simple functions but rather low-entropy functions. The ZZ feature map inductive bias is only useful for a particular task, namely performing well on the parity function, which it was designed for. The encoding method that most closely mimics classical deep neural networks is amplitude encoding as it has a bias towards simple functions. It has the disadvantage, however, of not being fully expressive. We demonstrated that the inductive bias affects the generalisation performance, which highlights the importance of the inductive bias on the performance of quantum neural networks. Our additional results

also confirmed that the inductive bias of quantum kernels is similar to quantum neural networks.

Overall, our results suggested a bias-expressivity tradeoff with regards to the encoding method used in a quantum neural network. Therefore, our work has showed the limitations of quantum neural networks, one being that unlike deep neural networks, the bias towards a particular function needs to be encoded in the quantum neural network. In addition, our work showed that the bias-expressivity tradeoff we observed is a limitation of quantum neural networks compared to deep neural networks, as deep neural networks are able to have a good inductive bias and high expressivity. Furthermore, the thought is that deep neural networks are good general-purpose learning tools because of their simplicity bias, but our work suggests that quantum neural networks will not be good as general-purpose learning tools on real-world data because the only case of a simplicity bias leads to reduced expressivity, which limits its performance. Our work also highlights the important properties for a general-purpose quantum neural network (the simplicity bias and high expressivity) and researchers can use our methods to evaluate their quantum neural network encoding methods.

8.3.1.2 Future Research

Upon these findings, there are a range of possibilities for future research. In our work, we investigated the most common encoding methods. It is possible for researchers to come up with their own encoding methods. Therefore, the techniques and methods we developed in this work for measuring the inductive bias and boolean expressivity can be used for other custom encoding methods, to help understand their performance. It may be possible to find an encoding method with a good inductive bias and high expressivity; therefore our work can help researchers know what they need to look for when designing a quantum neural network and they can use our methods to help them evaluate their own quantum neural network designs.

The bias-expressivity tradeoff we suggested could be studied further, in particular to see if it is possible to prove. Our work suggests that alternative quantum neural network architectures will be needed to be comparable to deep neural networks, so future work could explore alternative quantum neural network architectures with good expressivity and inductive bias. A follow-up paper [30], which builds upon our work, proposed an architecture, a *deep quantum*

neural network, that takes the simplicity bias of the amplitude encoding method and makes it fully expressive. This is a promising step towards new types of quantum neural network architectures that our work suggests is needed, but this particular architecture is not practical and thus future work can look into making more efficient versions of it.

Moreover, researchers could use the methods in our work to study the inductive bias and boolean expressivity of quantum neural networks used in the literature, in particular papers that claim a quantum advantage, which could help provide explanations for their quantum advantage and identify other potential areas for quantum advantage. Furthermore, future research could involve studying how noise affects the inductive bias, which would be relevant for running the quantum neural networks on quantum hardware. In addition, Dingle et al. [35] explain the conditions for a map to have a simplicity bias as we outlined in Chapter 4 Section 4.6; future work could involve studying these conditions more closely and seeing which conditions the encoding methods in the quantum neural network satisfy or fail that result in it having a simplicity bias or not respectively. Alternatively, if one finds that the quantum neural network's behaviour can not fit into the conditions presented, then the reason for those

particular conditions may need to be re-considered. As well as that, further work could involve studying how types of quantum gates affect the inductive bias; for example looking at whether quantum circuits with stabiliser gates (gates in the Clifford group which are generated from the Hadamard, phase gate S, and CX gate set, and can be efficiently simulated classically) or circuits with bounded entanglement produce a type of bias. Furthermore, our results suggest that quantum neural networks behave with a U-shaped error curve as described in Chapter 4 Section 4.4 as opposed to the double-descent curve in deep neural networks; future work could look into the possibility of a double-descent curve in quantum neural networks.

8.3.2 Optimisation

8.3.2.1 Summary of Results and Contributions

The second project had the aim of understanding the optimisation of quantum algorithms, which was done through developing a new method for optimising quantum circuits. In this research, we discovered new circuit identities and optimised quantum circuits in a way that other existing quantum compilers could not do, as the latter relied on a few hand-coded rules for optimising

quantum circuits, which limits the optimisations possible. Our quantum compiler eliminates the need for hand-coded rules and provides more possible solutions for quantum optimisations. Additionally, because previous compilers used a few hand-coded rules for optimising quantum circuits, they were unable to be used across a variety of quantum devices that had different operations. As quantum hardware development accelerates, more quantum computers are being built with different techniques and therefore unique quantum operations. Our solution aligns with the growing development in quantum hardware, as our solution is flexible and portable to be used for any quantum device. Therefore, it is possible for our database of new circuit identities to be used to create hand-coded rules for optimisation for specialised cases, such as a quantum computer with unique quantum operations. Furthermore, our compiler can optimise a quantum circuit according to a customised cost function, which could take into account a variety of metrics of the quantum circuit, leading to potentially a new error mitigation technique and improved optimisations. Our results also showed that optimised circuits lead to fewer errors in the solution and a reduced runtime.

8.3.2.2 Future Research

Future work could involve taking our generator of circuit identities and using them in compilers or quantum hardware that have custom quantum operations. Future research could adapt the methods in our quantum compiler for variational quantum algorithms specifically, where there are parametrised gates. A follow-up paper [231], which builds upon our work, expanded our research by including optimisations for gates with parameters and by using a different circuit representation. Future research could investigate different circuit representations and how that affects the optimisation of quantum circuits. Furthermore, future research could look at the practicality of optimising these quantum circuits in the context of hybrid quantum-classical algorithms and how that can be done more efficiently. For example, as the specific gates typically change in such algorithms throughout a feedback process, a quantum compiler that compiles at each iteration would need to be designed to be very efficient; it is possible that heuristics could be used in that context to suit the quantum compiler to the specific practicalities of hybrid quantum-classical algorithms.

Future work could also look into the effectiveness of this compiler as an explicit error-mitigation technique by using quantum gate errors in the cost function for evaluating quantum circuit substitutions in the optimisation process. Additionally, future work could create different cost functions, based on a variety of metrics (such as the number of quantum gates, the number of two-qubit gates, and the quantum gate errors on devices), and investigate which cost functions are most effective.

With regards to the limitations of our quantum compiler (as detailed in Chapter 7 Section 7.5.2), future work could involve making the circuit identity generator more efficient, studying how to improve the performance of the quantum compiler on larger quantum circuits, and doing a more in-depth analysis of the effects of optimisation on noise and runtime.

As detailed in Chapter 7 Section 7.7, I have thought of new ideas for future research as a result of the connection of the work on inductive bias and expressivity in Chapter 6 and the work on the quantum compiler in Chapter 7. This suggested future research includes studying how the inductive bias and expressivity affects the quantum compiler performance and designing heuristic

quantum compilers that use inductive bias and/or expressivity as approximately equivalent measures for finding approximately equivalent quantum circuits for quantum circuit optimisation. Furthermore, I suggest the creation of a *quantum bias transpiler* that can produce a given quantum circuit with a new type of specified inductive bias and/or expressivity, keeping other properties the same.

8.3.3 Application

8.3.3.1 Summary of Results and Contributions

The third project had the aim of understanding the application of quantum algorithms, which was done through customising a variational quantum algorithm to solve a real-world optimisation problem in the mining industry. Since the advent of traditional quantum algorithms, the aim of quantum computing has been to find quantum speedups for practical applications. Whilst theoretically, quantum speedups were found, practically they could not be realised due to the infancy of the quantum hardware. Later on, new types of quantum algorithms were designed, hybrid quantum-algorithms, which could be run on the quantum hardware but lacked the theoretical basis for a quantum speedup. Due to the growing interest in quantum computing, and the quantum hardware and

software becoming increasingly available, researchers were interested in finding real-world applications of these new types of algorithms. Various applications were found in finance, logistics, and machine learning. Our aim in this project was to find a new application, and we did so by looking at a commercial problem, the ultimate pit problem, in the mining industry. To the best of our knowledge, this was the first attempt of using heuristic quantum algorithms to solve this open-pit mining optimisation problem. We did this by choosing a combinatorial optimisation problem in mining engineering and developing a method to map this problem onto a quantum computer, as well as developing a quantum ansatz for solving this particular problem. This research contributes to an understanding of mapping a real-world combinatorial optimisation problem to a quantum device. This research is also the first step of applying quantum computers to the mining industry to the best of our knowledge. Furthermore, researchers could use the decomposition technique in their own context to solve larger-scale instances of a problem to smaller-scale devices.

8.3.3.2 Future Research

Future research could involve adapting our approach and applying it to other real-world problems. In addition, in our research, we simplified the open-pit mining optimisation problem so that we could run it on a classical simulator and quantum hardware. Future research could also involve investigating more complicated models of the open-pit mining optimisation problem to solve on the quantum computers; for example, by adding more sophisticated constraints, or generalising the problem to three dimensions. Additionally, future work could improve the decomposition technique by allowing overlapping fragmentations for example. In addition, in our work, we used a penalty term to enforce the smoothness constraints, but this increases the complexity of the algorithm, so penalty-free alternatives could be explored.

Follow-up papers, building upon our work, have created a quantum-inspired tensor network algorithm to solve this problem without using hyperparameters (e.g. the penalty term) [232] and have solved this problem in three dimensions [233] as we suggested in our work.

Given that there is no known generic quantum speedup for these quantum variational algorithms, we do not know if our algorithm offers a quantum speedup for this problem. If a generic quantum speedup for variational algorithms is found, it could be possible that this algorithm could provide a quantum advantage. Alternatively, if it is possible to theoretically compare the classical complexity and quantum complexity for this problem, then a possible quantum advantage could be found – this could be explored by future work.

Decomposing the problem into 'fragments' could potentially be a promising approach to many problems by using small-scale quantum computers for larger-scale real-life problems. This decomposition technique could potentially be transferable to problems that are composed of discrete units (the blocks in our case). Problems with continuous values could be discretised by defining discrete units based on intervals of the continuous values. Future work could investigate the applicability of this method to other problems.

In addition, future work could study the effect of the fragment size on the performance of the algorithm. If the fragment is too small (below the classical simulation limit and therefore can be efficiently simulated classically), the

decomposition technique would probably not lead to a quantum advantage over classical methods. If the fragment is too large (noting that the fragment size is constrained by the quantum hardware), the noise could affect the performance and the ability for quantum advantage. Examining the improvement in performance as the fragment size grows may provide an indication of potential quantum advantage. On the other hand, evaluating how noise degrades the results could reveal upper limits on the fragment size, given a quantum device with a particular fidelity level, which measures the accuracy of the device (based on how the actual quantum states produced by the device match the intended quantum states). In quantum error correction, there is a point where additional qubits can correct errors more than the noise introduced by the additional qubits. In a similar way, there may be an 'optimal point' where a larger fragment size can lead to more improved performance (for quantum advantage possibly) more than the noise introduced by the larger fragment size. Future work could discover such an optimal point.

As detailed in Appendix A Section A.10, I have thought of new ideas for future research as a result of the connection of the work on inductive bias and expressivity in Chapter 6, the work on the quantum compiler in Chapter 7,

and the work on a quantum application in Appendix A. This suggested future research includes studying the inductive bias and expressivity of the quantum circuit we used in the mining application, 'penalising' impractical solutions by restricting the expressivity of the quantum circuit, and optimising the quantum circuit with a quantum compiler.

8.4 Concluding Remarks

In conclusion, this thesis has contributed to an understanding of the foundation, optimisation, and application of particular quantum algorithms, whilst also making contributions to other fields including machine learning, classical compilation, and mining engineering. The hope is that the work in this thesis will enable the field to create quantum neural networks with the right inductive bias that will provide a quantum advantage on machine learning tasks. Furthermore, that this work will result in the optimisation of quantum algorithms so that a quantum circuit can be represented more efficiently and have a reduced runtime on simulators and fewer errors on quantum hardware, providing more accurate solutions. Additionally, that this work will help in finding additional applications of quantum algorithms, with the ultimate goal of using them for

solving real-world problems in mining engineering and other fields, such as finance, healthcare, and energy systems. Overall, the goal is that better and faster solutions on quantum computers for these computational problems will contribute to the flourishing of humankind.

A. Application of Quantum Computers

A.1 Preview 480

A.2 Summary 482

A.3 Introduction 482

A.4 Methods 488

A.5 Results 497

A.6 Conclusions 509

A.7 Comparison Between Optimisers 512

A.8 Fragmentation Details 513

A.9 Data Availability 520

A.10 Review 521

A.1 Preview

In this thesis, I have explained the key trends and milestones in quantum computing, including the development of hybrid-quantum classical algorithms. In Chapter 3, we studied the quantum neural network, which is a type of hybrid-quantum classical algorithm, and we provided a richer understanding of some of its fundamental properties in Chapter 6. In Chapter 7, we designed a quantum compiler that could optimise quantum circuits more broadly, which would help in the optimisation of quantum neural networks. Throughout this thesis, I have highlighted that one of the main goals of quantum computing is to solve real-world problems. One of the reasons for understanding the fundamental properties of quantum neural networks in Chapter 6 and improving the performance of quantum algorithms in Chapter 7 is to enable practical applications of quantum computers. In this appendix, Appendix A, we aim to address directly the ultimate goal, which is finding a new real-world application of a quantum computer, by using a variational quantum algorithm, a subset of hybrid-quantum classical algorithms, to approximately solve a real-world problem in the mining industry. We selected a particular problem, called the *ultimate pit problem*, which is solved by classical algorithms and is used com-

mercially for mining companies to maximise their value in the mining planning process. We mapped the problem to the variational quantum algorithm and ran it on classical simulators and quantum hardware. We found that the quantum algorithm was able to find good solutions to the problem. This appendix, Appendix A, presents original research on the application of variational quantum algorithms to a mining problem, which arises from the context of the development of NISQ hardware that researchers can run algorithms on and the development of hybrid quantum-classical algorithms that researchers can use to approximately solve optimisation problems. This appendix aims to find an *application* of quantum algorithms, the third conceptual layer of the framework presented in Chapter 1, Section 1.2. This appendix is based on the paper titled 'Application of the Variational Quantum Eigensolver to the Ultimate Pit Problem' [44].

I contributed to the research presented in this appendix as a co-first author; the words were mostly written by my co-authors (except Sections A.1 and A.10, which I wrote).

A.2 Summary

Determining the ultimate pit profile that maximises the profit while minimising the cost of mine extraction is a fundamental problem in open-pit mining. The development of quantum computing hardware and advances in heuristic quantum algorithms make it possible to explore quantum computing as a solution for the ultimate pit problem. Here, we cast the ultimate pit problem as a Hamiltonian ground-state search problem, which we approximately solve using the variational quantum eigensolver algorithm. Moreover, we use a domain decomposition approach to extend the reach of today's small-scale quantum hardware. We demonstrate the procedure on IBMQ devices using four qubits. To the best of our knowledge, this is the first attempt to use heuristic quantum algorithms to map small-scale instances of the ultimate pit problem on quantum hardware.

A.3 Introduction

Combinatorial optimisation problems have long been recognised as applications for a quantum computer [16, 234, 235, 236, 237]. Solving a combinatorial

optimisation problem typically means finding the global maximum of a function $C(\mathbf{z})$ of n binary variables $\mathbf{z} = (z_0 \dots z_{n-1})$, $z_i \in \{0, 1\}$. The exponential growth of the space of length-n binary strings with n makes brute-force search algorithms intractable. This limitation motivated the design of heuristic algorithms for classical computers that find local maxima of the cost function with polynomial cost, for example leveraging notions of graph theory [43, 238].

More recently, quantum algorithms were proposed as an alternative and complementary route to the solution of combinatorial optimisation problems [16, 234, 236]. It is not yet known if a quantum algorithm can provide computational advantage in a combinatorial optimisation problem, especially with the limitations of contemporary hardware [239, 152, 240, 241, 242, 243]. Designing quantum heuristics is nevertheless a valuable avenue of research, as they could be practically useful for finding near-optimal solutions.

Adiabatic state preparation (ASP) [113, 244] offers a polynomial cost solution to combinatorial optimisation problems under suitable conditions specifying a precise domain of applicability. The method often requires a circuit depth and a number of entangling gates far exceeding the capabilities of contemporary

quantum hardware. These limitations motivated the development of heuristic quantum algorithms trading exactness with more moderate use of quantum resources. Notable examples are the quantum approximate optimisation algorithm (QAOA) [16, 245], which is a variational quantum algorithm with ansatz inspired by ASP, the variational quantum eigensolver (VQE) [15, 246, 247], where the variational principle is combined with an arbitrarily flexible and hardware-aware ansatz. Other heuristic algorithms have been proposed as well [248, 249].

In this work, we explored a combinatorial optimisation problem motivated by open-pit mine design [41, 250, 251, 252, 253, 254, 255, 256, 257, 258]. The definition of a open-pit mine is "an excavation or cut made at the surface of the ground for the purpose of extracting ore and which is open to the surface for the duration of the mine's life" [259]. To expose and mine the ore, it is generally necessary to excavate and relocate large quantities of waste rock. One of the main objectives in any commercial mining operation is the exploitation of the mineral deposit at the lowest possible cost, under constraints imposed by geologic and mining engineering aspects, e.g. environmental impact and structural safety. An open-pit mining operation can be essentially viewed as a

process by which the open surface of a mine is continuously deformed, and the planning of a mining program involves the design of the final shape of this open surface. Therefore, optimising mine designs and production schedules are two fundamental problems in open-pit mining and are computationally challenging problems. In this work, we discuss a quantum computational framework for the first problem of designing optimal mines, which is known as the "ultimate pit" problem in mining literature.

We additionally note that the objective functions utilised in the ultimate pit problem have evolved over the past few decades to account for costs beyond the economic value needed to mine a specific block [260]. For example, more complex cost functions have begun accounting for the uncertainty in demand of the ore extracted from a given mine. Since ore value must account for stochastic variables such as customer demand and available supply, each of which can fluctuate and change throughout the mining optimisation process, some cost functions completely omit the economic value of a specific block. In such models, the ore value is often determined when the physical product is sold to a customer, well after the mining scheduling has been completed. More generally, more recent objective functions seek to maximise total revenue and

profit. This maximisation objective must account for costs beyond those needed to extract ore, including but not limited to the downstream transportation costs, processing costs, and more. Furthermore, each of these variables is fundamentally intertwined with the decisions of when, where, and how to mine a given complex. Therefore, the cost of mining of an individual block does not sufficiently account for the complexities and uncertainties of the total mining cost. Mathematical programming approaches that address mine production scheduling are discussed in References [261, 250, 262]. Furthermore, major mining companies have developed software for more complex optimisation schemes that involve production scheduling over multiple mines. Examples include Newmont's Nevada Optimizer [263], BHP's Blassor [264, 265] and the Whittle Consulting approach [266, 267].

In this work, we specifically focus on the simpler case of the ultimate pit problem. Following published literature in the field [41], we will assume that the type of material, its mining value and its extraction cost are given for each point of the mine, and that restrictions on the geometry of the pit are specified (surface boundaries and maximum allowable wall slopes). The combinatorial optimisation problem tackled here is to maximise the total profit

of the excavation process, defined as the total mine value of material extracted minus total extraction cost. Published literature indicates that the ultimate pit profile design problem, when formulated in three spatial dimensions and in presence of constraints dictated by the nature of the mining and processing operations, is in the NP-hard complexity class [251, 257]. NP-hardness can be proved [256] by reduction to the max clique problem. On the other hand, if the mining and processing constraints are eliminated, and if the scheduling horizon reduces to a single period, then the ultimate pit profile design problem reduces to an instance of the minimum-cut problem [268], which is solvable at polynomial cost on a classical computer [257, 258].

In the open-pit mining literature, several classical algorithms have been proposed for the ultimate pit problem, which include heuristic approximation, genetic and network flow algorithms [41, 259, 268, 256, 43, 269, 270]. One of the most widely recognised solutions to the ultimate pit problem is the Lerchs-Grossmann (LG) algorithm [41, 271], which uses a graph theoretic approach and solves the ultimate pit optimisation problem by converting it into a maximum closure problem. After the development of the LG method, other network flow algorithms were shown to compute the same results faster. Among those,

the pseudoflow algorithm [43] is one of the most widely accepted methods for the ultimate pit optimisation problem. In our work, we make a first attempt to map the ultimate pit problem to a quantum setting, and to examine the potential of quantum algorithms to deliver heuristic solutions.

A.4 Methods

A.4.1 Hamiltonian Formulation

To provide an approximate description of an open-pit mine, we model a portion of ore by a collection of finite volume elements (or blocks) with coordinates $i = (x_i^c, y_i^c)$ in 2D or $i = (x_i^c, y_i^c, z_i^c)$ in 3D (where the superscript c is indicating that the variable is a coordinate), forming a lattice L of n sites. We assume that the economic value v_i and digging cost c_i of each block is known, which in turn determine the profit $w_i = v_i - c_i$ from digging that block. To every block i we also associate a binary number z_i such that $z_i = 1$ (0) if that block is excavated (not excavated). Our goal is to maximise the profit $P(\mathbf{z}) = \sum_i w_i\, z_i$ under a constraint dictated by the open nature of the pit: every excavated block must be connected with the surface, and the walls of the pit must not

exceed a maximum steepness. We refer to these conditions as the "smoothness constraint" throughout the remainder of the present work.

In open pit mining literature, precedence constraints define the order of extraction of blocks in a pit, usually encoding the physical constraints presented by the maximum allowable slope of the earth [272]. Optimisations use them alongside other constraints to ensure safe, smooth, and reliable extraction of ore from a site. In other words, precedence constraints mathematically encode the idea that a particular block may only be extracted after the blocks above it have been extracted. Some attempts at encoding these constraints use directed graph techniques to encode the constraints of neighboring blocks [273]. Other techniques, such as diagonal domain decomposition, offer ways of encoding these relationships in the optimisation setup itself rather than explicitly as constraints.

In this work, we construct the smoothness function focusing on the example case of rectangular blocks with 45 degree angles. However, the presented approach also applies to the general case, because all precedence can be written as the smoothness function $S(\mathbf{z})$. Similarly for simplicity, the examples in

this work illustrate two-dimensional cuts, but the precedence can immediately be extended to the three-dimensional setting. To enforce the smoothness constraint we require that, if a block i is excavated, so must be all the blocks lying above it. We call these blocks the "parents" of i and denote them as $j \in P_i$. If the block i has coordinates (x_i^c, y_i^c, z_i^c), its parents have coordinates $z_j^c = z_i^c + 1$ and $x_j^c - x_i^c \leq 1$, $y_j^c - y_i^c \leq 1$. The smoothness constraint is then mathematically expressed requiring that the following smoothness function $S(\mathbf{z}) = \sum_i \sum_{j \in P_i} z_i(1 - z_j)$ vanishes. Examples of pit profiles satisfying the smoothness constraint are shown in Figure A.1: every excavated block (coloured green) is connected to the surface, and the walls of the pit (set of blocks coloured green) have slope below 45 degrees.

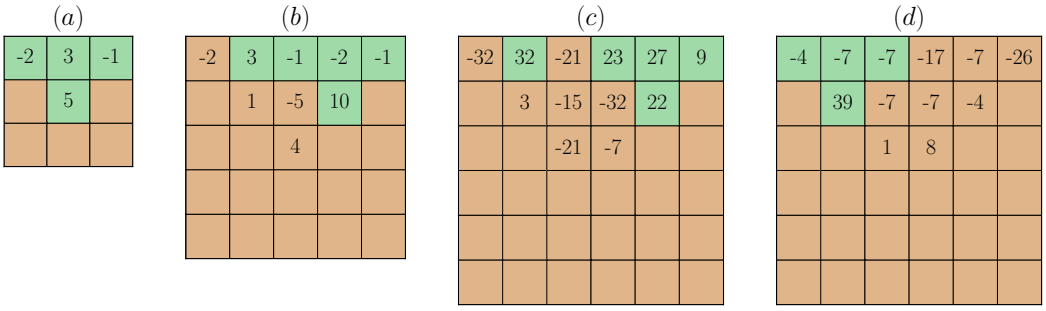


Figure A.1: **Pit profiles studied in the present work.** On each block of ore (square cells) the profit w_i is shown. Green (brown) blocks correspond to excavated (unexcavated) blocks in an optimal solution of the ultimate pit problem, and maximum profits are 5, 9, 113 and 21 respectively. The pit profiles are represented with systems of 4, 9, 12, and 12 qubits respectively.

The ultimate pit design problem is very naturally mapped onto a qubit problem. To each block i we associate a qubit, i.e. a quantum system whose states lie in a two-dimensional Hilbert space. The states of a single qubit have the form

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{A.1}$$

where α, β are complex numbers such that $|\alpha|^2 + |\beta|^2 = 1$, and the states $|0\rangle$ and $|1\rangle$ are mutually orthogonal. In this work, if a qubit is prepared in the state $|1\rangle$ (or $|0\rangle$) the corresponding block is excavated (not excavated). When a qubit is prepared in a superposition $|\psi\rangle$ of $|0\rangle$ and $|1\rangle$, the block is excavated (not excavated) with probability $|\beta|^2$ ($|\alpha|^2$). To model a pit partitioned into n blocks, we employ a register of n qubits. The states of such a register of qubits have the form

$$|\psi\rangle = \sum_{z_0 \dots z_{n-1}} c_{z_0 \dots z_{n-1}} |z_0 \dots z_{n-1}\rangle \equiv \sum_{\mathbf{z}} c_{\mathbf{z}} |\mathbf{z}\rangle \ . \tag{A.2}$$

When the register is prepared in such a state, the binary string $\mathbf{z} = (z_0 \dots z_{n-1})$

is measured with probability

$$p(\mathbf{z}) = |c_{\mathbf{z}}|^2 = |\langle \mathbf{z}|\psi\rangle|^2 \, . \tag{A.3}$$

In this study, we aim at preparing a state $|\psi\rangle$ with probability distribution peaked at bitstrings $\mathbf{z}$ that are solutions to the ultimate pit problem. To this end, we will introduce a quantum operator $\hat{H}$ whose expectation value $C = \langle\psi|\hat{H}|\psi\rangle$ over a state $|\psi\rangle$ serves as a cost function (i.e. the lower is C, the closer $|\psi\rangle$ is to an optimal solution) to be minimised in a variational simulation. Starting from the fundamental observables of a qubit, the Pauli operators

$$\begin{aligned} X &= |0\rangle\langle 1| + |1\rangle\langle 0| \, , \\ Y &= i|1\rangle\langle 0| - i|0\rangle\langle 1| \, , \\ Z &= |0\rangle\langle 0| - |1\rangle\langle 1| \, , \end{aligned} \tag{A.4}$$

we can easily map the profit and smoothness functions onto the operators

$$\hat{H}_p = \sum_i w_i \frac{1 - Z_i}{2} \, , \quad \hat{H}_s = \sum_i \sum_{j \in P_i} \frac{1 - Z_i}{2} \frac{1 + Z_j}{2} \, . \tag{A.5}$$

Here,

$$\frac{1-Z_i}{2} = |1\rangle\langle 1|_i \ , \ \frac{1+Z_j}{2} = |0\rangle\langle 0|_j \tag{A.6}$$

project the state of qubits i and j onto $|1\rangle$ and $|0\rangle$ respectively. Therefore, the action of $\hat{H}_p$ and $\hat{H}_s$ over a state $|\mathbf{z}\rangle$ reads

$$\begin{aligned} \hat{H}_p|\mathbf{z}\rangle &= \Big(\sum_i w_i z_i\Big)|\mathbf{z}\rangle = P(\mathbf{z})|\mathbf{z}\rangle \ , \\ \hat{H}_s|\mathbf{z}\rangle &= \Big(\sum_i \sum_{j\in P_i} z_i(1-z_j)\Big)|\mathbf{z}\rangle = S(\mathbf{z})|\mathbf{z}\rangle \ . \end{aligned} \tag{A.7}$$

It is worth remarking that $\hat{H}_s$ is non-negative, and its null space is the span of states $|\mathbf{z}\rangle$ such that $S(\mathbf{z}) = 0$.

In this framework, the ultimate pit design problem thus corresponds to finding binary strings $\mathbf{z}$ that lie in the ground eigenspace of $-\hat{H}_p$ and in the null space of $\hat{H}_s$ (optimal binary strings). Here, we replace this constraint problem by an unconstrained problem formed by adding a term to the objective function $-\hat{H}_p$, that consists of a penalty parameter $\gamma > 0$ multiplied by the operator $\hat{H}_s$, taken as a measure of violation of the smoothness constraint. Therefore, we search for optimal binary strings by minimising the expectation value of

the Hamiltonian $\hat{H} = -\hat{H}_p + \gamma\,\hat{H}_s$. In this work, we refer to the *negative cost function* $\hat{\tilde{H}}_p$ as $\hat{\tilde{H}}_p = \hat{H}_p + \gamma\,\hat{H}_s$ (noting that we only take the negative of $-\hat{H}_p$). We explore the use of the VQE algorithm, and we propose the following ansatz,

$$|\Psi(\theta)\rangle = \prod_i \prod_{j \in P_i} cR_y(\theta_{ij}) \prod_i R_y(\theta_i)|\Psi_0\rangle \;, \tag{A.8}$$

where a register of qubits is prepared in a tensor product $|\Psi_0\rangle$ of single-qubit states (here $|0\rangle^{\otimes n}$, $|1\rangle^{\otimes n}$ or $|+\rangle^{\otimes n}$) and manipulated with a sequence of single-qubit Y rotations ($R_y(\theta_i)$, applied to qubit i) and a sequence of controlled-Y rotations ($cR_y(\theta_{ij})$, applied to qubits i and j, where the target qubit j is a parent of the control qubit i). The ansatz (A.8) balances two needs: the desire to keep a limited circuit depth, and the desire to connect the ansatz structure with the digging process. Single-qubit Y rotations continuously change the state of a qubit between excavated and unexcavated (1 and 0 respectively), allowing exploration of various binary strings $\mathbf{z}$. Controlled-Y rotations excavate the parent j of a child i, if the qubit corresponding to i is in state 1 (excavated), and thus help enforcing smoothness constraints. Y-rotations are also chosen to maintain the wavefunction real-valued as the optimisation unfolds, thereby reducing the number of variational parameters. We remark that the choice

(A.8) is not unique, and the exploration of other ansatze is a valuable effort.

A.4.2 Domain Decomposition

As contemporary quantum devices are limited in the number and quality of qubits, they currently cannot tackle many problems of practical size directly. Variational algorithms such as VQE address the challenge posed by the quality of qubits, as they bypass the issue of quickly accumulating errors by imposing the use of shallow quantum circuits comprising a limited number of gates. A natural way to address the challenge posed by the number of qubits is to decompose the problem into a collection of subproblems, solve these subproblems on a small quantum device and combine these solutions on a classical computer to obtain a global solution [274, 275, 276, 277, 278, 279]. Here, we partition the lattice $\mathcal{L}$ into a collection $\{F_\alpha\}_\alpha$ of mutually disjoint fragments F_α, each comprising a group of blocks. Correspondingly, the Hamiltonian is written (see Section A.8) as

$$\hat{H} = \sum_\alpha \hat{V}_\alpha + \sum_{\beta \neq \alpha} \hat{W}_{\alpha,\beta} \, . \tag{A.9}$$

We focus on wavefunctions of the form $|\Psi\rangle = \otimes_\alpha |\Phi_\alpha\rangle$, where $|\Phi_\alpha\rangle$ has support on the qubits associated with fragment F_α. The expectation value of $\hat{H}$ over

a factored wavefunction is minimised when the wavefunctions $|\Phi_\alpha\rangle$ describing individual fragments satisfy a stationary Schrödinger equation of the form

$$\hat{T}_\alpha[\Phi]|\Phi_\alpha\rangle = E_\alpha|\Phi_\alpha\rangle \ , \tag{A.10}$$

$$\begin{aligned} \hat{T}_\alpha[\Phi] = \hat{V}_\alpha &+ \sum_\beta \mathrm{Tr}[\mathbb{I} \otimes |\Phi_\beta\rangle\langle\Phi_\beta|\hat{W}_{\alpha,\beta}] \\ &+ \sum_\beta \mathrm{Tr}[|\Phi_\beta\rangle\langle\Phi_\beta| \otimes \mathbb{I}\hat{W}_{\beta,\alpha}] \ , \end{aligned} \tag{A.11}$$

which we solve self-consistently.

We also note that some historical classical methods to solve the ultimate pit problem also focused on how to decompose the problem domain. For example, diagonal domain decompositions are a family of techniques for solving optimisation problems that break down large domain spaces into smaller ones for simplicity. When applied to the ultimate pit problem, they can help encode and therefore simplify precedence relationships within the mine. Instead of viewing individual blocks, diagonal domains consider diagonal slices containing multiple blocks. In the open-pit mining literature, the floating cone family of techniques stands as the most common example of diagonalisation, with

various iterations improving on the previous's ability to find the optimum pit [280, 281, 282].

A.5 Results

In this work, we consider the instances of the ultimate pit problem shown in Figure A.1. Such instances are chosen to test the proposed algorithm on small instances of different regimes of operation, due to current limitations of the quantum hardware and simulators. Profile (a) is a minimal example, where all sites are excavated. The slightly larger profile (b) illustrates the effect of the smoothness constraint (excavation of the blocks with profits $w_i = -1$). In profile (c), profit is concentrated along a diagonal $(w_i = 23, 22)$, providing a minimal mathematical representation of a stringer of valuable material, and in profile (d) the profit varies with more continuity across the pit profile.

To carry out numerical simulations, we used IBM's open-source Python library for quantum computing, Qiskit [283]. Qiskit provides tools for various tasks such as creating quantum circuits, performing simulations, and computations on real hardware. Qiskit has different types of built-in simulators, such as

QASM and statevector. Among these, the statevector simulator allows us to simulate an ideal noise-free quantum computer by keeping track of the exact wavefunction, which we utilise in this work. Currently the maximum number of qubits that we can have for the statevector simulator is 32.

Qiskit also contains an implementation of the VQE algorithm, a hybrid quantum-classical algorithm that uses both quantum and classical resources to solve the Schrödinger equation and a classical exact eigensolver algorithm to compare results. In the VQE algorithm, we take our wavefunction in the form of a quantum circuit, which is of the type described in Figure A.6. We then minimise the expectation value of the Hamiltonian with respect to the parameters of our circuit. The minimisation is carried out through the classical optimisation method L-BFGS-B [284, 285, 286] on the simulator, and Simultaneous Perturbation Stochastic Approximation (SPSA) [287, 288] on quantum hardware. L-BFGS-B is an optimisation algorithm in the family of quasi-Newton methods, that approximates the Broyden–Fletcher–Goldfarb–Shanno algorithm (BFGS) using a limited amount of computer memory. L-BFGS-B uses an adaptive approximation of the inverse Hessian matrix, which is stored in a low-rank form, to drive its search through the parameter space. The main feature of SPSA

is a gradient approximation that requires only two measurements of the cost function, regardless of the number of parameters. The gradient approximation has the form

$$\frac{\partial f}{\partial x_i}(\mathbf{x}) \simeq \frac{f(\mathbf{x}+c\mathbf{\Delta}) - f(\mathbf{x}-c\mathbf{\Delta})}{2c\Delta_i} \,, \tag{A.12}$$

where c is a constant (an optimisation hyperparameter) and $\mathbf{\Delta}$ is a random perturbation vector. The use of two measurements of the cost function, and the use of a finite number c (instead of the small numbers used in finite differences), make SPSA especially suited for simulations on quantum hardware. The performance of different optimisers is compared in Section A.7. Once the VQE is complete, we obtain the optimised variational form and the estimate for the ground state energy.

In addition, we measure the qubits in the register, obtaining probabilities for various binary strings,

$$p(\mathbf{z}) = |\langle \Psi(\theta) | \mathbf{z} \rangle|^2 \,. \tag{A.13}$$

From these, we compute the probability to extract a binary string corresponding

to an optimal solution of the ultimate pit design problem,

$$p_{\text{opt}} = \sum_{\substack{\mathbf{z}:S(\mathbf{z})=0,\\ P(\mathbf{z})=P_{\text{opt}}}} p(\mathbf{z}) \ , \quad P_{\text{opt}} = \max_{\mathbf{z}} P(\mathbf{z}) \ . \tag{A.14}$$

We performed hardware experiments on the ibmq_paris quantum device. Hardware simulations are affected by various forms of errors, arising, for example, from the unwanted interaction between qubits and their surrounding environment (decoherence), from imperfect implementation of quantum gates and measurements. To correct measurement errors, we employed readout-error mitigation [79, 289, 290] as implemented in the Qiskit Ignis library. In general, measurement errors over n qubits satisfy the relation $A\mathbf{p}_{ideal} = \mathbf{p}_{noisy}$ where $\mathbf{p}_{noisy}$ and $\mathbf{p}_{ideal}$ are vectors of probabilities (the former is returned by the noisy quantum system and the latter contains probabilities in the absence of measurement errors) and A is a $2^n \times 2^n$ stochastic matrix. To mitigate readout errors in such a scenario, one needs to execute 2^n circuits to measure $\mathbf{p}_{noisy}$ and compute A, and solve a square system of 2^n linear equations to compute $\mathbf{p}_{ideal}$ [291, 292]. However, it is often the case that errors on multiple qubits can be well approximated using at most $\mathcal{O}(n)$ calibration circuits. This result holds when: A can be approximated as a tensor product of n matrices

of shape 2×2 (tensored ansatz) [292]; A is diagonally-dominated, allowing efficient matrix-free solution of the linear system; $\mathbf{p}_{noisy}$ is well-approximated by a vector with sparsity at most $\mathcal{O}(n_s)$, where n_s is the number of accumulated statistical samples (or shots).

In the rest of this work, we call results from simulations not using readout error mitigation "raw", and otherwise "mitigated".

A.5.1 Simulations on Classical Computers

In Figure A.2 and Table A.1 we demonstrate convergence of the VQE algorithm for the three largest profiles in Figure A.1, using initial wavefunctions $|\Psi_0\rangle = |0\rangle^{\otimes n}$, $|1\rangle^{\otimes n}$ and $|+\rangle^{\otimes n}$ (unexcavated, excavated, superposition) and initial VQE parameters distributed uniformly in the interval $(-\pi/10, \pi/10)$. The optimisation is generally more challenging for profiles (c) and (d), where the larger system size and the continuous nature of the distribution of profit make it more difficult to identify and excavate valuable portions of ore. Convergence is typically more rapid and systematic for the initial state $|0\rangle^{\otimes n}$. As seen in Table A.1, when the conditions $\langle \hat{H}_s \rangle = 0$, $\langle H_p \rangle = \max_{\mathbf{z}} P(\mathbf{z})$ for successful

convergence are met, measurement of the qubits yields the optimal pit profile with probability 1.

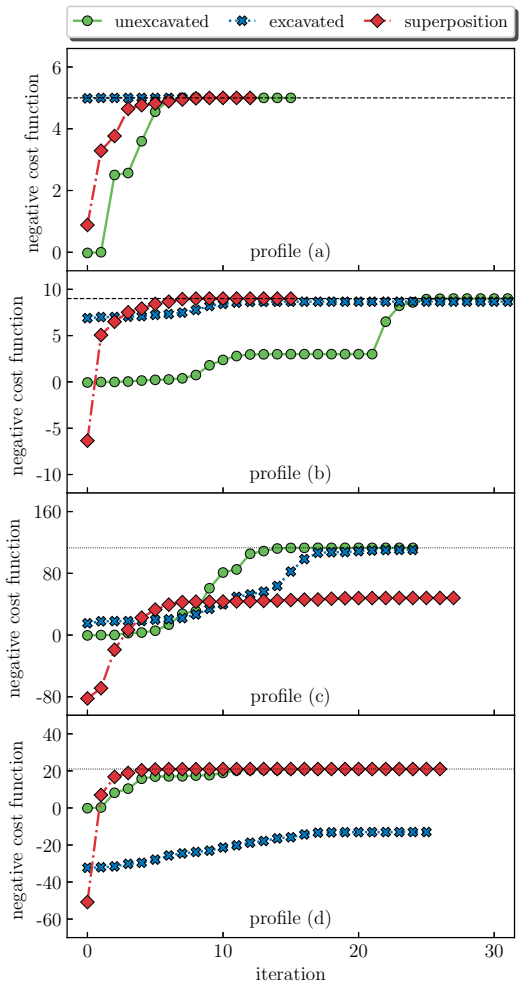


Figure A.2: **Evolution of the VQE negative cost function in the simulation of profiles in Figure A.1 (top to bottom), using different initial wavefunctions $|\Psi_0\rangle = |0\rangle^{\otimes n}$, $|1\rangle^{\otimes n}$ and $|+\rangle^{\otimes n}$ (respectively denoted unexcavated/excavated/superposition and shown in green/blue/red).** Black dashed lines denote maximum profit in optimal solutions of the problem.

profile	$\|\Psi_0\rangle$	γ	p_{opt}	profile	$\|\Psi_0\rangle$	γ	p_{opt}	profile	$\|\Psi_0\rangle$	γ	p_{opt}	profile	$\|\Psi_0\rangle$	γ	p_{opt}
a	$\|0\rangle^{\otimes n}$	7/3	1.00	b	$\|0\rangle^{\otimes n}$	10/3	1.00	c	$\|0\rangle^{\otimes n}$	53/3	1.00	d	$\|0\rangle^{\otimes n}$	22/3	1.00
a	$\|1\rangle^{\otimes n}$	7/3	1.00	b	$\|1\rangle^{\otimes n}$	10/3	1.00	c	$\|1\rangle^{\otimes n}$	53/3	1.00	d	$\|1\rangle^{\otimes n}$	22/3	0.00
a	$\|+\rangle^{\otimes n}$	7/3	1.00	b	$\|+\rangle^{\otimes n}$	10/3	1.00	c	$\|+\rangle^{\otimes n}$	53/3	0.00	d	$\|+\rangle^{\otimes n}$	22/3	1.00

Table A.1: **Penalty parameter γ and probability of detecting an optimal solution of the pit profile problem $\mathbf{p}_{\text{opt}}$ for the calculations reported in Figure A.2.** Penalty parameters were initialised to $\gamma_0 = \max_i(w_i - \sum_{j\in P_i} w_j)/3$ and then varied manually by integer multiples of 1 until maximisation of p_{opt} was achieved.

Further insight into the convergence of VQE is offered by the analysis of the expectation values of the single-qubit Pauli Z operators, $E[Z_i] = \langle\Psi(\theta)|Z_i|\Psi(\theta)\rangle$, which in turn provide the probability distribution $p(z_i = k) = (1+(-1)^k E[Z_i])/2$, $k = 0, 1$, for the measurement of a single qubit, and offer a way of visualising the pit profile as the optimisation of the VQE cost function unfolds.

In Figure A.3, we show the probabilities $p(z_i = 1)$, for the pit profile in Figure A.1b, at three steps of the optimisation. As seen, at the beginning of the optimisation, $p(z_i = 1) < 1$ for all sites. On the other hand, at the end of the optimisation, all sites have either $p(z_i = 1) = 1$ or 0, signaling convergence of VQE to an element of the computational basis $|\mathbf{z}\rangle$ corresponding to the optimal configuration in Figure A.1b.

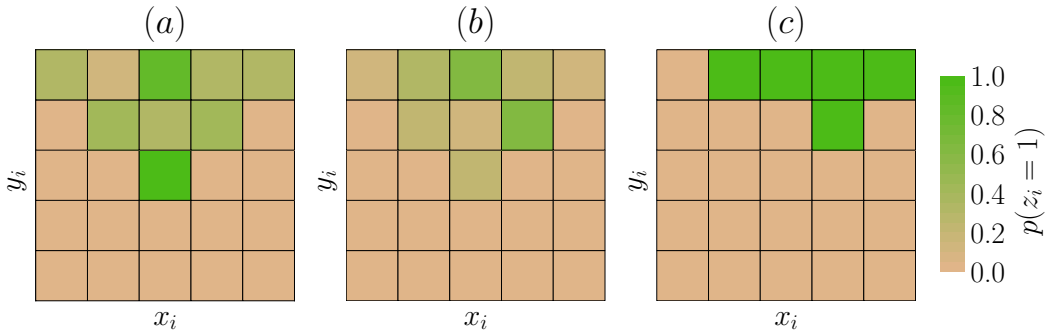


Figure A.3: **Evolution of the mining profile, i.e. the probability $\mathbf{p(z_i = 1)}$ for each site i, at three different moments (marked a,b,c) of the optimisation.** Green (brown) denote excavated (unexcavated) sites.

A.5.2 Domain Decomposition

In this Section, we explore the performance of domain decomposition. An illustrative calculation is reported in Figure A.4. We divide all pit profiles into fragments corresponding to horizontal stripes as depicted in Figure A.5. In each loop over all fragments, one iteration per fragment is performed. All fragments converge to the sum of their excavated sites in the optimal solution, thus the sum of the negative cost function of all fragments converges to the maximum possible profit of that pit profile, which is 5, 9, 113 and 21 respectively. All simulations are run using statevector simulator with L-BFGS-B optimiser, and penalties are set to 4, 5, 10 and 20 respectively. Initial states are all set to equal superposition. It is worth emphasising that the performance of fragmentation is

influenced by the choice of the partition in fragments, and on the quality of the initial collection of fragment wavefunctions $|\Phi_\alpha\rangle$. These effects are documented in Section A.8.1.

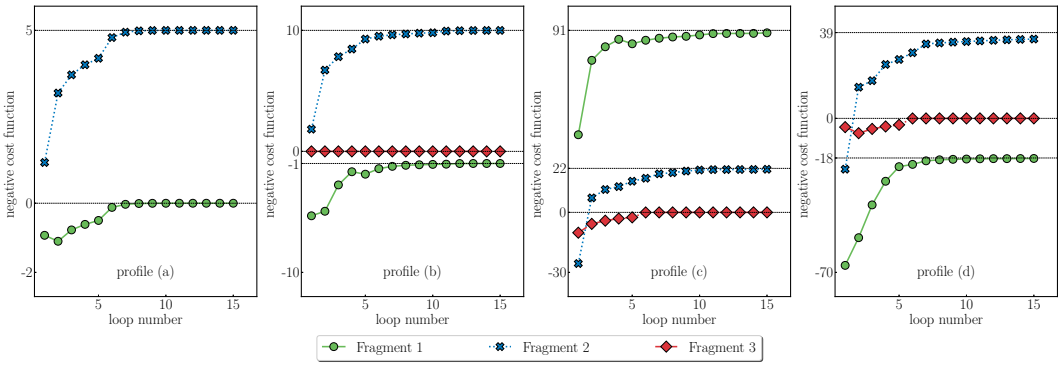


Figure A.4: **Evolution of the negative cost function of the *horizontal* fragments depicted in Figure A.5 using domain decomposition.**

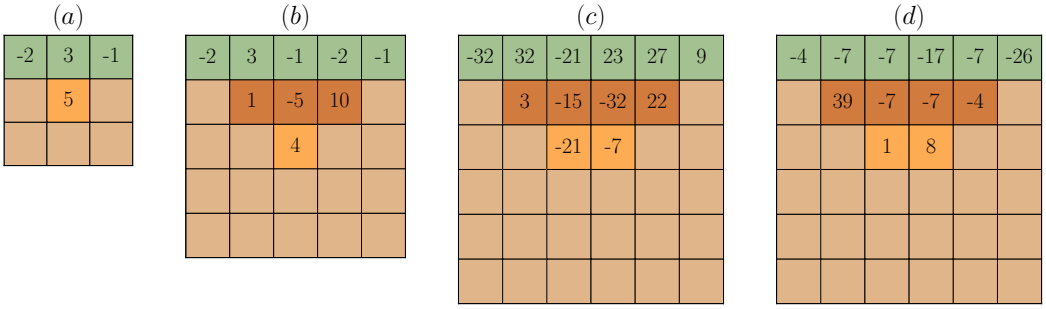


Figure A.5: ***Horizontal* cuts considered in fragmentation simulations.**

A.5.3 Hardware Experiments

We conclude by studying the profile in Figure A.1a on quantum hardware. To study this 4-site pit profile, we use the quantum circuit shown in Figure

A.6. Simulations are carried out on the ibmq_paris quantum device, using qubits [22,24,26,25] for sites 0,1,2,3 respectively. This choice is motivated by the desire of matching qubit connectivity with parent relationship, and to avoid an overhead of swap gates in the implementation of quantum gates on non-adjacent qubits.

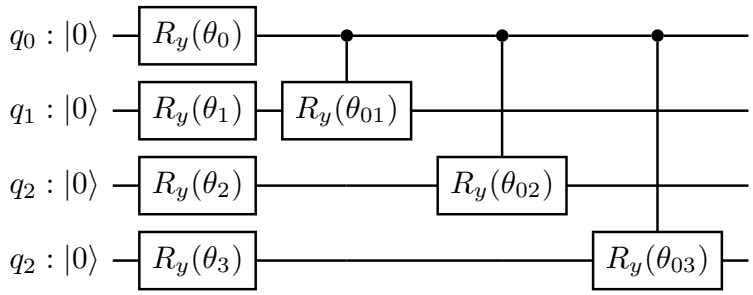


Figure A.6: **Quantum circuit used to study the 4-site pit profile shown in Figure A.1a.**

The evolution of the cost function in the VQE calculation in shown in Figure A.7. We use the SPSA optimiser, which is a stochastic optimiser particularly well-suited for simulations on quantum hardware, and employ readout error mitigation. The average negative cost function is $C_{raw} = 3.956 \pm 0.003$ and $C_{mitigated} = 4.651 \pm 0.003$, indicating that readout error mitigation reduces deviations between exact and VQE results by 66% for this particular problem.

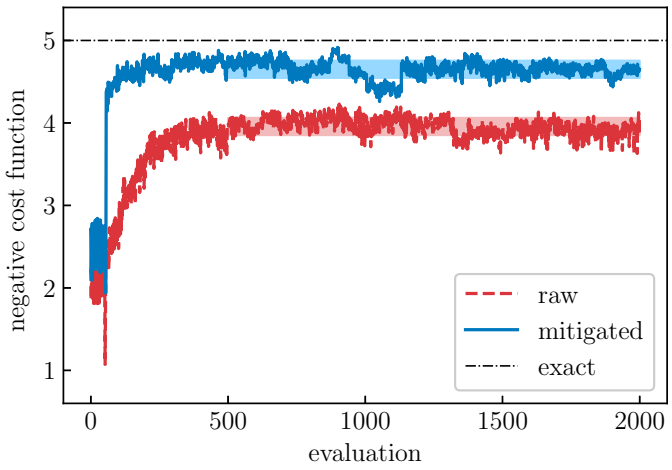


Figure A.7: **Evolution of the negative cost function during the variational optimisation.** Red, blue symbols denote raw and readout-error-mitigated results respectively, and the dashed horizontal black line denotes the optimal negative cost function value. Coloured bands denote a 67% confidence interval for the negative cost function samples after convergence of the optimisation.

In Figure A.8 we show the probability distribution for the measurement of the four qubits for the raw and error-mitigated hardware simulations, and for the exact ground state.

The probability to obtain an optimal pit profile when measuring the qubits is $p_{\text{opt}} = 0.951$, 0.996 and 1 in the three cases, and in all cases the probability of measuring a profile violating smoothness conditions is $p_{\text{v}} < 0.01$. To measure the similarity between the exact probability distribution and those arising from

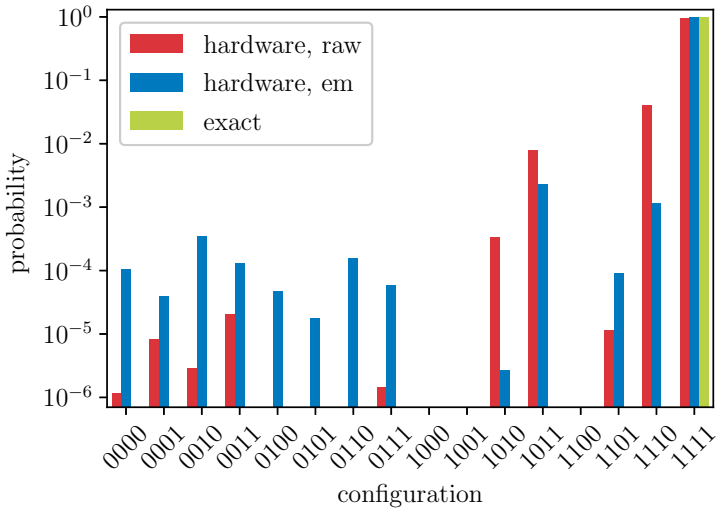


Figure A.8: **Probability distribution for the measurement of the four qubits of the register in Figure A.6 over the converged VQE wavefunction using a noiseless classical simulator of a quantum computer (green) and on quantum hardware with (blue) and without (red) readout error mitigation.** In all cases, the optimal open-pit configuration $(1, 1, 1, 1)$ is the most likely outcome of the measurement.

raw and error-mitigated simulations, we employ the Bhattacharyya distance

$$d(p, q) = -\log\left(\sum_{\mathbf{z}} \sqrt{p(\mathbf{z})q(\mathbf{z})}\right) . \tag{A.15}$$

If and only if the two distributions coincide, $p = q$, then $d(p, q) = -\log(1) = 0$.

The Bhattacharyya distances between the three probability distributions are

$$\begin{aligned} d(\text{raw}, \text{exact}) &= 0.025 \ , \\ d(\text{mitigated}, \text{exact}) &= 0.002 \ , \\ d(\text{raw}, \text{mitigated}) &= 0.027 \ , \end{aligned} \tag{A.16}$$

a further confirmation of the good quality of our simulations, and of the effectiveness of readout error mitigation techniques.

A.6 Conclusions

In this work, we proposed to tackle the computational optimisation of open-pit profiles, i.e. the ultimate pit problem, by quantum computational algorithms. We formulated the problem as a quadratic constrained binary combinatorial optimisation problem, introducing a Hamiltonian operator that comprises two terms: $\hat{H}_p$, aimed at maximising the profit of excavating a certain pit profile, and $\hat{H}_s$, aimed at enforcing continuity of the pit profile. We introduced a domain decomposition approach to simplify and accelerate the quantum computational treatment of the ultimate pit problem, which can produce results of quality comparable with an original calculation, albeit tackling a

collection of smaller and simpler problems. We discussed the limitations of such a domain decomposition approach, namely the sensitivity to initial conditions and partition into fragments. The latter difficulty can be alleviated by employing fragments of increasingly large size.

There are many quantum algorithms to approximate the ground state of a Hamiltonian, including ASP and QAOA. In this work, we elected to explore the VQE due to its generality. We introduced a variational ansatz inspired by the nature of the problem and comprising y rotations and controlled-y rotations, which proved able to yield accurate approximations for the pit profiles considered here. Despite these favorable features, the proposed ansatz is heuristic, as in typical VQE applications, and its accuracy for larger and more complex problems remains an open question. In such a situation, improved ansatze may have to be designed, on the basis of mathematical considerations and empirical data [293, 294, 295, 25], or other methods may have to be adopted. Examples include QAOA and other variational quantum algorithms [16, 293, 249, 296], quantum algorithms for evolution along a prescribed path in the Hilbert space [113, 248], as well as algorithms for long-term quantum devices [297, 298, 299, 300]. Our work also enables the exploration of alternative and complementary

research directions, such as the implementation and development of quantum-inspired algorithms for classical computers.

Our work takes a first step towards fostering the synergy between quantum computation and numerical simulations for mining applications. While the present work offers prospects for quantum algorithms to be applied to the computational design of open-pit profiles, it should be regarded as a proof-of-principle study. Limitations of the number and the quality of the qubits available on current quantum hardware restrict the problem sizes we can deal with today. Indeed, in view of the formal simplifications and the heuristic algorithms employed here, considerable methodological and experimental progress is necessary to tackle larger, more challenging and more realistic problem instances.

A systematic and comparative study of quantum algorithms is required to assess the actual relevancy of quantum computation to the present problem. Further, generalisation from two- to three-dimensional systems is also an important goal, as the dimensionality of the problem has implications on the required hardware connectivity and the cost function landscape. Moreover, improved fragmentation schemes, e.g. allowing for overlapping fragments and systematic choices

of the fragmentation choices are also important perspective improvements. Finally, penalty-free methods should be considered: these algorithms restrict the simulation to profiles with $S(\mathbf{z}) = 0$, rather than considering all possible profiles and approximately enforcing the smoothness constraint by a penalty operator $\gamma \hat{H}_s$. However, implementing a penalty-free method requires identifying, enumerating, and mapping onto qubit states the profiles with $S(\mathbf{z}) = 0$, as well as mapping the profit function onto a qubit operator and designing suitable variational ansatze. Future computational studies can represent an occasion for detailed comparison of various computational techniques, their refinement, and their extension to more realistic situations.

A.7 Comparison Between Optimisers

In this Section, we compare the performance of different optimisers using the pit profile in Figure A.1c as a test case. Experiments are run using statevector simulator with penalty $\gamma = 53/3$. All the employed optimisers (L-BFGS-B, CG, SLSQP, and SPSA) converge to the maximum profit of 113, and thus deliver the correct result. Convergence takes 806, 1333, 797 and 2051 evaluations for L-BFGS-B, CG, SLSQP, and SPSA respectively. Note that multiple evaluations

are computed in each optimisation iteration, as shown in Figure A.9.

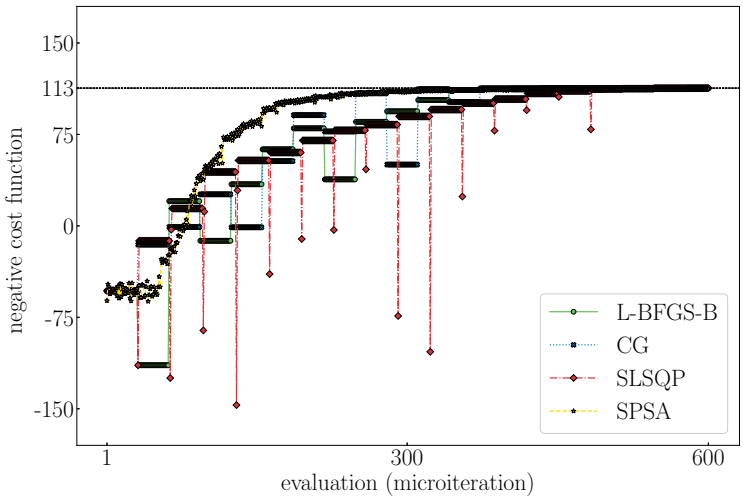


Figure A.9: **Evolution of the negative cost function for the pit profile in Figure A.1c run on a selection of different optimisers with respect to evaluation number.**

A.8 Fragmentation Details

In this Section, we derive Equations (A.9) and (A.10). Recall that the unfragmented Hamiltonian for the ultimate pit problem can be written as

$$\hat{H} = -\sum_i w_i \hat{\Pi}_i^{(1)} + \sum_{ij} \gamma \Delta(ij) \hat{\Pi}_i^{(1)} \hat{\Pi}_j^{(0)} , \tag{A.17}$$

having defined $\hat{\Pi}^{(k)} = |k\rangle\langle k|$ and

$$\Delta(i,j) = \begin{cases} 1 & \text{if } j \in P_i \\ 0 & \text{otherwise} \end{cases} \tag{A.18}$$

for brevity. Introducing a set of fragments and performing the substitution $\sum_i \to \sum_\alpha \sum_{i \in F_\alpha}$, we readily obtain

$$\begin{aligned} \hat{H} = -\sum_\alpha \sum_{i \in F_\alpha} w_i \, \hat{\Pi}_i^{(1)} + \sum_\alpha \sum_{ij \in F_\alpha} \gamma \, \Delta(ij) \, \hat{\Pi}_i^{(1)} \hat{\Pi}_j^{(0)} \\ + \sum_{\alpha \neq \beta} \sum_{\substack{i \in F_\alpha \\ j \in F_\beta}} \gamma \, \Delta(ij) \, \hat{\Pi}_i^{(1)} \hat{\Pi}_j^{(0)} \, , \end{aligned} \tag{A.19}$$

which has the form Equation (A.9) with

$$\begin{aligned} \hat{V}_\alpha = -\sum_{i \in F_\alpha} w_i \, \hat{\Pi}_i^{(1)} + \sum_{ij \in F_\alpha} \gamma \, \Delta(ij) \, \hat{\Pi}_i^{(1)} \hat{\Pi}_j^{(0)} \, , \\ \hat{W}_{\alpha\beta} = \sum_{\substack{i \in F_\alpha \\ j \in F_\beta}} \gamma \, \Delta(ij) \, \hat{\Pi}_i^{(1)} \hat{\Pi}_j^{(0)} \, . \end{aligned} \tag{A.20}$$

We have thus derived Equation (A.9). Let us now derive Equation (A.10). First, we consider the expectation value of $\hat{H}$ over a factored wavefunction,

$$E = \sum_{\alpha} \langle \Phi_\alpha | \hat{V}_\alpha | \Phi_\alpha \rangle + \sum_{\alpha \neq \beta} \langle \Phi_\alpha, \Phi_\beta | \hat{W}_{\alpha,\beta} | \Phi_\alpha, \Phi_\beta \rangle \ . \tag{A.21}$$

To ensure the Φ_α are normalised, i.e. $\langle \Phi_\alpha | \Phi_\alpha \rangle = 1$ for all α, we consider the Lagrangian $\mathcal{L} = E - \sum_\alpha E_\alpha \langle \Phi_\alpha | \Phi_\alpha \rangle$. The derivative of $\mathcal{L}$ with respect to Φ^*_ρ reads

$$\begin{aligned} \frac{\delta \mathcal{L}}{\delta \Phi^*_\rho} = \hat{V}_\rho |\Phi_\rho\rangle + \sum_{\beta \neq \rho} \langle \cdot, \Phi_\beta | \hat{W}_{\rho,\beta} | \Phi_\rho, \Phi_\beta \rangle \\ + \sum_{\alpha \neq \rho} \langle \Phi_\alpha, \cdot | \hat{W}_{\alpha,\rho} | \Phi_\alpha, \Phi_\rho \rangle - E_\rho |\Phi_\rho\rangle \end{aligned} \tag{A.22}$$

and vanishes when $\hat{T}_\rho |\Phi_\rho\rangle = E_\rho |\Phi_\rho\rangle$, with $\hat{T}_\rho$ as in Equation (A.10).

As an example consider the 4-block mine in Figure A.10, partitioned in 2 horizontal fragments (indicated with green and orange).

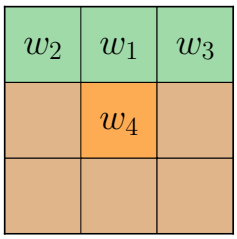


Figure A.10: **Fragmented pit profile.**

The upper fragment has the Hamiltonian

$$\begin{aligned} \mathrm{H}_1 = \frac{1}{2}\Big[&(w_1 Z_1 + w_2 Z_2 + w_3 Z_3 + w_4 \langle Z_4 \rangle) \\ &- (w_1 + w_2 + w_3 + w_4)\Big] \\ &+ \frac{\gamma}{4}\Big[(I + Z_1 - \langle Z_4 \rangle - Z_1 \langle Z_4 \rangle) \\ &+ (I + Z_2 - \langle Z_4 \rangle - Z_2 \langle Z_4 \rangle) \\ &+ (I + Z_3 - \langle Z_4 \rangle - Z_3 \langle Z_4 \rangle)\Big] \end{aligned} \tag{A.23}$$

and the lower one is described by the Hamiltonian

$$\begin{aligned} \mathrm{H}_2 = \frac{1}{2}\Big[&(w_1 \langle Z_1 \rangle + w_2 \langle Z_2 \rangle + w_3 \langle Z_3 \rangle + w_4 Z_4) \\ &- (w_1 + w_2 + w_3 + w_4)\Big] \\ &+ \frac{\gamma}{4}\Big[(I + \langle Z_1 \rangle - Z_4 - \langle Z_1 \rangle Z_4) \\ &+ (I + \langle Z_2 \rangle - Z_4 - \langle Z_2 \rangle Z_4) \\ &+ (I + \langle Z_3 \rangle - Z_4 - \langle Z_3 \rangle Z_4)\Big]. \end{aligned} \tag{A.24}$$

Thus if a particular pair of blocks contributing to the smoothness Hamiltonian is severed by fragmentation, the block outside the fragment of consideration still contributes to the smoothness Hamiltonian with its expected value.

A.8.1 Comparison Between Fragmentation Strategies

There are combinatorially many ways of partitioning a domain into fragments, and the choice of a fragmentation scheme can drastically affect the quality of final results. To assess the sensitivity of VQE calculations to the fragmentation, we compare the fragmentation strategies in Figures A.5 and A.12. Results for the fragmentation strategy based on horizontal cuts (see Figure A.5) are shown in the main text, and for the fragmentation strategy based on other cuts

(see Figure A.12) are shown in Figure A.11.

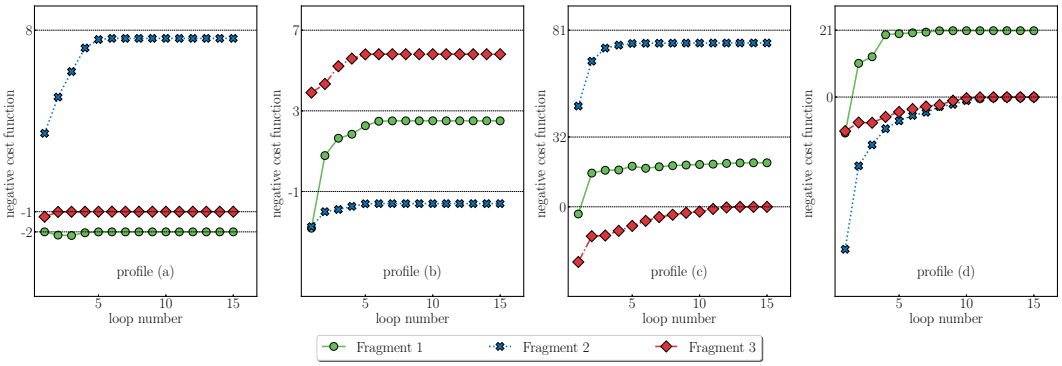


Figure A.11: **Evolution of the negative cost function of the *other* fragments depicted in Figure A.12 using the domain decomposition procedure.**

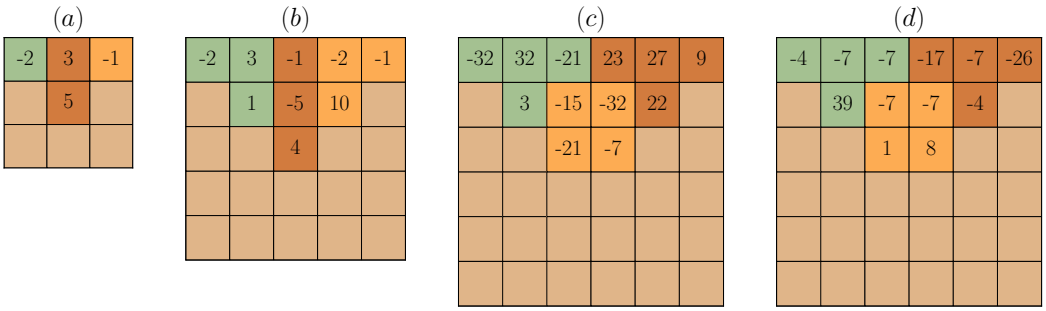


Figure A.12: ***Other* cuts considered in fragmentation simulations.**

All simulations are run using statevector simulator with L-BFGS-B optimiser, and penalties are set to 4, 5, 10 and 20 respectively. Initial states are all set to equal superposition. In each loop over all fragments, one iteration per fragment is performed. All fragments converge to the sum of their excavated sites in the optimal solution, thus the sum of the negative cost function of all fragments

converges to the maximum possible profit of that pit profile, which is 5, 9, 113 and 21 respectively.

As seen, we obtain the correct solution with ways of cutting the fragments other than horizontal fragmentation. The probability of finding the correct bit string is generally lower than using horizontal simulations, however the correct bit string still has higher probability than all other bit strings in our simulations. The reason for the decrease in optimal probability is that when the bond between parent-child pairs is preserved within a fragment, we have an additional controlled Y-rotation gate preserved in the ansatz circuit, meaning there are more parameters to be optimised over. Thus horizontal fragmentation proves to be the most reliable but has the disadvantage that it will scale worse than an arbitrary, constant-sized way of cutting as the pit size increases. Also note that the equal superposition state proves to be the most consistent choice of initial states across different profiles. This result is presented in Table A.2. That is the reason why we present fragmentation results with initial states as equal superposition and with horizontal cuts in the main section of the present work. Another technical remark is that during domain decomposition optimisation, we have experimentally observed that the ansatz parameters

can get stuck at the boundaries of their allowed range of $[0, \pi]$. If parameters get very close to either 0 or π, we introduce temperature to the optimisation procedure by randomly shifting these parameters away from the boundary. For fragments where the parent-child pairs have been preserved, we have also observed that it is rather beneficial to constrain the sum of the parameters of the single and controlled Y-rotation gates acting on a particular qubit to the said interval than constraining parameters individually.

profile	$\|\Psi_0\rangle$	cut	γ	p_{opt}	optimal	profile	$\|\Psi_0\rangle$	cut	γ	p_{opt}	optimal
a	$\|0\rangle^{\otimes n}$	h	4	0.00	no	c	$\|0\rangle^{\otimes n}$	h	10	0.93	yes
a	$\|1\rangle^{\otimes n}$	h	4	**1.00**	yes	c	$\|1\rangle^{\otimes n}$	h	10	0.92	yes
a	$\|+\rangle^{\otimes n}$	h	4	**1.00**	yes	c	$\|+\rangle^{\otimes n}$	h	10	**0.95**	yes
a	$\|+\rangle^{\otimes n}$	o	4	0.92	yes	c	$\|+\rangle^{\otimes n}$	o	10	0.49	yes
b	$\|0\rangle^{\otimes n}$	h	5	0.00	no	d	$\|0\rangle^{\otimes n}$	h	20	0.00	no
b	$\|1\rangle^{\otimes n}$	h	5	0.00	no	d	$\|1\rangle^{\otimes n}$	h	20	0.92	yes
b	$\|+\rangle^{\otimes n}$	h	5	**1.00**	yes	d	$\|+\rangle^{\otimes n}$	h	20	0.92	yes
b	$\|+\rangle^{\otimes n}$	o	5	0.65	yes	d	$\|+\rangle^{\otimes n}$	o	20	**0.98**	yes

Table A.2: **Penalty parameter γ and probability of detecting an optimal solution $\mathbf{p}_{\text{opt}}$, for the calculations in Figure A.4 and A.11.** "h": *horizontal* cut, "o": *other* cut.

A.9 Data Availability

The code used to generate the data presented in this appendix can be publicly accessed on GitHub at [301].

A.10 Review

Our goal for this appendix was to apply a quantum variational algorithm to a real-world problem; we were able to achieve this goal by focusing on a problem in the mining industry called the ultimate pit problem. We formulated the problem as a Hamiltonian and constructed a carefully-designed quantum ansatz to solve the problem. We also decomposed the Hamiltonian so larger-scale instances of the problem could be solved beyond the limits of the quantum hardware. This research provided a quantum algorithm that could solve a tangible, practical, commercial real-world problem.

Our work is a proof-of-concept and can not be applied commercially yet as our work applies to small-scale instances of the problem, whereas commercial solutions need to be applied on larger-scale instances. In addition, our work makes some assumptions to simplify the problem to be run on a quantum computer (e.g. the simple sloping constraints), whereas commercial solutions may need to make more complex constraints and fewer assumptions. Furthermore, there is no guarantee of a quantum speedup for this algorithm, as there is no known general quantum speedups for variational quantum algorithms, which

our algorithm is based on. If a generic quantum speedup for variational quantum algorithms could be found in the future, then this work could potentially provide a quantum advantage.

Based on the work in Chapter 6, future work could involve studying the inductive bias and expressivity of the quantum circuit we used in the work in this appendix, Appendix A, which would help us to further understand the properties of the quantum ansatz and why the algorithm was able to find a good solution. Furthermore, with a better understanding of expressivity from Chapter 6, future work could look into 'penalising' impractical solutions by constructing quantum circuits that do not even express that solution. In our work, we 'penalised' impractical solutions, such as those that did not meet the digging slope constraints, by adding a penalising term to the Hamiltonian; an alternative or addition to this could include creating a quantum circuit that is unable to express such solutions. Based on the work in Chapter 7, we could use the quantum compiler to see if it could optimise the quantum circuits in this appendix, Appendix A, further and whether that affects its performance.



References

[1] P. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, Nov. 1994, pp. 124–134. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/365700

[2] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of Computing*, ser. STOC '96. New York, NY, USA: Association for Computing Machinery, Jul. 1996, pp. 212–219. [Online]. Available: https://dl.acm.org/doi/10.1145/237814.237866

[3] A. Bouland, W. van Dam, H. Joorati, I. Kerenidis, and A. Prakash, "Prospects and challenges of quantum finance," Nov. 2020, arXiv:2011.06492 [quant-ph, q-fin]. [Online]. Available: http://arxiv.org/abs/2011.06492

[4] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, no. 7671, pp. 195–202, Sep. 2017, number: 7671 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/nature23474

[5] Y. Cao, J. Romero, J. P. Olson, M. Degroote, P. D. Johnson, M. Kieferová, I. D. Kivlichan, T. Menke, B. Peropadre, N. P. D. Sawaya, S. Sim, L. Veis, and A. Aspuru-Guzik, "Quantum Chemistry in the Age of Quantum Computing," *Chemical Reviews*, vol. 119, no. 19, pp. 10 856–10 915, Oct. 2019, publisher: American Chemical Society. [Online]. Available: https://doi.org/10.1021/acs.chemrev.8b00803

[6] K. Igeta and Y. Yamamoto, "Quantum mechanical computers with single atom and photon fields," in *International Conference on Quantum*

Electronics (1988), paper TuI4. Optica Publishing Group, Jul. 1988, p. TuI4. [Online]. Available: https://opg.optica.org/abstract.cfm?uri=IQEC-1988-TuI4

[7] J. I. Cirac and P. Zoller, "Quantum Computations with Cold Trapped Ions," *Physical Review Letters*, vol. 74, no. 20, pp. 4091–4094, May 1995, publisher: American Physical Society. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.74.4091

[8] Y. Nakamura, Y. A. Pashkin, and J. S. Tsai, "Coherent control of macroscopic quantum states in a single-Cooper-pair box," *Nature*, vol. 398, no. 6730, pp. 786–788, Apr. 1999, publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/19718

[9] M. V. G. Dutt, L. Childress, L. Jiang, E. Togan, J. Maze, F. Jelezko, A. S. Zibrov, P. R. Hemmer, and M. D. Lukin, "Quantum Register Based on Individual Electronic and Nuclear Spin Qubits in Diamond," *Science*, vol. 316, no. 5829, pp. 1312–1316, Jun. 2007, publisher: American Association for the Advancement of Science. [Online]. Available:

https://www.science.org/doi/10.1126/science.1139831

[10] I. L. Chuang, N. Gershenfeld, and M. Kubinec, "Experimental Implementation of Fast Quantum Searching," *Physical Review Letters*, vol. 80, no. 15, pp. 3408–3411, Apr. 1998, publisher: American Physical Society. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.80.3408

[11] "Five years ago today, we put the first quantum computer on the cloud. Here's how we did it. | IBM Quantum Computing Blog." [Online]. Available: https://www.ibm.com/quantum/blog/quantum-five-years

[12] "Qiskit," 2017. [Online]. Available: https://qiskit.org/

[13] S. Sivarajah, S. Dilkes, A. Cowtan, W. Simmons, A. Edgington, and R. Duncan, "t|ket⟩: a retargetable compiler for NISQ devices," *Quantum Science and Technology*, vol. 6, no. 1, p. 014003, Nov. 2020, publisher: IOP Publishing. [Online]. Available: https://dx.doi.org/10.1088/2058-9565/ab8e92

[14] J. Preskill, "Quantum Computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, Aug. 2018, publisher: Verein zur Förderung des Open Access Publizierens in den Quantenwissenschaften. [Online]. Available: https://quantum-journal.org/papers/q-2018-08-06-79/

[15] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, "A variational eigenvalue solver on a photonic quantum processor," *Nature Communications*, vol. 5, no. 1, p. 4213, Jul. 2014, number: 1 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/ncomms5213

[16] E. Farhi, J. Goldstone, and S. Gutmann, "A Quantum Approximate Optimization Algorithm," Nov. 2014, arXiv:1411.4028 [quant-ph]. [Online]. Available: http://arxiv.org/abs/1411.4028

[17] S. Khatri, R. LaRose, A. Poremba, L. Cincio, A. T. Sornborger, and P. J. Coles, "Quantum-assisted quantum compiling," *Quantum*, vol. 3, p. 140, May 2019, publisher: Verein zur Förderung des Open Access Publizierens in den Quantenwissenschaften. [Online]. Available:

https://quantum-journal.org/papers/q-2019-05-13-140/

[18] X. Xu, J. Sun, S. Endo, Y. Li, S. C. Benjamin, and X. Yuan, "Variational algorithms for linear algebra," *Science Bulletin*, vol. 66, no. 21, pp. 2181–2188, Nov. 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2095927321004631

[19] E. Farhi and H. Neven, "Classification with Quantum Neural Networks on Near Term Processors," arXiv, Tech. Rep. arXiv:1802.06002, Aug. 2018, arXiv:1802.06002 [quant-ph] type: article. [Online]. Available: http://arxiv.org/abs/1802.06002

[20] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, and P. J. Coles, "Variational quantum algorithms," *Nature Reviews Physics*, vol. 3, no. 9, pp. 625–644, Sep. 2021, number: 9 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s42254-021-00348-9

[21] J. Chow, E. Greplová, F. Heijman, C. Kuchkovsky, D. O'Halloran, J. Pointing, G. Shutko, and C. Williams, "State of Quantum Computing: Building a Quantum Economy," Tech. Rep., Sep. 2022. [Online]. Available: https://www.weforum.org/publications/state-of-quantum-computing-building-a-quantum-economy/

[22] A. W. Harrow, A. Hassidim, and S. Lloyd, "Quantum Algorithm for Linear Systems of Equations," *Physical Review Letters*, vol. 103, no. 15, p. 150502, Oct. 2009, publisher: American Physical Society. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.103.150502

[23] S. C. Kak, "Quantum Neural Computing," in *Advances in Imaging and Electron Physics*, P. W. Hawkes, Ed. Elsevier, Jan. 1995, vol. 94, pp. 259–313. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1076567008701472

[24] M. Broughton, G. Verdon, T. McCourt, A. J. Martinez, J. H. Yoo, S. V. Isakov, P. Massey, R. Halavati, M. Y. Niu, A. Zlokapa, E. Peters, O. Lockwood, A. Skolik, S. Jerbi, V. Dunjko, M. Leib, M. Streif,

D. Von Dollen, H. Chen, S. Cao, R. Wiersema, H.-Y. Huang, J. R. McClean, R. Babbush, S. Boixo, D. Bacon, A. K. Ho, H. Neven, and M. Mohseni, "TensorFlow Quantum: A Software Framework for Quantum Machine Learning," Aug. 2021, arXiv:2003.02989 [cond-mat, physics:quant-ph]. [Online]. Available: http://arxiv.org/abs/2003.02989

[25] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, "Barren plateaus in quantum neural network training landscapes," *Nature Communications*, vol. 9, no. 1, p. 4812, Nov. 2018, number: 1 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41467-018-07090-4

[26] M. Schuld and N. Killoran, "Quantum Machine Learning in Feature Hilbert Spaces," *Physical Review Letters*, vol. 122, no. 4, p. 040504, Feb. 2019, publisher: American Physical Society. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.122.040504

[27] Y. Liu, S. Arunachalam, and K. Temme, "A rigorous and robust quantum speed-up in supervised machine learning," *Nature*

Physics, vol. 17, no. 9, pp. 1013–1017, Sep. 2021, number: 9 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41567-021-01287-z

[28] H.-Y. Huang, M. Broughton, M. Mohseni, R. Babbush, S. Boixo, H. Neven, and J. R. McClean, "Power of data in quantum machine learning," *Nature Communications*, vol. 12, no. 1, p. 2631, May 2021, number: 1 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41467-021-22539-9

[29] A. Abbas, D. Sutter, and S. Wörner, "The power of quantum neural networks," Jul. 2021. [Online]. Available: https://research.ibm.com/blog/quantum-neural-network-power

[30] C. Mingard, J. Pointing, C. London, Y. Nam, and A. A. Louis, "Exploiting the equivalence between quantum neural networks and perceptrons," Jul. 2024, arXiv:2407.04371 [quant-ph]. [Online]. Available: http://arxiv.org/abs/2407.04371

[31] M. Schuld and N. Killoran, "Is Quantum Advantage the Right Goal for Quantum Machine Learning?" *PRX Quantum*, vol. 3, no. 3, p. 030101, Jul. 2022. [Online]. Available: https://link.aps.org/doi/10.1103/PRXQuantum.3.030101

[32] K. Hu and K. Hu, "ChatGPT sets record for fastest-growing user base - analyst note," *Reuters*, Feb. 2023. [Online]. Available: https://www.reuters.com/technology/chatgpt-sets-record-fastest-growing-user-base-analyst-note-2023-02-01/

[33] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," Nov. 2016. [Online]. Available: https://openreview.net/forum?id=Sy8gdB9xx

[34] G. Valle-Pérez, C. Q. Camargo, and A. A. Louis, "Deep learning generalizes because the parameter-function map is biased towards simple functions," arXiv, Tech. Rep. arXiv:1805.08522, Apr. 2019, arXiv:1805.08522 [cs, stat] type: article. [Online]. Available: http://arxiv.org/abs/1805.08522

[35] K. Dingle, C. Q. Camargo, and A. A. Louis, "Input–output maps are strongly biased towards simple outputs," *Nature Communications*, vol. 9, no. 1, p. 761, Dec. 2018. [Online]. Available: http://www.nature.com/articles/s41467-018-03101-6

[36] J. Pointing, "Do Quantum Neural Networks have Simplicity Bias?" Jul. 2024, arXiv:2407.03266 [physics, physics:quant-ph]. [Online]. Available: http://arxiv.org/abs/2407.03266

[37] J. Kübler, S. Buchholz, and B. Schölkopf, "The Inductive Bias of Quantum Kernels," in *Advances in Neural Information Processing Systems*, vol. 34. Curran Associates, Inc., 2021, pp. 12 661–12 673. [Online]. Available: https://proceedings.neurips.cc/paper/2021/hash/69adc1e107f7f7d035d7baf04342e1ca-Abstract.html

[38] M. Schuld and F. Petruccione, *Machine Learning with Quantum Computers*, ser. Quantum Science and Technology. Cham: Springer International Publishing, 2021. [Online]. Available: https://link.springer.com/10.1007/978-3-030-83098-4

[39] Z. Jia, O. Padon, J. Thomas, T. Warszawski, M. Zaharia, and A. Aiken, "TASO: optimizing deep learning computation with automatic generation of graph substitutions," in *Proceedings of the 27th ACM Symposium on Operating Systems Principles*. Huntsville Ontario Canada: ACM, Oct. 2019, pp. 47–62. [Online]. Available: https://dl.acm.org/doi/10.1145/3341301.3359630

[40] J. Pointing, O. Padon, Z. Jia, H. Ma, A. Hirth, J. Palsberg, and A. Aiken, "Quanto: optimizing quantum circuits with automatic generation of circuit identities," *Quantum Science and Technology*, vol. 9, no. 4, p. 045009, Jul. 2024, publisher: IOP Publishing. [Online]. Available: https://dx.doi.org/10.1088/2058-9565/ad5b16

[41] H. Lerchs, "Optimum design of open-pit mines," *Trans CIM*, vol. 68, pp. 17–24, 1965.

[42] J. Poniewierski, "Pseudoflow Explained," Tech. Rep., Oct. 2020. [Online]. Available: https://www.deswik.com/news/pseudoflow-explained/

[43] D. S. Hochbaum, "The Pseudoflow Algorithm: A New Algorithm for the Maximum-Flow Problem," *Operations Research*, vol. 56, no. 4, pp. 992–1009, Jul. 2008. [Online]. Available: https://doi.org/10.1287/opre.1080.0524

[44] Y. Hindy, J. Pointing, M. Tolunay, S. Venkatarao, M. Motta, and J. A. Latone, "Application of the Variational Quantum Eigensolver to the Ultimate Pit Problem | IEEE Conference Publication | IEEE Xplore." Bellevue, WA, USA: IEEE, Nov. 2023. [Online]. Available: https://doi.org/10.1109/QCE57702.2023.00083

[45] M. Treinish, J. Gambetta, S. Thomas, P. Nation, qiskit bot, P. Kassebaum, D. M. Rodríguez, S. d. l. P. González, E. Arellano, J. Lishman, L. Bello, S. Hu, J. Garrison, J. Huang, K. Krsulich, J. Yu, J. Gacon, M. Marques, D. McKay, J. Gomez, L. Capelluto, S. Wood, Travis-S-IBM, A. Mitchell, A. Panigrahi, K. Hartman, lerongil, R. I. Rahman, T. Itoko, and A. Pozas-Kerstjens, "Qiskit/qiskit-metapackage: Qiskit 0.43.1," Jun. 2023. [Online]. Available: https://zenodo.org/record/8003781

[46] "Hydra, Theoretical Physics' compute cluster | University of Oxford Department of Physics." [Online]. Available: https://www2.physics.ox.ac.uk/it-services/computing-at-theoretical-physics/compute-cluster

[47] "IBM Quantum." [Online]. Available: https://quantum-computing.ibm.com/

[48] A. M. Turing, "On computable numbers, with an application to the Entscheidungsproblem," *J. of Math*, vol. 58, no. 345-363, p. 5, 1936. [Online]. Available: http://www.wolframscience.com/prizes/tm23/images/Turing.pdf

[49] A. Einstein, "Über einem die Erzeugung und Verwandlung des Lichtes betreffenden heuristischen Gesichtspunkt," *Annalen der physik*, vol. 4, 1905. [Online]. Available: http://sedici.unlp.edu.ar/handle/10915/2784

[50] M. Born, "Über Quantenmechanik," *Zeitschrift für Physik*, vol. 26, no. 1, pp. 379–395, Dec. 1924. [Online]. Available: https://doi.org/10.1007/BF01327341

[51] L. De Broglie, "Recherches sur la théorie des quanta," PhD Thesis, Migration-université en cours d'affectation, 1924. [Online]. Available: https://theses.hal.science/tel-00006807/document

[52] W. Heisenberg, "Über den anschaulichen Inhalt der quantentheoretischen Kinematik und Mechanik," *Zeitschrift für Physik*, vol. 43, no. 3, pp. 172–198, Mar. 1927. [Online]. Available: https://doi.org/10.1007/BF01397280

[53] R. S. Ingarden, "Quantum information theory," *Reports on Mathematical Physics*, vol. 10, no. 1, pp. 43–72, Aug. 1976. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0034487776900057

[54] P. Benioff, "The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines," *Journal of Statistical Physics*, vol. 22, no. 5, pp. 563–591, May 1980. [Online]. Available: https://doi.org/10.1007/BF01011339

[55] D. Deutsch and R. Penrose, "Quantum theory, the Church–Turing principle and the universal quantum computer," *Proceedings of the Royal*

Society of London. A. Mathematical and Physical Sciences, vol. 400, no. 1818, pp. 97–117, Jul. 1985, publisher: Royal Society. [Online]. Available: https://royalsocietypublishing.org/doi/abs/10.1098/rspa.1985.0070

[56] R. P. Feynman, "Simulating physics with computers," *International Journal of Theoretical Physics*, vol. 21, no. 6, pp. 467–488, Jun. 1982. [Online]. Available: https://doi.org/10.1007/BF02650179

[57] D. Deutsch and R. Jozsa, "Rapid solution of problems by quantum computation," *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, vol. 439, no. 1907, pp. 553–558, Dec. 1992, publisher: Royal Society. [Online]. Available: https://royalsocietypublishing.org/doi/abs/10.1098/rspa.1992.0167

[58] C. H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, and W. K. Wootters, "Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels," *Physical Review Letters*, vol. 70, no. 13, pp. 1895–1899, Mar. 1993, publisher: American Physical Society. [Online]. Available: https:

//link.aps.org/doi/10.1103/PhysRevLett.70.1895

[59] D. Simon, "On the Power of Quantum Computation," *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1474–1483, Oct. 1997. [Online]. Available: https://epubs.siam.org/doi/10.1137/S0097539796298637

[60] A. Ballon, "Trapped ion quantum computers," *PennyLane Demos*, Nov. 2021, publisher: Xanadu. [Online]. Available: https://pennylane.ai/qml/demos/tutorial_trapped_ions/

[61] "What is Trapped Ions." [Online]. Available: https://www.quera.com/glossary/trapped-ions

[62] I. Georgescu, "Trapped ion quantum computing turns 25," *Nature Reviews Physics*, vol. 2, no. 6, pp. 278–278, Jun. 2020, publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s42254-020-0189-1

[63] R. Davis, "How The First Superconducting Qubit Changed Quantum

Computing Forever," Sep. 2022. [Online]. Available: https://medium.com/qiskit/how-the-first-superconducting-qubit-changed-quantum-computing-forever-96cf261b8498

[64] D. P. DiVincenzo, "The Physical Implementation of Quantum Computation," *Fortschritte der Physik*, vol. 48, no. 9-11, pp. 771–783, 2000. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/1521-3978%28200009%2948%3A9/11%3C771%3A%3AAID-PROP771%3E3.0.CO%3B2-E

[65] C. S. Hamilton, R. Kruse, L. Sansoni, S. Barkhofen, C. Silberhorn, and I. Jex, "Gaussian Boson Sampling," *Physical Review Letters*, vol. 119, no. 17, p. 170501, Oct. 2017, publisher: American Physical Society. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.119.170501

[66] S. Aaronson and A. Arkhipov, "The computational complexity of linear optics," in *Proceedings of the forty-third annual ACM symposium on Theory of computing*, ser. STOC '11. New York, NY, USA: Association

for Computing Machinery, Jun. 2011, pp. 333–342. [Online]. Available: https://dl.acm.org/doi/10.1145/1993636.1993682

[67] A. Y. Kitaev, "Fault-tolerant quantum computation by anyons," *Annals of Physics*, vol. 303, no. 1, pp. 2–30, Jan. 2003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0003491602000180

[68] E. Gibney, "Inside Microsoft's quest for a topological quantum computer," *Nature*, Oct. 2016, publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/nature.2016.20774

[69] J. Langston, "In a historic milestone, Azure Quantum demonstrates formerly elusive physics needed to build scalable topological qubits," Mar. 2022. [Online]. Available: https://news.microsoft.com/source/features/innovation/azure-quantum-majorana-topological-qubit/

[70] C. Nayak, "Microsoft has demonstrated the underlying physics required to create a new kind of qubit," Mar. 2022. [Online]. Available: https://www.microsoft.com/en-us/research/blog/microsoft-has-demon

strated-the-underlying-physics-required-to-create-a-new-kind-of-qubit/

[71] Morteza Aghaee and et al., "Phys. Rev. B 107, 245423 (2023) - InAs-Al hybrid devices passing the topological gap protocol," Jun. 2023. [Online]. Available: https://journals.aps.org/prb/abstract/10.1103/PhysRevB.107.245423

[72] "IBM Makes Quantum Computing Available on IBM Cloud to Accelerate Innovation," May 2016. [Online]. Available: https://uk.newsroom.ibm.com/2016-May-04-IBM-Makes-Quantum-Computing-Available-on-IBM-Cloud-to-Accelerate-Innovation

[73] C. Rigetti, "Introducing Rigetti Quantum Cloud Services," Sep. 2018. [Online]. Available: https://medium.com/rigetti/introducing-rigetti-quantum-cloud-services-c6005729768c

[74] E. Lucero, "Unveiling our new Quantum AI campus," May 2021. [Online]. Available: https://blog.google/technology/ai/unveiling-our-new-quantum-ai-campus/

[75] "Quantum computing is now available on AWS through Amazon Braket," Aug. 2020. [Online]. Available: https://aws.amazon.com/about-aws/whats-new/2020/08/quantum-computing-available-aws-through-amazon-braket/

[76] F. Lardinois, "Microsoft's Azure Quantum platform is now in public preview," Feb. 2021. [Online]. Available: https://techcrunch.com/2021/02/01/microsofts-azure-quantum-platform-is-now-in-public-preview/

[77] Qiskit, "The Quantum Convolution Neural Network — Qiskit Machine Learning 0.6.0 documentation." [Online]. Available: https://qiskit.org/documentation/machine-learning/tutorials/11_quantum_convolutional_neural_networks.html

[78] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, S. Ahmed, V. Ajith, M. S. Alam, G. Alonso-Linaje, B. AkashNarayanan, A. Asadi, J. M. Arrazola, U. Azad, S. Banning, C. Blank, T. R. Bromley, B. A. Cordier, J. Ceroni, A. Delgado, O. Di Matteo, A. Dusko, T. Garg, D. Guala, A. Hayes, R. Hill, A. Ijaz, T. Isacsson, D. Ittah, S. Jahangiri,

P. Jain, E. Jiang, A. Khandelwal, K. Kottmann, R. A. Lang, C. Lee, T. Loke, A. Lowe, K. McKiernan, J. J. Meyer, J. A. Montañez-Barrera, R. Moyard, Z. Niu, L. J. O'Riordan, S. Oud, A. Panigrahi, C.-Y. Park, D. Polatajko, N. Quesada, C. Roberts, N. Sá, I. Schoch, B. Shi, S. Shu, S. Sim, A. Singh, I. Strandberg, J. Soni, A. Száva, S. Thabet, R. A. Vargas-Hernández, T. Vincent, N. Vitucci, M. Weber, D. Wierichs, R. Wiersema, M. Willmann, V. Wong, S. Zhang, and N. Killoran, "PennyLane: Automatic differentiation of hybrid quantum-classical computations," Jul. 2022, arXiv:1811.04968 [physics, physics:quant-ph]. [Online]. Available: http://arxiv.org/abs/1811.04968

[79] K. Temme, S. Bravyi, and J. M. Gambetta, "Error Mitigation for Short-Depth Quantum Circuits," *Physical Review Letters*, vol. 119, no. 18, p. 180509, Nov. 2017, publisher: American Physical Society. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.119.180509

[80] R. Orús, S. Mugel, and E. Lizaso, "Quantum computing for finance: Overview and prospects," *Reviews in Physics*, vol. 4, p. 100028, Nov. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/

pii/S2405428318300571

[81] G. Noone, "Banks are having a quantum computing revolution - Tech Monitor," Jan. 2023, section: Quantum. [Online]. Available: https://techmonitor.ai/hardware/quantum/who-are-early-adopters-quantum-computers-big-banks-finance

[82] J. Preskill, "Quantum computing and the entanglement frontier," Nov. 2012, arXiv:1203.5813 [cond-mat, physics:quant-ph]. [Online]. Available: http://arxiv.org/abs/1203.5813

[83] ——, "Why I Called It 'Quantum Supremacy'," Oct. 2019. [Online]. Available: https://www.quantamagazine.org/john-preskill-explains-quantum-supremacy-20191002/

[84] S. Boixo, S. V. Isakov, V. N. Smelyanskiy, R. Babbush, N. Ding, Z. Jiang, M. J. Bremner, J. M. Martinis, and H. Neven, "Characterizing quantum supremacy in near-term devices," *Nature Physics*, vol. 14, no. 6, pp. 595–600, Jun. 2018, number: 6 Publisher: Nature Publishing Group.

[Online]. Available: https://www.nature.com/articles/s41567-018-0124-x

[85] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, D. A. Buell, B. Burkett, Y. Chen, Z. Chen, B. Chiaro, R. Collins, W. Courtney, A. Dunsworth, E. Farhi, B. Foxen, A. Fowler, C. Gidney, M. Giustina, R. Graff, K. Guerin, S. Habegger, M. P. Harrigan, M. J. Hartmann, A. Ho, M. Hoffmann, T. Huang, T. S. Humble, S. V. Isakov, E. Jeffrey, Z. Jiang, D. Kafri, K. Kechedzhi, J. Kelly, P. V. Klimov, S. Knysh, A. Korotkov, F. Kostritsa, D. Landhuis, M. Lindmark, E. Lucero, D. Lyakh, S. Mandrà, J. R. McClean, M. McEwen, A. Megrant, X. Mi, K. Michielsen, M. Mohseni, J. Mutus, O. Naaman, M. Neeley, C. Neill, M. Y. Niu, E. Ostby, A. Petukhov, J. C. Platt, C. Quintana, E. G. Rieffel, P. Roushan, N. C. Rubin, D. Sank, K. J. Satzinger, V. Smelyanskiy, K. J. Sung, M. D. Trevithick, A. Vainsencher, B. Villalonga, T. White, Z. J. Yao, P. Yeh, A. Zalcman, H. Neven, and J. M. Martinis, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, Oct. 2019, number: 7779 Publisher: Nature Publishing Group. [Online].

Available: https://www.nature.com/articles/s41586-019-1666-5

[86] "Quantum Machine Learning and the Power of Data," Jun. 2021. [Online]. Available: https://blog.research.google/2021/06/quantum-machine-learning-and-power-of.html

[87] E. Pednault, J. A. Gunnels, G. Nannicini, L. Horesh, and R. Wisnieff, "Leveraging Secondary Storage to Simulate Deep 54-qubit Sycamore Circuits," Oct. 2019, arXiv:1910.09534 [quant-ph]. [Online]. Available: http://arxiv.org/abs/1910.09534

[88] S. Pichai, "Our progress toward quantum error correction," Feb. 2023. [Online]. Available: https://blog.google/inside-google/message-ceo/our-progress-toward-quantum-error-correction/

[89] R. Acharya, I. Aleiner, R. Allen, T. I. Andersen, M. Ansmann, F. Arute, K. Arya, A. Asfaw, J. Atalaya, R. Babbush, D. Bacon, J. C. Bardin, J. Basso, A. Bengtsson, S. Boixo, G. Bortoli, A. Bourassa, J. Bovaird, L. Brill, M. Broughton, B. B. Buckley, D. A. Buell, T. Burger, B. Burkett,

N. Bushnell, Y. Chen, Z. Chen, B. Chiaro, J. Cogan, R. Collins, P. Conner, W. Courtney, A. L. Crook, B. Curtin, D. M. Debroy, A. Del Toro Barba, S. Demura, A. Dunsworth, D. Eppens, C. Erickson, L. Faoro, E. Farhi, R. Fatemi, L. Flores Burgos, E. Forati, A. G. Fowler, B. Foxen, W. Giang, C. Gidney, D. Gilboa, M. Giustina, A. Grajales Dau, J. A. Gross, S. Habegger, M. C. Hamilton, M. P. Harrigan, S. D. Harrington, O. Higgott, J. Hilton, M. Hoffmann, S. Hong, T. Huang, A. Huff, W. J. Huggins, L. B. Ioffe, S. V. Isakov, J. Iveland, E. Jeffrey, Z. Jiang, C. Jones, P. Juhas, D. Kafri, K. Kechedzhi, J. Kelly, T. Khattar, M. Khezri, M. Kieferová, S. Kim, A. Kitaev, P. V. Klimov, A. R. Klots, A. N. Korotkov, F. Kostritsa, J. M. Kreikebaum, D. Landhuis, P. Laptev, K.-M. Lau, L. Laws, J. Lee, K. Lee, B. J. Lester, A. Lill, W. Liu, A. Locharla, E. Lucero, F. D. Malone, J. Marshall, O. Martin, J. R. McClean, T. McCourt, M. McEwen, A. Megrant, B. Meurer Costa, X. Mi, K. C. Miao, M. Mohseni, S. Montazeri, A. Morvan, E. Mount, W. Mruczkiewicz, O. Naaman, M. Neeley, C. Neill, A. Nersisyan, H. Neven, M. Newman, J. H. Ng, A. Nguyen, M. Nguyen, M. Y. Niu, T. E. O'Brien, A. Opremcak, J. Platt, A. Petukhov, R. Potter, L. P.

Pryadko, C. Quintana, P. Roushan, N. C. Rubin, N. Saei, D. Sank, K. Sankaragomathi, K. J. Satzinger, H. F. Schurkus, C. Schuster, M. J. Shearn, A. Shorter, V. Shvarts, J. Skruzny, V. Smelyanskiy, W. C. Smith, G. Sterling, D. Strain, M. Szalay, A. Torres, G. Vidal, B. Villalonga, C. Vollgraff Heidweiller, T. White, C. Xing, Z. J. Yao, P. Yeh, J. Yoo, G. Young, A. Zalcman, Y. Zhang, N. Zhu, and Google Quantum AI, "Suppressing quantum errors by scaling a surface code logical qubit," *Nature*, vol. 614, no. 7949, pp. 676–681, Feb. 2023, number: 7949 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41586-022-05434-1

[90] M. Oliv, A. Matic, T. Messerer, and J. M. Lorenz, "Evaluating the impact of noise on the performance of the Variational Quantum Eigensolver," Sep. 2022, arXiv:2209.12803 [quant-ph]. [Online]. Available: http://arxiv.org/abs/2209.12803

[91] E. Fontana, N. Fitzpatrick, D. M. Ramo, R. Duncan, and I. Rungger, "Evaluating the noise resilience of variational quantum algorithms," *Physical Review A*, vol. 104, no. 2, p. 022403, Aug. 2021. [Online].

Available: https://link.aps.org/doi/10.1103/PhysRevA.104.022403

[92] T. Menneer and A. Narayanan, "Quantum-inspired Neural Networks," in *Proceedings of the Neural Information Processing Systems*, vol. 95, Jun. 1995, pp. 27–30.

[93] M. Schuld and F. Petruccione, "Introduction," in *Machine Learning with Quantum Computers*, ser. Quantum Science and Technology, M. Schuld and F. Petruccione, Eds. Cham: Springer International Publishing, 2021, pp. 1–21. [Online]. Available: https://doi.org/10.1007/978-3-030-83098-4_1

[94] R. Chrisley, "Quantum Learning," in *Proceedings of the international symposium*, vol. 4. Saariselka: New directions in cognitive science, Jul. 1995, pp. 4–9.

[95] D. Ventura and T. Martinez, "Quantum associative memory," *Information Sciences*, vol. 124, no. 1-4, pp. 273–296, 2000.

[96] D. Anguita, S. Ridella, F. Rivieccio, and R. Zunino, "Quantum optimization for training support vector machines," *Neural networks*, vol. 16, no. 5-6, pp. 763–770, Jun. 2003. [Online]. Available: https://doi.org/10.1016/S0893-6080(03)00087-X

[97] A. A. Ezhov and D. Ventura, "Quantum Neural Networks," in *Future Directions for Intelligent Systems and Information Sciences: The Future of Speech and Image Technologies, Brain Computers, WWW, and Bioinformatics*, ser. Studies in Fuzziness and Soft Computing, N. Kasabov, Ed. Heidelberg: Physica-Verlag HD, 2000, pp. 213–235. [Online]. Available: https://doi.org/10.1007/978-3-7908-1856-7_11

[98] B. Ricks and D. Ventura, "Training a Quantum Neural Network," in *Advances in Neural Information Processing Systems*, vol. 16. MIT Press, 2003. [Online]. Available: https://papers.nips.cc/paper_files/paper/2003/hash/505259756244493872b7709a8a01b536-Abstract.html

[99] R. Bonner, "Quantum Computation and Learning," 2002.

[100] K. Najafi, S. F. Yelin, and X. Gao, "The Development of Quantum Machine Learning," *Harvard Data Science Review*, vol. 4, no. 1, Jan. 2021. [Online]. Available: https://hdsr.mitpress.mit.edu/pub/cgmjzm3c/release/3

[101] S. Lloyd, M. Mohseni, and P. Rebentrost, "Quantum principal component analysis," *Nature Physics*, vol. 10, no. 9, pp. 631–633, Sep. 2014, number: 9 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/nphys3029

[102] P. Rebentrost, M. Mohseni, and S. Lloyd, "Quantum Support Vector Machine for Big Data Classification," *Physical Review Letters*, vol. 113, no. 13, p. 130503, Sep. 2014, publisher: American Physical Society. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.113.130503

[103] I. Kerenidis and A. Prakash, "Quantum Recommendation Systems," Sep. 2016, arXiv:1603.08675 [quant-ph]. [Online]. Available: http://arxiv.org/abs/1603.08675

[104] "Front Matter," in *Quantum Machine Learning*, P. Wittek, Ed. Boston: Academic Press, Jan. 2014, pp. i–ii. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128009536000153

[105] V. Giovannetti, S. Lloyd, and L. Maccone, "Quantum Random Access Memory," *Physical Review Letters*, vol. 100, no. 16, p. 160501, Apr. 2008, publisher: American Physical Society. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.100.160501

[106] S. Aaronson, "Read the fine print," *Nature Physics*, vol. 11, no. 4, pp. 291–293, Apr. 2015, number: 4 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/nphys3272

[107] E. Tang, "A quantum-inspired classical algorithm for recommendation systems," in *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, ser. STOC 2019. New York, NY, USA: Association for Computing Machinery, Jun. 2019, pp. 217–228. [Online]. Available: https://doi.org/10.1145/3313276.3316310

[108] ——, "Quantum Principal Component Analysis Only Achieves an Exponential Speedup Because of Its State Preparation Assumptions," *Physical Review Letters*, vol. 127, no. 6, p. 060503, Aug. 2021, publisher: American Physical Society. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.127.060503

[109] E. Aïmeur, G. Brassard, and S. Gambs, "Quantum speed-up for unsupervised learning," *Machine Learning*, vol. 90, no. 2, pp. 261–287, Feb. 2013. [Online]. Available: https://doi.org/10.1007/s10994-012-5316-5

[110] S. Lloyd, M. Mohseni, and P. Rebentrost, "Quantum algorithms for supervised and unsupervised machine learning," Nov. 2013, arXiv:1307.0411 [quant-ph]. [Online]. Available: http://arxiv.org/abs/1307.0411

[111] N. Wiebe, A. Kapoor, and K. M. Svore, "Quantum algorithms for nearest-neighbor methods for supervised and unsupervised learning," *Quantum Information & Computation*, vol. 15, no. 3-4, pp. 316–356, Mar. 2015.

[112] S. Herbert, "Quantum Computing (CST Part II) - Lecture 15: Adiabatic Quantum Computing," 2019. [Online]. Available: https://www.cl.cam.ac.uk/teaching/1920/QuantComp/Quantum_Computing_Lecture_15.pdf

[113] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, "Quantum Computation by Adiabatic Evolution," Jan. 2000, arXiv:quant-ph/0001106. [Online]. Available: http://arxiv.org/abs/quant-ph/0001106

[114] H. Neven, V. S. Denchev, G. Rose, and W. G. Macready, "QBoost: Large Scale Classifier Training withAdiabatic Quantum Optimization," in *Proceedings of the Asian Conference on Machine Learning*. PMLR, Nov. 2012, pp. 333–348, iSSN: 1938-7228. [Online]. Available: https://proceedings.mlr.press/v25/neven12.html

[115] M. Schuld, I. Sinayskiy, and F. Petruccione, "The quest for a Quantum Neural Network," *Quantum Information Processing*, vol. 13, no. 11, pp. 2567–2586, Nov. 2014. [Online]. Available: https://doi.org/10.1007/s11128-014-0809-8

[116] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini, "Parameterized quantum circuits as machine learning models," *Quantum Science and Technology*, vol. 4, no. 4, p. 043001, Nov. 2019, publisher: IOP Publishing. [Online]. Available: https://dx.doi.org/10.1088/2058-9565/ab4eb5

[117] M. Zak and C. P. Williams, "Quantum Neural Nets," *International Journal of Theoretical Physics*, vol. 37, no. 2, pp. 651–684, Feb. 1998. [Online]. Available: https://doi.org/10.1023/A:1026656110699

[118] K. H. Wan, O. Dahlsten, H. Kristjánsson, R. Gardner, and M. S. Kim, "Quantum generalisation of feedforward neural networks," *npj Quantum Information*, vol. 3, no. 1, pp. 1–8, Sep. 2017, number: 1 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41534-017-0032-4

[119] E. C. Behrman, L. R. Nash, J. E. Steck, V. G. Chandrashekar, and S. R. Skinner, "Simulations of quantum neural networks," *Information Sciences*, vol. 128, no. 3, pp. 257–269, Oct. 2000. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025500000566

[120] Y. Cao, G. G. Guerreschi, and A. Aspuru-Guzik, "Quantum Neuron: an elementary building block for machine learning on quantum computers," arXiv, Tech. Rep. arXiv:1711.11240, Nov. 2017, arXiv:1711.11240 [quant-ph] type: article. [Online]. Available: http://arxiv.org/abs/1711.11240

[121] M. Schuld, A. Bocharov, K. M. Svore, and N. Wiebe, "Circuit-centric quantum classifiers," *Physical Review A*, vol. 101, no. 3, p. 032308, Mar. 2020, publisher: American Physical Society. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.101.032308

[122] M. Benedetti, D. Garcia-Pintos, O. Perdomo, V. Leyton-Ortega, Y. Nam, and A. Perdomo-Ortiz, "A generative modeling approach for benchmarking and training shallow quantum circuits," *npj Quantum Information*, vol. 5, no. 1, pp. 1–9, May 2019, number: 1 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41534-019-0157-8

[123] G. Cybenko, "Approximation by superpositions of a sigmoidal function,"

Mathematics of Control, Signals and Systems, vol. 2, no. 4, pp. 303–314, Dec. 1989. [Online]. Available: https://doi.org/10.1007/BF02551274

[124] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251–257, Jan. 1991. [Online]. Available: https://www.sciencedirect.com/science/article/pii/089360809190009T

[125] M. Schuld, "Supervised quantum machine learning models are kernel methods," Apr. 2021, arXiv:2101.11020 [quant-ph, stat]. [Online]. Available: http://arxiv.org/abs/2101.11020

[126] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," Jan. 2017, arXiv:1412.6980 [cs]. [Online]. Available: http://arxiv.org/abs/1412.6980

[127] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, "Supervised learning with quantum-enhanced feature spaces," *Nature*, vol. 567, no. 7747, pp. 209–212, Mar. 2019,

number: 7747 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41586-019-0980-2

[128] "ZZFeatureMap." [Online]. Available: https://docs.quantum.ibm.com/api/qiskit/qiskit.circuit.library.ZZFeatureMap

[129] A. Pérez-Salinas, A. Cervera-Lierta, E. Gil-Fuster, and J. I. Latorre, "Data re-uploading for a universal quantum classifier," *Quantum*, vol. 4, p. 226, Feb. 2020, publisher: Verein zur Förderung des Open Access Publizierens in den Quantenwissenschaften. [Online]. Available: https://quantum-journal.org/papers/q-2020-02-06-226/

[130] J. Landman, S. Thabet, C. Dalyac, H. Mhiri, and E. Kashefi, "Classically Approximating Variational Quantum Machine Learning with Random Fourier Features," Oct. 2022, arXiv:2210.13200. [Online]. Available: http://arxiv.org/abs/2210.13200

[131] S. Ahmed, "Data-reuploading classifier," *PennyLane Demos*, Oct. 2019, publisher: Xanadu. [Online]. Available: https://pennylane.ai/qml/demos

/tutorial_data_reuploading_classifier/

[132] V. P. Ngoc and H. Wiklicky, "Tunable Quantum Neural Networks for Boolean Functions," Nov. 2020, arXiv:2003.14122 [quant-ph]. [Online]. Available: http://arxiv.org/abs/2003.14122

[133] T. Quantum, "MNIST classification | TensorFlow Quantum." [Online]. Available: https://www.tensorflow.org/quantum/tutorials/mnist

[134] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran, "Evaluating analytic gradients on quantum hardware," *Physical Review A*, vol. 99, no. 3, p. 032331, Mar. 2019, publisher: American Physical Society. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.99.032331

[135] M. A. Aizerman, M. Braverman, and L. I. Rozonoèr, "Theoretical foundation of potential functions method in pattern recognition," *Avtomat. i Telemekh.*, vol. 25, no. 6, pp. 917–936, 1964.

[136] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*, ser. COLT '92. New York, NY, USA: Association for Computing Machinery, Jul. 1992, pp. 144–152. [Online]. Available: https://dl.acm.org/doi/10.1145/130385.130401

[137] D. Wilimitis, "The Kernel Trick," Feb. 2019. [Online]. Available: https://towardsdatascience.com/the-kernel-trick-c98cdbcaeb3f

[138] R. Chatterjee and T. Yu, "Generalized coherent states, reproducing kernels, and quantum support vector machines," *Quantum Information & Computation*, vol. 17, no. 15-16, pp. 1292–1306, Dec. 2017.

[139] M. Schuld, M. Fingerhuth, and F. Petruccione, "Implementing a distance-based classifier with a quantum interference circuit," *Europhysics Letters*, vol. 119, no. 6, p. 60002, Dec. 2017, publisher: EDP Sciences, IOP Publishing and Società Italiana di Fisica. [Online]. Available: https://dx.doi.org/10.1209/0295-5075/119/60002

[140] X. Vasques, H. Paik, and L. Cif, "Application of quantum machine learning using quantum kernel algorithms on multiclass neuron M-type classification," *Scientific Reports*, vol. 13, no. 1, p. 11541, Jul. 2023, number: 1 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41598-023-38558-z

[141] S. L. Wu, S. Sun, W. Guan, C. Zhou, J. Chan, C. L. Cheng, T. Pham, Y. Qian, A. Z. Wang, R. Zhang, M. Livny, J. Glick, P. K. Barkoutsos, S. Woerner, I. Tavernelli, F. Carminati, A. Di Meglio, A. C. Y. Li, J. Lykken, P. Spentzouris, S. Y.-C. Chen, S. Yoo, and T.-C. Wei, "Application of quantum machine learning using the quantum kernel algorithm on high energy physics analysis at the LHC," *Physical Review Research*, vol. 3, no. 3, p. 033221, Sep. 2021, publisher: American Physical Society. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevResearch.3.033221

[142] K. Bartkiewicz, C. Gneiting, A. Černoch, K. Jiráková, K. Lemr, and F. Nori, "Experimental kernel-based quantum machine learning in finite feature space," *Scientific Reports*, vol. 10, no. 1, p. 12356, Jul. 2020,

number: 1 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41598-020-68911-5

[143] X. Liu, G. Liu, J. Huang, H.-K. Zhang, and X. Wang, "Mitigating barren plateaus of variational quantum eigensolvers," Aug. 2022, arXiv:2205.13539 [quant-ph]. [Online]. Available: http://arxiv.org/abs/2205.13539

[144] E. Grant, L. Wossnig, M. Ostaszewski, and M. Benedetti, "An initialization strategy for addressing barren plateaus in parametrized quantum circuits," *Quantum*, vol. 3, p. 214, Dec. 2019, publisher: Verein zur Förderung des Open Access Publizierens in den Quantenwissenschaften. [Online]. Available: https://quantum-journal.org/papers/q-2019-12-09-214/

[145] A. Skolik, J. R. McClean, M. Mohseni, P. van der Smagt, and M. Leib, "Layerwise learning for quantum neural networks," *Quantum Machine Intelligence*, vol. 3, no. 1, p. 5, Jan. 2021. [Online]. Available: https://doi.org/10.1007/s42484-020-00036-4

[146] M. Cerezo, A. Sone, T. Volkoff, L. Cincio, and P. J. Coles, "Cost function dependent barren plateaus in shallow parametrized quantum circuits," *Nature Communications*, vol. 12, no. 1, p. 1791, Mar. 2021, number: 1 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41467-021-21728-w

[147] L. G. Wright and P. L. McMahon, "The Capacity of Quantum Neural Networks," in *Conference on Lasers and Electro-Optics (2020), paper JM4G.5.* Optica Publishing Group, May 2020, p. JM4G.5. [Online]. Available: https://opg.optica.org/abstract.cfm?uri=CLEO_SI-2020-JM4G.5

[148] P. J. Coles, "Seeking quantum advantage for neural networks," *Nature Computational Science*, vol. 1, no. 6, pp. 389–390, Jun. 2021, number: 6 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s43588-021-00088-x

[149] "Quantum data | TensorFlow Quantum." [Online]. Available: https://www.tensorflow.org/quantum/tutorials/quantum_data

[150] "Engineering Quantum Algorithms." [Online]. Available: https://www.goldmansachs.com/careers/blog/possibilities-quantum-computing

[151] I. a. T. Department for Science and A. Griffith, "Unlocking the potential of quantum: £45 million investment to drive breakthroughs in brain scanners, navigation systems, and quantum computing," Feb. 2024. [Online]. Available: https://www.gov.uk/government/news/unlocking-the-potential-of-quantum-45-million-investment-to-drive-breakthroughs-in-brain-scanners-navigation-systems-and-quantum-computing

[152] E. Bernstein and U. Vazirani, "Quantum Complexity Theory," *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1411–1473, Oct. 1997, publisher: Society for Industrial and Applied Mathematics. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/S0097539796300921

[153] S. Bravyi, D. Gosset, and R. König, "Quantum advantage with shallow circuits," *Science*, vol. 362, no. 6412, pp. 308–311, Oct. 2018, publisher: American Association for the Advancement of Science. [Online]. Available: https://www.science.org/doi/10.1126/science.aar3106

[154] H.-Y. Huang, M. Broughton, J. Cotler, S. Chen, J. Li, M. Mohseni, H. Neven, R. Babbush, R. Kueng, J. Preskill, and J. R. McClean, "Quantum advantage in learning from experiments," *Science*, vol. 376, no. 6598, pp. 1182–1186, Jun. 2022, publisher: American Association for the Advancement of Science. [Online]. Available: https://www.science.org/doi/10.1126/science.abn7293

[155] A. J. Daley, I. Bloch, C. Kokail, S. Flannigan, N. Pearson, M. Troyer, and P. Zoller, "Practical quantum advantage in quantum simulation," *Nature*, vol. 607, no. 7920, pp. 667–676, Jul. 2022, publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41586-022-04940-6

[156] D. Aharonov and A. Ta-Shma, "Adiabatic quantum state generation and statistical zero knowledge," in *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, ser. STOC '03. New York, NY, USA: Association for Computing Machinery, Jun. 2003, pp. 20–29. [Online]. Available: https://dl.acm.org/doi/10.1145/780542.780546

[157] M. Schuld and F. Petruccione, *Supervised Learning with Quantum Computers*, ser. Quantum Science and Technology. Cham: Springer International Publishing, 2018. [Online]. Available: https://link.springer.com/10.1007/978-3-319-96424-9

[158] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, number: 7553 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/nature14539

[159] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, Dec. 1943. [Online]. Available: https://doi.org/10.1007/BF02478259

[160] D. O. Hebb, *The organization of behavior; a neuropsychological theory*. Oxford, England: Wiley, 1949, pages: xix, 335.

[161] A. M. Turing, "I.— Computing Machinery and Intelligence," *Mind*,

vol. LIX, no. 236, pp. 433–460, Oct. 1950. [Online]. Available: https://doi.org/10.1093/mind/LIX.236.433

[162] J. McCarthy, M. L. Minsky, N. Rochester, I. B. M. Corporation, and C. E. Shannon, "A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence," *AI Magazine*, vol. 27, no. 4, p. 12, Aug. 1955. [Online]. Available: https://doi.org/10.1609/aimag.v27i4.1904

[163] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958, place: US Publisher: American Psychological Association. [Online]. Available: https://doi.org/10.1037/h0042519

[164] A. Ivakhnenko and V. G. Lapa, "Cybernetic Predicting Devices," Apr. 1966. [Online]. Available: https://api.semanticscholar.org/CorpusID:61108232

[165] S. Amari, "A Theory of Adaptive Pattern Classifiers," *IEEE Transactions on Electronic Computers*, vol. EC-16, no. 3, pp. 299–307, Jun. 1967,

conference Name: IEEE Transactions on Electronic Computers. [Online]. Available: https://ieeexplore.ieee.org/document/4039068

[166] M. Marvin and S. Papert, "Perceptrons," Jan. 1969. [Online]. Available: https://mitpress.mit.edu/9780262630221/perceptrons/

[167] S. Linnainmaa, "Taylor expansion of the accumulated rounding error," *BIT Numerical Mathematics*, vol. 16, no. 2, pp. 146–160, Jun. 1976. [Online]. Available: https://doi.org/10.1007/BF01931367

[168] P. J. Werbos, "Applications of advances in nonlinear sensitivity analysis," in *System Modeling and Optimization*, R. F. Drenick and F. Kozin, Eds. Berlin, Heidelberg: Springer, 1982, pp. 762–770. [Online]. Available: https://doi.org/10.1007/BFb0006203

[169] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*, vol. 25. Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2

012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html

[170] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2009, pp. 248–255, iSSN: 1063-6919. [Online]. Available: https://ieeexplore.ieee.org/document/5206848

[171] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., Dec. 2017, pp. 6000–6010.

[172] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," May 2016, arXiv:1409.0473 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1409.0473

[173] S. Geman, E. Bienenstock, and R. Doursat, "Neural Networks and the

Bias/Variance Dilemma," *Neural Computation*, vol. 4, no. 1, pp. 1–58, Jan. 1992. [Online]. Available: https://direct.mit.edu/neco/article/4/1/1-58/5624

[174] M. Belkin, D. Hsu, S. Ma, and S. Mandal, "Reconciling modern machine-learning practice and the classical bias–variance trade-off," *Proceedings of the National Academy of Sciences*, vol. 116, no. 32, pp. 15 849–15 854, Aug. 2019, publisher: Proceedings of the National Academy of Sciences. [Online]. Available: https://www.pnas.org/doi/10.1073/pnas.1903070116

[175] S. Gunasekar, B. Woodworth, S. Bhojanapalli, B. Neyshabur, and N. Srebro, "Implicit Regularization in Matrix Factorization," in *2018 Information Theory and Applications Workshop (ITA)*. San Diego, CA: IEEE, Feb. 2018, pp. 1–10. [Online]. Available: https://ieeexplore.ieee.org/document/8503198/

[176] S. Arora, N. Cohen, W. Hu, and Y. Luo, "Implicit regularization in deep matrix factorization," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA:

Curran Associates Inc., Dec. 2019, pp. 7413–7424. [Online]. Available: https://dl.acm.org/doi/abs/10.5555/3454287.3454953

[177] Z. Li, Y. Luo, and K. Lyu, "Towards Resolving the Implicit Bias of Gradient Descent for Matrix Factorization: Greedy Low-Rank Learning," 2021. [Online]. Available: https://doi.org/10.48550/arXiv.2012.09839

[178] G. V. Pérez, A. A. Louis, and C. Q. Camargo, "Deep learning generalizes because the parameter-function map is biased towards simple functions," 2019. [Online]. Available: https://arxiv.org/abs/1805.08522

[179] M. Huh, H. Mobahi, R. Zhang, B. Cheung, P. Agrawal, and P. Isola, "The Low-Rank Simplicity Bias in Deep Networks," Mar. 2023, arXiv:2103.10427 [cs]. [Online]. Available: http://arxiv.org/abs/2103.10427

[180] C. Mingard, G. Valle-Pérez, J. Skalse, and A. A. Louis, "Is SGD a Bayesian sampler? well, almost," *J. Mach. Learn. Res.*, vol. 22, no. 1, pp. 79:3579–79:3642, Jan. 2021.

[181] C. Mingard, H. Rees, G. Valle-Pérez, and A. A. Louis, "Do deep neural networks have an inbuilt Occam's razor?" Apr. 2023, arXiv:2304.06670 [cs, stat]. [Online]. Available: http://arxiv.org/abs/2304.06670

[182] C. Mingard, J. Skalse, G. Valle-Pérez, D. Martínez-Rubio, V. Mikulik, and A. A. Louis, "Neural networks are a priori biased towards Boolean functions with low entropy," Jan. 2020, arXiv:1909.11522 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1909.11522

[183] H. Shah, K. Tamuly, A. Raghunathan, P. Jain, and P. Netrapalli, "The Pitfalls of Simplicity Bias in Neural Networks," in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 9573–9585. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/6cfe0e6127fa25df2a0ef2ae1067d915-Abstract.html

[184] G. De Palma, B. Kiani, and S. Lloyd, "Random deep neural networks are biased towards simple functions," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/hash/feab05aa9

1085b7a8012516bc3533958-Abstract.html

[185] J. Schmidhuber, "Discovering Problem Solutions with Low Kolmogorov Complexity and High Generalization Capability," Technical Report FKI-194-94, Aug. 1994.

[186] H. W. Lin, M. Tegmark, and D. Rolnick, "Why Does Deep and Cheap Learning Work So Well?" *Journal of Statistical Physics*, vol. 168, no. 6, pp. 1223–1247, Sep. 2017. [Online]. Available: https://doi.org/10.1007/s10955-017-1836-5

[187] S. Spigler, M. Geiger, and M. Wyart, "Asymptotic learning curves of kernel methods: empirical data versus teacher–student paradigm," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2020, no. 12, p. 124001, Dec. 2020, publisher: IOP Publishing and SISSA. [Online]. Available: https://dx.doi.org/10.1088/1742-5468/abc61d

[188] R. J. Solomonoff, "A preliminary report on a general theory of inductive inference," 1960.

[189] ——, "A formal theory of inductive inference. Part I," *Information and Control*, vol. 7, no. 1, pp. 1–22, Mar. 1964. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0019995864902232

[190] A. N. Kolmogorov, "Three approaches to the quantitative definition of information," *International Journal of Computer Mathematics*, vol. 2, no. 1-4, pp. 157–168, Jan. 1968, publisher: Taylor & Francis _eprint: https://doi.org/10.1080/00207166808803030. [Online]. Available: https://doi.org/10.1080/00207166808803030

[191] G. J. Chaitin, "A Theory of Program Size Formally Identical to Information Theory," *J. ACM*, vol. 22, no. 3, pp. 329–340, Jul. 1975. [Online]. Available: https://dl.acm.org/doi/10.1145/321892.321894

[192] ——, *Algorithmic Information Theory*, ser. Cambridge Tracts in Theoretical Computer Science. Cambridge: Cambridge University Press, 1987. [Online]. Available: https://www.cambridge.org/core/books/algorithmic-information-theory/66D88D412DE158C21D392E2EF3112CC1

[193] L. A. Levin, "Laws of Information Conservation (Nongrowth) and Aspects of the Foundation of Probability Theory," *Russian Academy of Sciences, Branch of Informatics*, vol. 20, no. 3, pp. 30–35, 1974.

[194] S. Sayad, "Perceptron." [Online]. Available: https://www.saedsayad.com/artificial_neural_network_bkp.htm

[195] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, Jan. 1989. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0893608089900208

[196] I. Safran and O. Shamir, "Depth-width tradeoffs in approximating natural functions with neural networks," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ser. ICML'17. Sydney, NSW, Australia: JMLR.org, Aug. 2017, pp. 2979–2987.

[197] H. Mhaskar, Q. Liao, and T. Poggio, "When and why are deep networks better than shallow ones?" in *Proceedings of the Thirty-First AAAI*

Conference on Artificial Intelligence, ser. AAAI'17. San Francisco, California, USA: AAAI Press, Feb. 2017, pp. 2343–2349.

[198] P. Drmota, D. Main, E. M. Ainley, A. Agrawal, G. Araneda, D. P. Nadlinger, B. C. Nichol, R. Srinivas, A. Cabello, and D. M. Lucas, "Experimental Quantum Advantage in the Odd-Cycle Game," Jun. 2024, arXiv:2406.08412 [quant-ph]. [Online]. Available: http://arxiv.org/abs/2406.08412

[199] Y. Du, M.-H. Hsieh, T. Liu, and D. Tao, "Expressive power of parametrized quantum circuits," *Physical Review Research*, vol. 2, no. 3, p. 033125, Jul. 2020, publisher: American Physical Society. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevResearch.2.033125

[200] S. Sim, P. D. Johnson, and A. Aspuru-Guzik, "Expressibility and Entangling Capability of Parameterized Quantum Circuits for Hybrid Quantum-Classical Algorithms," *Advanced Quantum Technologies*, vol. 2, no. 12, p. 1900070, 2019, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/qute.201900070. [On-

line]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/qute.201900070

[201] Y. Wu, J. Yao, P. Zhang, and H. Zhai, "Expressivity of quantum neural networks," *Physical Review Research*, vol. 3, no. 3, p. L032049, Aug. 2021, publisher: American Physical Society. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevResearch.3.L032049

[202] L. De Moura and N. Bjørner, "Z3: an efficient SMT solver," in *Proceedings of the Theory and practice of software, 14th international conference on Tools and algorithms for the construction and analysis of systems*, ser. TACAS'08/ETAPS'08. Berlin, Heidelberg: Springer-Verlag, Mar. 2008, pp. 337–340.

[203] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, Dec. 2010. [Online]. Available: https://www.cambridge.org/highereducation/books/quantum-computation-and-quantum-information/01E10196D0A682A6AEFFEA52D53BE9AE

[204] A. W. Harrow, "Quantum states and operations," 2018. [Online]. Available: https://web.mit.edu/8.371/www/lectures/lect02.pdf

[205] F. Vatan and C. Williams, "Optimal quantum circuits for general two-qubit gates," *Physical Review A*, vol. 69, no. 3, p. 032315, Mar. 2004, publisher: American Physical Society. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.69.032315

[206] O. D. Matteo, "Understanding the Haar measure," *PennyLane Demos*, Mar. 2021, publisher: Xanadu. [Online]. Available: https://pennylane.ai/qml/demos/tutorial_haar_measure/

[207] H. de Guise, O. Di Matteo, and L. L. Sánchez-Soto, "Simple factorization of unitary transformations," *Physical Review A*, vol. 97, no. 2, p. 022328, Feb. 2018, publisher: American Physical Society. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.97.022328

[208] A. A. Mele, "Introduction to Haar Measure Tools in Quantum Information: A Beginner's Tutorial," *Quantum*, vol. 8, p. 1340,

May 2024, publisher: Verein zur Förderung des Open Access Publizierens in den Quantenwissenschaften. [Online]. Available: https://quantum-journal.org/papers/q-2024-05-08-1340/

[209] I. Cong, S. Choi, and M. D. Lukin, "Quantum convolutional neural networks," *Nature Physics*, vol. 15, no. 12, pp. 1273–1278, Dec. 2019, number: 12 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41567-019-0648-8

[210] Qiskit, "The Quantum Convolution Neural Network," Feb. 2024. [Online]. Available: https://qiskit-community.github.io/qiskit-machine-learning/tutorials/11_quantum_convolutional_neural_networks.html

[211] "PauliFeatureMap | IBM Quantum Documentation." [Online]. Available: https://docs.quantum.ibm.com/api/qiskit/0.19/qiskit.circuit.library.PauliFeatureMap

[212] J. C. Garcia-Escartin and P. Chamorro-Posada, "Equivalent Quantum Circuits," Oct. 2011, arXiv:1110.2998 [quant-ph]. [Online]. Available:

http://arxiv.org/abs/1110.2998

[213] C. Lomont, "Quantum Circuit Identities," Jul. 2003, arXiv:quant-ph/0307111. [Online]. Available: http://arxiv.org/abs/quant-ph/0307111

[214] K. Hietala, R. Rand, S.-H. Hung, X. Wu, and M. Hicks, "A verified optimizer for Quantum circuits," *Proc. ACM Program. Lang.*, vol. 5, no. POPL, pp. 1–29, 2021. [Online]. Available: https://doi.org/10.1145/3434318

[215] G. E. Crooks, "Gates, States, and Circuits," 2021. [Online]. Available: https://threeplusone.com/pubs/on_gates.pdf

[216] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, "Elementary gates for quantum computation," *Physical Review A*, vol. 52, no. 5, pp. 3457–3467, Nov. 1995, publisher: American Physical Society. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.52.3457

[217] S. Bravyi and D. Gosset, "Improved Classical Simulation of Quantum Circuits Dominated by Clifford Gates," *Physical Review Letters*, vol. 116, no. 25, p. 250501, Jun. 2016, publisher: American Physical Society. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.116.250501

[218] "Get backend information with Qiskit." [Online]. Available: https://docs.quantum.ibm.com/run/get-backend-information

[219] S. Johnstun and J.-F. Van Huele, "Understanding and compensating for noise on IBM quantum computers," *American Journal of Physics*, vol. 89, no. 10, pp. 935–942, Oct. 2021. [Online]. Available: https://doi.org/10.1119/10.0006204

[220] "Device backend noise model simulations - Qiskit Aer 0.14.0.1." [Online]. Available: https://qiskit.github.io/qiskit-aer/tutorials/2_device_noise_simulation.html#

[221] H. Massalin, "Superoptimizer: a look at the smallest program," *ACM*

SIGARCH Computer Architecture News, vol. 15, no. 5, pp. 122–126, Nov. 1987, publisher: Association for Computing Machinery (ACM). [Online]. Available: https://doi.org/10.1145/36177.36194

[222] S. Bansal and A. Aiken, "Binary Translation Using Peephole Superoptimizers," in *Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'08. USA: USENIX Association, 2008, pp. 177–192, event-place: San Diego, California.

[223] ——, "Automatic generation of peephole superoptimizers," in *Proceedings of the 12th international conference on Architectural support for programming languages and operating systems*, ser. ASPLOS XII. New York, NY, USA: Association for Computing Machinery, Oct. 2006, pp. 394–403. [Online]. Available: https://dl.acm.org/doi/10.1145/1168857.1168906

[224] A. W. Cross, L. S. Bishop, J. A. Smolin, and J. M. Gambetta, "Open Quantum Assembly Language," Jul. 2017, arXiv:1707.03429 [quant-ph]. [Online]. Available: http://arxiv.org/abs/1707.03429

[225] K. Svore, A. Aho, A. Cross, I. Chuang, and I. Markov, "A layered software architecture for quantum computing design tools," *Computer*, vol. 39, no. 1, pp. 74–83, Jan. 2006, conference Name: Computer. [Online]. Available: https://ieeexplore.ieee.org/document/1580386

[226] I. Chuang, "qasm2circ," 2005. [Online]. Available: http://www.media.mit.edu/quanta/qasm2circ/

[227] A. Cross, "qasm-tools," 2005. [Online]. Available: http://www.media.mit.edu/quanta/quanta-web/projects

[228] S. Balensiefer, L. Kreger-Stickles, and M. Oskin, "QUALE: quantum architecture layout evaluator," in *Quantum Information and Computation III*, vol. 5815. SPIE, May 2005, pp. 103–114. [Online]. Available: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/5815/0000/QUALE-quantum-architecture-layout-evaluator/10.1117/12.604073.full

[229] M. J. Dousti, A. Shafaei, and M. Pedram, "Squash 2: A

Hierarchical Scalable Quantum Mapper Considering Ancilla Sharing," Dec. 2015, arXiv:1512.07402 [quant-ph]. [Online]. Available: http://arxiv.org/abs/1512.07402

[230] "Cirq," 2018. [Online]. Available: https://quantumai.google/cirq

[231] M. Xu, Z. Li, O. Padon, S. Lin, J. Pointing, A. Hirth, H. Ma, J. Palsberg, A. Aiken, U. A. Acar, and Z. Jia, "Quartz: superoptimization of Quantum circuits," in *Proceedings of the 43rd ACM SIGPLAN International Conference on Programming Language Design and Implementation*. San Diego CA USA: ACM, Jun. 2022, pp. 625–640. [Online]. Available: https://dl.acm.org/doi/10.1145/3519939.3523433

[232] T. Hao, X. Huang, C. Jia, and C. Peng, "A Quantum-Inspired Tensor Network Algorithm for Constrained Combinatorial Optimization Problems," *Frontiers in Physics*, vol. 10, Jul. 2022, publisher: Frontiers. [Online]. Available: https://www.frontiersin.org/journals/physics/articles/10.3389/fphy.2022.906590/full

[233] G. Paradezhenko, A. Pervishko, and D. Yudin, "Quantum-Assisted Open-Pit Optimization," *JETP Letters*, vol. 119, no. 6, pp. 470–478, Mar. 2024. [Online]. Available: https://doi.org/10.1134/S0021364023604256

[234] A. Montanaro, "Quantum algorithms: an overview," *npj Quantum Information*, vol. 2, no. 1, pp. 1–8, Jan. 2016, publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/npjqi201523

[235] J. S. Otterbach, R. Manenti, N. Alidoust, A. Bestwick, M. Block, B. Bloom, S. Caldwell, N. Didier, E. S. Fried, S. Hong, P. Karalekas, C. B. Osborn, A. Papageorge, E. C. Peterson, G. Prawiroatmodjo, N. Rubin, C. A. Ryan, D. Scarabelli, M. Scheer, E. A. Sete, P. Sivarajah, R. S. Smith, A. Staley, N. Tezak, W. J. Zeng, A. Hudson, B. R. Johnson, M. Reagor, M. P. da Silva, and C. Rigetti, "Unsupervised Machine Learning on a Hybrid Quantum Computer," Dec. 2017, arXiv:1712.05771 [quant-ph]. [Online]. Available: http://arxiv.org/abs/1712.05771

[236] N. Moll, P. Barkoutsos, L. S. Bishop, J. M. Chow, A. Cross, D. J.

Egger, S. Filipp, A. Fuhrer, J. M. Gambetta, M. Ganzhorn, A. Kandala, A. Mezzacapo, P. Müller, W. Riess, G. Salis, J. Smolin, I. Tavernelli, and K. Temme, "Quantum optimization using variational algorithms on near-term quantum devices," *Quantum Science and Technology*, vol. 3, no. 3, p. 030503, Jun. 2018, publisher: IOP Publishing. [Online]. Available: https://dx.doi.org/10.1088/2058-9565/aab822

[237] H. Neven, G. Rose, and W. G. Macready, "Image recognition with an adiabatic quantum computer I. Mapping to quadratic unconstrained binary optimization," Apr. 2008, arXiv:0804.4457 [quant-ph]. [Online]. Available: http://arxiv.org/abs/0804.4457

[238] A. Lucas, "Ising formulations of many NP problems," *Frontiers in Physics*, vol. 2, Feb. 2014, publisher: Frontiers. [Online]. Available: https://www.frontiersin.org/journals/physics/articles/10.3389/fphy.2014.00005/full

[239] C. H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani, "Strengths and Weaknesses of Quantum Computing," *SIAM Journal*

on Computing, vol. 26, no. 5, pp. 1510–1523, Oct. 1997, publisher: Society for Industrial and Applied Mathematics. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/S0097539796300933

[240] G. E. Crooks, "Performance of the Quantum Approximate Optimization Algorithm on the Maximum Cut Problem," Nov. 2018, arXiv:1811.08419 [quant-ph]. [Online]. Available: http://arxiv.org/abs/1811.08419

[241] G. G. Guerreschi and A. Y. Matsuura, "QAOA for Max-Cut requires hundreds of qubits for quantum speed-up," *Scientific Reports*, vol. 9, no. 1, p. 6903, May 2019, publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41598-019-43176-9

[242] V. E. Elfving, B. W. Broer, M. Webber, J. Gavartin, M. D. Halls, K. P. Lorton, and A. Bochevarov, "How will quantum computers provide an industrially relevant computational advantage in quantum chemistry?" Sep. 2020, arXiv:2009.12472 [physics, physics:quant-ph]. [Online]. Available: http://arxiv.org/abs/2009.12472

[243] L. Zhou, S.-T. Wang, S. Choi, H. Pichler, and M. D. Lukin, "Quantum Approximate Optimization Algorithm: Performance, Mechanism, and Implementation on Near-Term Devices," *Physical Review X*, vol. 10, no. 2, p. 021067, Jun. 2020, publisher: American Physical Society. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevX.10.021067

[244] E. Farhi, J. Goldstone, S. Gutmann, J. Lapan, A. Lundgren, and D. Preda, "A Quantum Adiabatic Evolution Algorithm Applied to Random Instances of an NP-Complete Problem," *Science*, vol. 292, no. 5516, pp. 472–475, Apr. 2001, publisher: American Association for the Advancement of Science. [Online]. Available: https://www.science.org/doi/10.1126/science.1057726

[245] S. Hadfield, Z. Wang, B. O'Gorman, E. G. Rieffel, D. Venturelli, and R. Biswas, "From the Quantum Approximate Optimization Algorithm to a Quantum Alternating Operator Ansatz," *Algorithms*, vol. 12, no. 2, p. 34, Feb. 2019, number: 2 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/1999-4893/12/2/34

[246] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets," *Nature*, vol. 549, no. 7671, pp. 242–246, Sep. 2017, publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/nature23879

[247] G. Nannicini, "Performance of hybrid quantum-classical variational heuristics for combinatorial optimization," *Physical Review E*, vol. 99, no. 1, p. 013304, Jan. 2019, publisher: American Physical Society. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.99.013304

[248] M. Motta, C. Sun, A. T. K. Tan, M. J. O'Rourke, E. Ye, A. J. Minnich, F. G. S. L. Brandão, and G. K.-L. Chan, "Determining eigenstates and thermal states on a quantum computer using quantum imaginary time evolution," *Nature Physics*, vol. 16, no. 2, pp. 205–210, Feb. 2020, publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41567-019-0704-4

[249] S. McArdle, T. Jones, S. Endo, Y. Li, S. C. Benjamin, and

X. Yuan, "Variational ansatz-based quantum simulation of imaginary time evolution," *npj Quantum Information*, vol. 5, no. 1, pp. 1–6, Sep. 2019, publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41534-019-0187-2

[250] T. B. Johnson, *Optimum open pit mine production scheduling*. University of California, Berkeley, 1968.

[251] D. Bienstock and M. Zuckerberg, "Solving LP Relaxations of Large-Scale Precedence Constrained Problems," in *Integer Programming and Combinatorial Optimization*, F. Eisenbrand and F. B. Shepherd, Eds. Berlin, Heidelberg: Springer, 2010, pp. 1–14. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-13036-6_1

[252] D. Hochbaum and A. Chen, "Improved planning for the open-pit mining problem," *Oper. Res*, vol. 48, no. 6, pp. 894–914, 2000.

[253] C. Meagher, R. Dimitrakopoulos, and D. Avis, "Optimized open pit mine design, pushbacks and the gap problem—a review," *Journal of Mining*

Science, vol. 50, no. 3, pp. 508–526, May 2014. [Online]. Available: https://doi.org/10.1134/S1062739114030132

[254] M. Brazil and M. Zachariasen, *Optimal Interconnection Trees in the Plane: Theory, Algorithms and Applications*, ser. Algorithms and Combinatorics. Cham: Springer International Publishing, 2015, vol. 29. [Online]. Available: https://link.springer.com/10.1007/978-3-319-13915-9

[255] T. Lagos, M. Armstrong, T. Homem-de Mello, G. Lagos, and D. Sauré, "A framework for adaptive open-pit mining planning under geological uncertainty," *Optimization and Engineering*, vol. 23, no. 1, pp. 111–146, Sep. 2020. [Online]. Available: https://doi.org/10.1007/s11081-020-09557-0

[256] D. S. Hochbaum and A. Chen, "Performance Analysis and Best Implementations of Old and New Algorithms for the Open-Pit Mining Problem," *Operations Research*, vol. 48, no. 6, pp. 894–914, Dec. 2000, publisher: INFORMS. [Online]. Available: https://pubsonline.informs.org/doi/abs/10.1287/opre.48.6.894.12392

[257] A. Lamghari, R. Dimitrakopoulos, and J. A. Ferland, "A variable neighbourhood descent algorithm for the open-pit mine production scheduling problem with metal uncertainty," *Journal of the Operational Research Society*, vol. 65, no. 9, pp. 1305–1314, Sep. 2014. [Online]. Available: https://doi.org/10.1057/jors.2013.81

[258] A. Lamghari and R. Dimitrakopoulos, "Hyper-heuristic approaches for strategic mine planning under uncertainty," *Comput. Oper. Res.*, vol. 115, no. C, Mar. 2020. [Online]. Available: https://doi.org/10.1016/j.cor.2018.11.010

[259] D. Whittle, "Open pit planning and design," in *SME Mining Engineering Handbook*, G. A. Fourie and G. C. Dohm, Eds. SME, 1992, pp. 1274–1290, section: 13. [Online]. Available: https://books.google.com.au/books?dq=SME%20Mine%20Engineering%20Handbook&hl=en&id=5uq-kdfHLWUC&lr=&oi=fnd&ots=wqpmyiw0YG&pg=PP1&sig=Z_UeLcyH2tpO-j4SqKUUsApcTLM#v=onepage&q=SME%20Mine%20Engineering%20Handbook&f=false

[260] R. Dimitrakopoulos and A. Lamghari, "Simultaneous stochastic optimization of mining complexes - mineral value chains: an overview of concepts, examples and comparisons," *International Journal of Mining, Reclamation and Environment*, vol. 36, no. 6, pp. 443–460, Jul. 2022, publisher: Taylor & Francis _eprint: https://doi.org/10.1080/17480930.2022.2065730. [Online]. Available: https://doi.org/10.1080/17480930.2022.2065730

[261] Y. C. Kim, *Mathematical programming analysis of mine planning problems*. The Pennsylvania State University, 1967.

[262] R. Venkataramani, *Mathematical programming applications in the crushed stone industry*. The Pennsylvania State University, 1970.

[263] S. Hoerger, F. Seymour, and L. Hoffman, "Mine planning at Newmont's Nevada operations," *Mining engineering*, vol. 51, no. 10, pp. 26–30, 1999, publisher: Society for Mining, Metallurgy, and Exploration.

[264] P. Stone, G. Froyland, M. Menabde, B. Law, R. Pasyar, and

P. H. L. Monkhouse, "Blasor—Blended Iron Ore Mine Planning Optimisation at Yandi, Western Australia," in *Advances in Applied Strategic Mine Planning*, R. Dimitrakopoulos, Ed. Cham: Springer International Publishing, 2018, pp. 39–46. [Online]. Available: https://doi.org/10.1007/978-3-319-69320-0_4

[265] M. Menabde, P. Stone, B. Law, and B. Baird, "Blasor–A generalized strategic mine planning optimization tool," in *2007 SME Annual Meeting and Exhibit*, 2007.

[266] G. Whittle, "Misguided objectives that destroy value," in *Proceedings orebody modelling and strategic mine planning*, 2009, pp. 97–101. [Online]. Available: https://www.whittleconsulting.com.au/wp-content/uploads/2023/04/Misguided-Objectives-That-Destroy-Value.pdf

[267] ——, "Whittling away the waste," *Mining Magazine*, 2014. [Online]. Available: https://www.whittleconsulting.com.au/wp-content/uploads/2023/04/Whittling-away-the-waste-MM-June-2014.pdf

[268] J.-C. Picard, "Maximal Closure of a Graph and Applications to Combinatorial Problems," *Management Science*, vol. 22, no. 11, pp. 1268–1272, Jul. 1976, publisher: INFORMS. [Online]. Available: https://pubsonline.informs.org/doi/abs/10.1287/mnsc.22.11.1268

[269] L. Caccetta, "Application of Optimisation Techniques in Open Pit Mining," in *Handbook Of Operations Research In Natural Resources*, A. Weintraub, C. Romero, T. Bjørndal, R. Epstein, and J. Miranda, Eds. Boston, MA: Springer US, 2007, pp. 547–559. [Online]. Available: https://doi.org/10.1007/978-0-387-71815-6_29

[270] G. Thomas, "Pit optimization and mine production scheduling-the way ahead," in *Proc. 26th APCOM Symposium of the Society of Mining Engineers*, 1996, pp. 221–228.

[271] G. D. Bond, "A mathematical analysis of the lerchs and grossmann algorithm and the nested lerchs and grossmann algorithm," PhD Thesis, Colorado School of Mines, USA, 1995.

[272] E. Jélvez, N. Morales, P. Nancel-Penard, and F. Cornillier, "A new hybrid heuristic algorithm for the Precedence Constrained Production Scheduling Problem: A mining application," *Omega*, vol. 94, p. 102046, Jul. 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0305048318301750

[273] D. Whittle, M. Brazil, P. A. Grossman, J. H. Rubinstein, and D. A. Thomas, "Combined optimisation of an open-pit mine outline and the transition depth to underground mining," *European Journal of Operational Research*, vol. 268, no. 2, pp. 624–634, Jul. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221718301000

[274] Z. Bian, F. Chudak, R. B. Israel, B. Lackey, W. G. Macready, and A. Roy, "Mapping Constrained Optimization Problems to Quantum Annealing with Application to Fault Diagnosis," *Frontiers in ICT*, vol. 3, Jul. 2016, publisher: Frontiers. [Online]. Available: https://www.frontiersin.org/journals/ict/articles/10.3389/fict.2016.00014/full

[275] G. Rosenberg, M. Vazifeh, B. Woods, and E. Haber, "Building an

iterative heuristic solver for a quantum annealer," *Computational Optimization and Applications*, vol. 65, no. 3, pp. 845–869, Dec. 2016. [Online]. Available: https://doi.org/10.1007/s10589-016-9844-y

[276] H. Karimi, G. Rosenberg, and H. G. Katzgraber, "Effective optimization using sample persistence: A case study on quantum annealers and various Monte Carlo optimization methods," *Physical Review E*, vol. 96, no. 4, p. 043312, Oct. 2017, publisher: American Physical Society. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.96.043312

[277] R. Shaydulin, H. Ushijima-Mwesigwa, I. Safro, S. Mniszewski, and Y. Alexeev, "Community Detection Across Emerging Quantum Architectures," Oct. 2018, arXiv:1810.07765 [quant-ph]. [Online]. Available: http://arxiv.org/abs/1810.07765

[278] ——, "Network community detection on small quantum computers," *Adv. Quant. Tech*, vol. 2, no. 9, p. 1900029, 2019, publisher: Wiley Online Library. [Online]. Available: https://doi.org/10.1002/qute.201900029

[279] R. Shaydulin, H. Ushijima-Mwesigwa, C. F. A. Negre, I. Safro, S. M. Mniszewski, and Y. Alexeev, "A Hybrid Approach for Solving Optimization Problems on Small Quantum Computers," *Computer*, vol. 52, no. 6, pp. 18–26, Jun. 2019, conference Name: Computer. [Online]. Available: https://ieeexplore.ieee.org/document/8728102

[280] T. R. Carlson, J. Erickson, D. O'Brain, and M. Pana, "Computer techniques in mine planning," *Mining Engineering*, vol. 18, no. 5, pp. 53–56, 1966.

[281] A. Wright, "MOVING CONE II-A simple algorithm for optimum pit limits design," *Proceedings of the 28rd APCOM*, pp. 367–374, 1999.

[282] K. Dagdelen, "Optimum multi period open pit mine production scheduling." Ph.D. dissertation, Colorado School of Mines. Arthur Lakes Library, 1985. [Online]. Available: https://hdl.handle.net/11124/170225

[283] G. Aleksandrowicz, T. Alexander, P. Barkoutsos, L. Bello, Y. Ben-Haim, D. Bucher, F. Cabrera-Hernández, J. Carballo-Franquis, A. Chen,

C. Chen, and others, "Qiskit: An open-source framework for quantum computing," 2017. [Online]. Available: https://github.com/Qiskit/qiskit

[284] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization," *ACM Trans. Math. Softw.*, vol. 23, no. 4, pp. 550–560, Dec. 1997. [Online]. Available: https://dl.acm.org/doi/10.1145/279232.279236

[285] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A Limited Memory Algorithm for Bound Constrained Optimization," *SIAM Journal on Scientific Computing*, vol. 16, no. 5, pp. 1190–1208, Sep. 1995, publisher: Society for Industrial and Applied Mathematics. [Online]. Available: https://epubs.siam.org/doi/10.1137/0916069

[286] J. L. Morales and J. Nocedal, "Remark on "algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound constrained optimization"," *ACM Trans. Math. Softw.*, vol. 38, no. 1, pp. 7:1–7:4, Dec. 2011. [Online]. Available: https://dl.acm.org/doi/10.1145/2049662.2049669

[287] J. C. Spall, "An overview of the simultaneous perturbation method for efficient optimization," *Johns Hopkins APL Tech. Dig.*, vol. 19, no. 4, pp. 482–492, 1998. [Online]. Available: https://secwww.jhuapl.edu/techdigest/Content/techdigest/pdf/V19-N04/19-04-Spall.pdf

[288] J. Spall, "Adaptive stochastic approximation by the simultaneous perturbation method," in *Proceedings of the 37th IEEE Conference on Decision and Control (Cat. No.98CH36171)*, vol. 4, Dec. 1998, pp. 3872–3879 vol.4, iSSN: 0191-2216. [Online]. Available: https://ieeexplore.ieee.org/document/761833

[289] A. Kandala, K. Temme, A. D. Córcoles, A. Mezzacapo, J. M. Chow, and J. M. Gambetta, "Error mitigation extends the computational reach of a noisy quantum processor," *Nature*, vol. 567, no. 7749, pp. 491–495, Mar. 2019, publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41586-019-1040-7

[290] S. Bravyi, S. Sheldon, A. Kandala, D. C. Mckay, and J. M. Gambetta, "Mitigating measurement errors in multiqubit experiments,"

Physical Review A, vol. 103, no. 4, p. 042605, Apr. 2021, publisher: American Physical Society. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.103.042605

[291] F. B. Maciejewski, Z. Zimborás, and M. Oszmaniec, "Mitigation of readout noise in near-term quantum devices by classical post-processing based on detector tomography," *Quantum*, vol. 4, p. 257, Apr. 2020, publisher: Verein zur Förderung des Open Access Publizierens in den Quantenwissenschaften. [Online]. Available: https://quantum-journal.org/papers/q-2020-04-24-257/

[292] P. D. Nation, H. Kang, N. Sundaresan, and J. M. Gambetta, "Scalable Mitigation of Measurement Errors on Quantum Computers," *PRX Quantum*, vol. 2, no. 4, p. 040326, Nov. 2021, publisher: American Physical Society. [Online]. Available: https://link.aps.org/doi/10.1103/PRXQuantum.2.040326

[293] H. R. Grimsley, S. E. Economou, E. Barnes, and N. J. Mayhall, "An adaptive variational algorithm for exact molecular simulations on a

quantum computer," *Nature Communications*, vol. 10, no. 1, p. 3007, Jul. 2019, publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41467-019-10988-2

[294] A. J. McCaskey, Z. P. Parks, J. Jakowski, S. V. Moore, T. D. Morris, T. S. Humble, and R. C. Pooser, "Quantum chemistry as a benchmark for near-term quantum computers," *npj Quantum Information*, vol. 5, no. 1, pp. 1–8, Nov. 2019, publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41534-019-0209-0

[295] M. Foss-Feig, D. Hayes, J. M. Dreiling, C. Figgatt, J. P. Gaebler, S. A. Moses, J. M. Pino, and A. C. Potter, "Holographic quantum algorithms for simulating correlated spin systems," *Physical Review Research*, vol. 3, no. 3, p. 033002, Jul. 2021, publisher: American Physical Society. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevResearch.3.033002

[296] P. K. Barkoutsos, G. Nannicini, A. Robert, I. Tavernelli, and S. Woerner, "Improving Variational Quantum Optimization using CVaR," *Quantum*, vol. 4, p. 256, Apr. 2020, publisher: Verein zur Förderung des Open

Access Publizierens in den Quantenwissenschaften. [Online]. Available: https://quantum-journal.org/papers/q-2020-04-20-256/

[297] R. B. Griffiths and C.-S. Niu, "Semiclassical Fourier Transform for Quantum Computation," *Physical Review Letters*, vol. 76, no. 17, pp. 3228–3231, Apr. 1996, publisher: American Physical Society. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.76.3228

[298] M. Dobšíček, G. Johansson, V. Shumeiko, and G. Wendin, "Arbitrary accuracy iterative quantum phase estimation algorithm using a single ancillary qubit: A two-qubit benchmark," *Physical Review A*, vol. 76, no. 3, p. 030306, Sep. 2007, publisher: American Physical Society. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.76.030306

[299] T. E. O'Brien, B. Tarasinski, and B. M. Terhal, "Quantum phase estimation of multiple eigenvalues for small-scale (noisy) experiments," *New Journal of Physics*, vol. 21, no. 2, p. 023022, Feb. 2019, publisher: IOP Publishing. [Online]. Available: https://dx.doi.org/10.1088/1367-2630/aafb8e

[300] P. M. Q. Cruz, G. Catarina, R. Gautier, and J. Fernández-Rossier, "Optimizing quantum phase estimation for the simulation of Hamiltonian eigenstates," *Quantum Science and Technology*, vol. 5, no. 4, p. 044005, Aug. 2020, publisher: IOP Publishing. [Online]. Available: https://dx.doi.org/10.1088/2058-9565/abaa2c

[301] Y. Hindy, J. Pointing, M. Tolunay, S. Venkatarao, M. Motta, and J. A. Latone, "Quantum Mining Github Repository," Jul. 2021. [Online]. Available: https://github.com/quantum-mining/quantum-mining-publication