

Efficient Quantum Device Tuning using Machine Learning

Dominic T. Lennon

Linacre College
University of Oxford

*A thesis submitted for the degree of
Doctor of Philosophy*

Michaelmas 2021

Abstract

Abstract

Classical computation is foundational to the digital world in which we find ourselves. Quantum computation promises to overhaul the landscape of computation and with it many benefits to society as a whole are predicted. This thesis focuses on spin qubits, devices that hold promise for quantum computation, but are hampered by their control challenges. A large time sink is faced by those in industry and research alike; signals must be acquired, processed, and interpreted. From these signals decisions must be made. This thesis identifies a set of general automated methods to control and interpret devices. All methods are verified through extensive experimental demonstrations proving their efficacy in the real world.

Efficient Quantum Device Tuning using Machine Learning



Dominic T. Lennon

*Supervisors: Prof G.A.D. Briggs, Prof N. Ares, Prof E.A Laird &
Prof S.C. Benjamin*

Linacre College
University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Michaelmas 2021

I dedicate this work to my wife, Kimberley Lennon, on the occasion of our first wedding anniversary. She supported me without fail from the first day to the last and our bond has only grown stronger as a result.

Acknowledgements

Professional

The nature of this work is highly collaborative, algorithm development requires an interdisciplinary perspective while algorithm validation requires extensive testing in many systems. In my journey towards the production of this work I have had the privilege of working with many brilliant collaborators, as such I truly feel lucky.

I would first like to thank my supervisors: Prof Natalia Ares, Dr Edward laird, Prof Andrew Briggs, and Prof Simon Benjamin. Their input has been a consistent source of value and without their combined effort the work that follows would have been lacking. I would like to thank Natalia for taking the time to guide me and train me.

Machine learning techniques are at the heart of this thesis, featuring in every chapter. With this in mind I would like to thank: Dr Hyungil Moon, Prof Michael Osborne, Prof Dino Sejdinovic, and Dr James Kirkpatrick. Hyungil was fundamental to the success of these projects, his close collaboration was the driving force behind developing the active learning method in Chapter 3 and the original sampler for the algorithm in Chapter 4. Michael, and Dino guided me and Hyungil in the application of various machine learning techniques by highlighting useful methods and explaining proper practices. Finally, James worked closely myself and Hyungil, he provided weekly guidance and input helping us formalise the methods described in Chapter 4.

Throughout my journey I was fortunate enough to be able to supervise and support other students, some of whom contributed to the work in this thesis. I would like to thank Nina van Esbroeck who implemented the VAE and optimisation algorithm in Chapter 7 under my supervision. Her hard work and determination was key to the projects success. I would like to thank Brandon Severin who developed the modified coulomb peak classifier and aided with the running of experiments and data analysis used in Chapter 5. Finally, I would like to thank Sebastian Orbell for his help with developing the Bayesain inference used in section 3.9.

This work was highly dependent on collaborators who supplied devices and even permitted me use of their laboratory equipment. To all members of the Zumbühl group I would like to thank you for your warm hospitality on my visit in late 2019. I would like to thank Prof Dominic Zumbühl, Dr Leon Camenzind, Dr Liuqi Yu,

Dr Andreas Kuhlmann, Simon Geyer, Dr Floris Braakman, and Dr Florian Froning for supplying devices that allowed this project to get off the ground - these devices were used in Chapter 3, Chapter 4, Chapter 7, Chapter 5, and Section 6.1. Without this contribution the majority of this thesis would simply not have been possible. I would also like to further thank Dr Leon Camenzind for being my main point of contact with the Zumbühl group and for his valuable input. Secondly, I would like to thank Prof Georgios Katsaros and Dr Daniel Jirovec for allowing the use of their sample and experimental setup for part of the experiments that feature in Chapter 5. Thirdly, I would like to thank Dr Fernando Gonzalez-Zalba and Jacob Chittock-Wood for managing the experimental setup and for the derivation of the sensitivity measure used in Section 6.2.2. Finally, I would like to thank Prof Ferdinand Kuemmeth, Dr Anasua Chatterjee, Dr Joost van der Heijden, Fabrizio Barritta, and Dr Federico Fedele for so generously permitting me use of one of their fridges and supplying the devices used for experimental demonstrations in Chapter 8.

Personal

The contributions of others were not limited to bringing together the components that constitute the work. I would not be in this position today if it were not for the work of others who supported me emotionally and educationally.

I start by thanking my wife Kimberley Lennon, who has supported me vigilantly for the last 9 years. She helped financially support my pursuit of this work and provided daily emotional support. The number of ways in which she has enabled this work can not be listed and I certainly would not be where I am today without her.

I would like to thank my parents and brother. Throughout my early education I struggled with anything involving words and literature, (in fact I still do) eventually being diagnosed as dyslexic. My parents, apart from being responsible for raising me, highly valued my education. With their help I was able to negotiate a path through the education system that eventually led to this thesis. I can not begin to formalise how grateful I am for this.

Throughout my time in the Ares group many members joined and left. All contributed to the positive environment which I will sadly miss. Prof Natalia Ares, Dr Florian Vigneau, Dr Federico Fedele, Dr Yutian Wen, Dr Reuben Harding, Nina van Esbroeck, Dr Hyungil Moon, Jonas Schuff, Brandon Severin, Sebastian Orbell, Barnaby Van Straaten, Joseph Hickie, and David Craig to name a few.

Featured Publications

Part of the data presented in chapter 3 features in the following publication:

Lennon, D. T., Moon, H., Camenzind, L. C., Yu, L., Zumbühl, D. M., Briggs, G. A. D., ... & Ares, N. (2019). Efficiently measuring a quantum device using machine learning. *npj Quantum Information*, 5(1), 1-8.

The data presented in chapter 4 and part of the data presented in chapter 6 features in the following publication:

Moon, H., **Lennon, D. T.**, Kirkpatrick, J., van Esbroeck, N. M., Camenzind, L. C., Yu, L., ... & Ares, N. (2020). Machine learning enables completely automatic tuning of a quantum device faster than human experts. *Nature communications*, 11(1), 1-10.

The data presented in chapter 5 features in the following publication currently under review:

Severin, B., **Lennon, D. T.**, Camenzind, L. C., Vigneau, F., Fedele, F., Jirovec, D., ... & Ares, N. (2021). Cross-architecture Tuning of Silicon and SiGe-based Quantum Devices Using Machine Learning. *arXiv preprint* arXiv:2107.12975.

The data presented in chapter 7 features in the following publication:

van Esbroeck, N. M., **Lennon, D. T.**, Moon, H., Nguyen, V., Vigneau, F., Camenzind, L. C., ... & Ares, N. (2020). Quantum device fine-tuning using unsupervised embedding learning. *New Journal of Physics*, 22(9), 095003.

Data presented in chapter 8 and part of the data presented in chapter 6 will feature in two future publications that are currently in draft.

Abstract

Abstract

Classical computation is foundational to the digital world in which we find ourselves. Quantum computation promises to overhaul the landscape of computation and with it many benefits to society as a whole are predicted. This thesis focuses on spin qubits, devices that hold promise for quantum computation, but are hampered by their control challenges. A large time sink is faced by those in industry and research alike; signals must be acquired, processed, and interpreted. From these signals decisions must be made. This thesis identifies a set of general automated methods to control and interpret devices. All methods are verified through extensive experimental demonstrations proving their efficacy in the real world.

Contents

List of Abbreviations	xii
1 Introduction	1
1.1 The DiVincenzo criteria	3
1.2 The qubit	3
1.3 Spin qubits	5
1.4 Laterally defined quantum dots	6
1.5 Constant interaction model	7
1.5.1 Double quantum dots	9
1.6 Characteristics of laterally defined devices in GaAs/AlGaAs	11
1.7 Qubits in quantum dots	12
1.7.1 Single spin qubit	12
1.7.2 Singlet triplet qubit	13
1.7.3 Tuning, the current paradigm	14
1.8 Automation in spin qubits	17
1.8.1 Grid scanning	17
1.8.2 Signal processing	18
1.8.3 Virtual gates	19
1.8.4 Tuning	20
1.9 Relevant ML methods	21
1.9.1 Deep learning	21
1.10 Multi-armed bandits	22
1.10.1 Bayesian statistics	23
1.10.2 Gaussian processes	24
2 Methodology	26
2.1 Sample preparation	26
2.2 Classical electronics	28
2.3 Classical electronics and low level control	30
2.4 Algorithms	32

3	Efficient measurement of quantum devices	35
3.1	Device and experiment	35
3.2	Algorithmic framework	36
3.3	Predicting unobserved measurements with a deep generative model	38
3.4	Training data	40
3.5	Leveraging the trained Deep Generative model	41
3.6	Choosing optimally what to measure next	42
3.7	Efficient measurement of single quantum dot devices	44
3.7.1	Generalising the algorithm	49
3.8	Altering acquisition with segmentation for context aware decisions .	51
3.9	Physics aware active learning	54
3.9.1	Rabi oscillations	54
3.9.2	Predicting unobserved measurements with a physical model	55
3.10	Efficient measurement of Rabi oscillations	57
3.11	Discussion	58
4	Automatic coarse tuning of a quantum device	60
4.1	Devices	60
4.2	The gate voltage space	61
4.3	Hypersurface definition	63
4.4	A search based algorithm for double quantum dot tuning	64
4.5	Investigation stage	65
4.5.1	Search for the hypersurface	66
4.5.2	Coulomb peak check	67
4.5.3	Low resolution scan	68
4.5.4	High resolution scan	71
4.6	Sampling stage	71
4.6.1	Modelling the behavior of the device with a Gaussian process	71
4.6.2	Generating samples using the models	73
4.7	The pruning heuristic	75
4.8	Algorithm Hyperparameters	76
4.9	Analysis of expected tuning times and ablation study	84
4.9.1	Labelling procedure	84
4.9.2	Inferring the algorithm's performance	84
4.9.3	The Pure random baseline	86
4.9.4	Device tuning	86
4.9.5	Tuning by grouping gates	88
4.9.6	Ablation study	88
4.10	Discussion	91

5	Automatic coarse tuning of silicon devices	92
5.1	Devices	92
5.1.1	The FinFET	92
5.1.2	The Nanowire	93
5.1.3	The Heterostructure	94
5.2	Generalising the approach to different devices	95
5.2.1	Sampling modifications	95
5.2.2	Other modifications	97
5.3	Tuning Si and SiGe-based devices and analysis of expected tuning times	99
5.4	Analysis of score function for classification	101
5.5	Discussion	102
6	Characterising device variability	104
6.1	Variability via point-set registration	104
6.1.1	Devices	105
6.1.2	Sampling the hypersurface	105
6.1.3	Point set registration	105
6.2	Variability via operable region sampling and characterisation	108
6.2.1	Device	108
6.2.2	SET operation and sensitivity	109
6.2.3	The operable region of gate voltage space and the algorithm's goal	109
6.2.4	Investigation stage	110
6.2.5	Sampling stage	113
6.2.6	Unstructured point-sets	114
6.2.7	Alpha-shapes for volume reconstruction	116
6.2.8	Reproducibility	117
6.3	Discussion	119
7	Automatic fine tuning of a quantum device	121
7.1	The device	121
7.2	Algorithm's overview	121
7.3	Scoring bias triangles via embedding and distance metric	122
7.3.1	Low dimensional embedding via VAE	123
7.3.2	Scoring via distance metric	125
7.4	Greedy optimisation algorithm	128
7.5	Experimental optimisation of bias triangles	128
7.6	Discussion	131

8	Automatic tuning and optimisation of spin qubit readout	133
8.1	The device	134
8.2	Estimating the sensitivity of spin state readout	134
8.2.1	A parametric density for spin readout	137
8.2.2	Unsupervised learning via Expectation Maximisation	139
8.2.3	Validation of model training	140
8.2.4	Readout detection via model selection	141
8.2.5	Distribution overlap and optimal Bayes classification	143
8.2.6	Detection of switches	145
8.3	Complete methodology for evaluating readout fidelity	146
8.4	Optimisation of readout fidelity	147
8.4.1	Optimising the averaging time	147
8.5	Discussion	148
9	Conclusion	150
Appendices		
.1	CVAE architecture	155
.2	Applying BALD to a CVAE	157
.3	Noisy reconstructions	157
.4	Applying BALD to a physics model	158
.5	Simple tuner	158
.6	Gaussian process specification	159
.7	Tuning algorithm raw labels	160
.8	Thompson sampling for device tuning	161
.9	Additional stability diagrams	162
.10	VAE architecture	173
.11	Additional stability diagrams	174
.12	Bounding length scales	193
.13	Coulomb peak classifier	193
.14	Expectation maximisation function	194
	References	196

List of Abbreviations

1D, 2D, 3D	One-, two-, or three-dimensional
2DEG	Two-dimensional electron gas
2DHG	Two-dimensional hole gas
ANN	Artificial neural network
CNN	Convolutional neural network
MH	Metropolis Hastings
VAE	Variational auto encoder
CVAE	Conditional variational autoencoder
GAN	Generative adversarial networks
GP	Gaussian process
MCMC	Markov chain Monte Carlo
MLE	Maximum likelihood estimation
MAP	Maximum a posteriori estimation
EM	Expectation maximisation
GPU	Graphics processing unit
PSB	Pauli spin blockade
ESR	Electron paramagnetic resonance
EDSR	Electron dipole spin resonance
CMOS	Complementary metal–oxide–semiconductor
DC	Direct current
DAC	Digital to analogue converter
DMM	Digital multi meter
SET	Single electron transistor

1

Introduction

The modern era is fuelled by computation; it has become an integral part of every day life: automated decision making algorithms support industries like trading and logistics; information that amounts to a library's worth can be stored by a civilian in the equivalent volume of a book; and most carry a miniature computer around in their pocket at all times. Thanks to the immense scalability of the processes used to fabricate these devices, these incredibly complex devices can be made affordable. However, for all the success these devices have bought about they are built upon the concept of classical computation. Quantum computation in contrast promises to revolutionise what we can feasibly compute as well as the way in which we perform and implement that computation.

Since the initial proposal by Richard Feynman [1] of a computer that operates using the principles of quantum mechanics, the experimental pursuit of such a machine has been the focus of intense research. However, its realisation has so far remained elusive. Qubits - quantum two level systems form the fundamental building block of a quantum computer. A qubit is to a quantum computer what a bit is to a classical computer, its unit of information. Practically useful quantum computers will require millions of interacting, individually controllable qubits, and herein lies the difficulty. Due to the vast numbers of qubits required and the need for controlled interaction between qubits, the very technology that originally drove the classical

computation revolution, the semiconductor, has become a popular approach for building a quantum computer [2]. However, control of these semiconductor qubits proves immensely challenging; at the time of writing all semiconductor qubits require some form of manual control despite the field being over 20 years old. Paving a way for automated control with experimental demonstrations is the core of this thesis.

The thesis aims to address this by considering one core question:

"Can general purpose automated strategies for quantum device control be identified that work with real devices?"

My hope is that by highlighting general purpose strategies, we can make a toolbox of tricks and methods that can be employed by all researchers in academia and industry. They might not necessarily be reapplied in the ways that appear in this thesis, the specific demonstrations are to provide proof that the methods are robust.

Chapter 3 discusses an active learning methodology for selecting the most informative measurements to perform next. This method is applied to different common scenarios. Chapter 4 introduces the most developed algorithm featured in this thesis - a statistical search algorithm that performs a queue of experiments. It is demonstrated it can outperform a human baseline in the challenging problem of quantum device tuning. Chapter 5 further studies this algorithm by considering its applicability to multiple device architectures and material systems, demonstrating the algorithms generality. Chapter 6 leverages the same algorithm but for a different task; acquiring data so that post processing methods can be applied producing new metrics for quantifying device variability and performance. Chapter 7 introduces a method by which complex signals can be processed to produce a smooth score metric using very little human-labelled data. The approach is verified by using a simple optimiser to improve the score metric. Finally, chapter 8 introduces a method to evaluate quantum device readout with no labelled observations. The method is demonstrated by optimising experimental parameters to achieve improved readout fidelities.

1.1 The DiVincenzo criteria

The DiVincenzo criteria set out a set of key conditions a platform must achieve in order to be viable for quantum computation [3]. Quantum computing using semiconductors is such an attractive solution because it meets or is expected to meet all these goals. The relevant criteria are:

- 1) *A scalable physical system with well characterised qubits*
- 2) *The ability to initialise the state of the qubits to a simple fiducial state*
- 3) *Long relevant decoherence times, much longer than the gate operation time*
- 4) *A "universal" set of quantum gates*
- 5) *A qubit-specific measurement capability*

In essence the criteria mandates that implementations must be strongly decoupled from their environment, whilst being controllable, and thus well coupled to the mechanisms used to prepare, operate and readout the qubit. Physical systems that meet these criteria are hard to realise and are prone to errors. Error correction methods which combine many physical qubits into a single logical qubit have been proposed to counter this but estimates place the number of physical qubits needed to perform relevant algorithms at $10^6 - 10^8$ [4–6]. This increases the importance on requirement 1) which has yet to be demonstrated at this scale.

1.2 The qubit

A classical bit is a fundamental unit of information; it conveys the smallest unit of information by only taking on one of two possible states often referred to as 1 and 0. This classical information can be encoded in multiple ways, popular approaches in modern computation are: the direction of magnetisation of a thin film of ferromagnetic material; the charged or uncharged state of a capacitor; and the flow of current through a junction that can be turned on or off. A qubit however, can take on a linear combination of states 1 and 0, referred to as a superposition. Mathematically this means the state of a qubit $|\phi\rangle$ with basis states $|1\rangle$ and $|0\rangle$ can be represented in the following form:

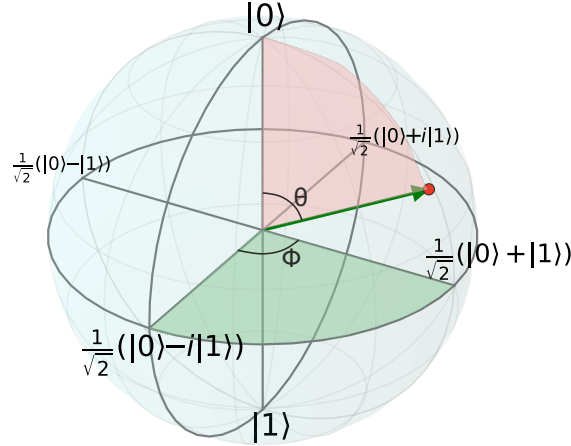


Figure 1.1: The Bloch sphere An annotated Bloch sphere.

$$|\phi\rangle = \alpha|0\rangle + \beta|1\rangle \quad (1.1)$$

for any complex α or β subject to the condition that the vector $\mathbf{v} = [\alpha, \beta]$ is normalised so ($\|\mathbf{v}\| = 1$). However, the state $|\phi\rangle$ can never be directly observed. Instead when attempting to observe the state of a qubit (measuring in the qubits basis) the qubit will be observed to be in either the $|0\rangle$ or $|1\rangle$ state. Whether the qubit assumes the $|0\rangle$ or the $|1\rangle$ state is purely probabilistic with the probabilities of either state being equal to $|\alpha|^2$ and $|\beta|^2$ respectively.

As $\mathbf{v}$ normalises to one and the global phase of the qubit is unimportant, every possible state of the qubit can be directly mapped onto the surface of a unit sphere (Fig. 1.1). This representation is referred to as a Bloch sphere and is helpful for visualising the state of a qubit.

In classical computation logic operations are performed on bits to carry out computation. All logic operations in classical computation can be thought of as flipping the state of bits, with flipping sometimes being conditional on the state of another bit. Similarly, logic operations are performed on qubits to carry out computation. A single qubit operation takes the qubit from one point on the Bloch sphere to another. The best known single qubit operations are the X, Y, and Z gates represented mathematically by the X, Y, and Z Pauli matrices,

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \quad (1.2)$$

Each of these operations performs a π rad rotation about its respective axis, i.e. X produces a rotation about the x axis mapping $|0\rangle$ to $|1\rangle$. For single qubit operation a set of gates that in combination allow any state on the Bloch sphere to be reached is required.

Signal qubit gates can be thought of as the quantum analogue of single bit operations. Two bit operations like the AND gate are also vital for classical computation. The quantum analogue of these are two qubit gates. They can be thought of as a rotation of a qubit conditional on the state of another qubit. An example of such a gate would be the controlled NOT gate.

1.3 Spin qubits

A qubit can be realised in any two level quantum system. The most classic example of a two level quantum system is that of a spin $\frac{1}{2}$ particle. Many popular physical qubit implementations leverage this. They fall into two broad categories - optically controlled and electrically controlled.

Optically controlled spin qubits include those realised in nitrogen vacancy centers in diamond [7]. Optically controlled implementations have seen great progress in recent years. However, they are not the focus of this thesis which concerns electrically controlled spin qubits.

Electrically controlled spin qubits have been successfully realised in quantum dots and impurities in silicon. Quantum dot devices are quite varied: One of the first approached was laterally-defined quantum dots in GaAs heterostructures [8]. More recently SiGe heterostructures have also been studied [9]. Similar demonstrations in nanotubes and nanowires consisting of various material systems exist [10–13]. Implementations based on CMOS technologies have been demonstrated [14, 15]. Finally, research into impurities in silicon has produced impressive results [16–18]. Each material system, and gate architecture brings with it benefits and drawbacks that themselves are subject to intense study. However, the core concepts remain the same and so this thesis is dedicated to developing methods that are applicable to all quantum dot based spin qubits.

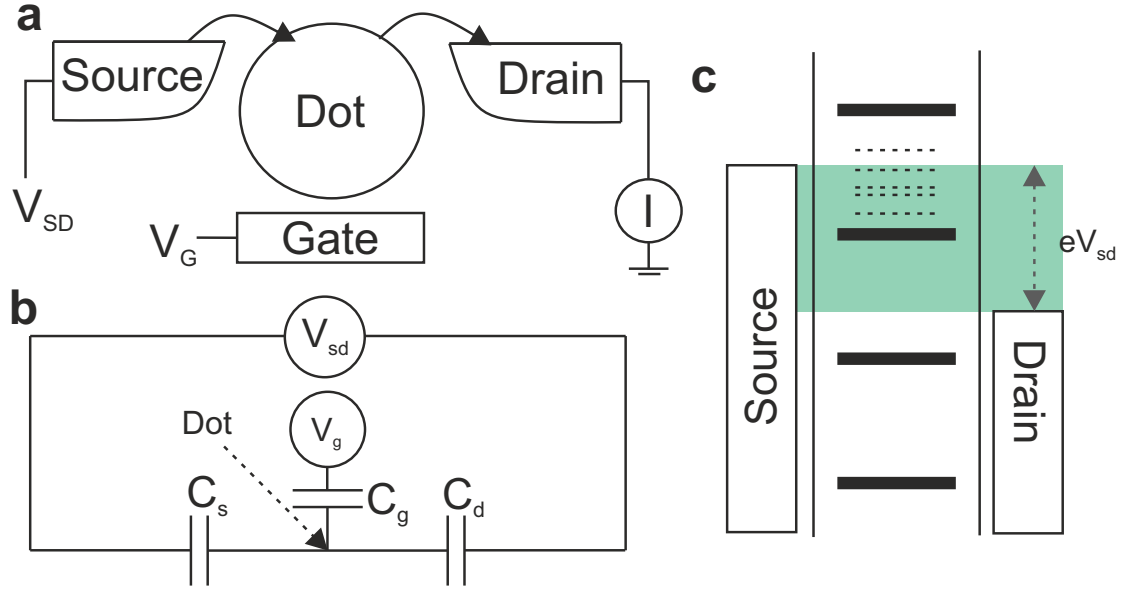


Figure 1.2: Constant interaction model schematic **a** Schematic of a quantum dot device. A quantum dot is coupled to two reservoirs, labelled source and drain and can be controlled via a nearby gate electrode that is biased to a voltage V_g . A voltage V_{sd} is applied across the reservoirs such that charge transport occurs. The current through the dot I is measured. **b** Constant interaction model approximation. To model the device in **a** the system is treated as a lumped element circuit with capacitors approximating the coupling of the junction labelled dot to the applied voltages. **c** Energy level diagram. Applying bias voltage V_{sd} produces bias window highlighted green. Electrostatic energy levels represented by solid black lines with excited states shown as dashed lines.

1.4 Laterally defined quantum dots

Quantum dots can be formed by creating a confinement potential to trap charge carriers like electrons. These devices are of comparable size to the fermi wavelength of the charge carrier allowing discrete energy levels to form much like an atom, thus Quantum dots are often referred to as artificial atoms. By adjusting the confinement potential of the quantum dot the number of confined charge carriers can be adjusted even to the point that the quantum dot contains only one charge carrier. To understand the behavior of the quantum dots we consider the physics of the constant interaction model; a simple yet powerful approximation.

1.5 Constant interaction model

The constant interaction model is useful for making predictions about the structure of current measurements with respect to applied voltages given physical parameters. Measurements that probe this, such as stability diagrams, are featured heavily throughout this thesis. This section introduced the model and then analyses the structures it produces with respect to voltage.

The model assumes that the quantum dot can be parameterised by a single capacitance C_Σ . This capacitance is the sum of individual capacitance terms each representing different interactions with the environment. For simplicity, I will introduce the case where a single quantum dot is coupled to a source and drain reservoir with a single gate electrode as depicted in Fig. 1.2a. In this case $C_\Sigma = C_s + C_d + C_g$, where C_s represents the source, C_d the drain, and C_g the gate electrode. Given that the total energy stored in a capacitor for charge Q , voltage V , and capacitance C is $U_c = \frac{1}{2} \frac{Q^2}{C}$ and that $Q = CV$, the total energy of the quantum dot is:

$$U(n) = \frac{(-|e|(n - n_0) + C_s V_s + C_d V_d + C_g V_g)^2}{2C_\Sigma} + \sum_{n=1}^N E_n(B), \quad (1.3)$$

where n_0 is the number of background charges. The voltages V_s , V_d , and V_g are applied to the source, drain, and gate, respectively. Finally E_n represents the single particle energy levels and is dependent on the confinement potential and the magnetic field. The electrochemical potential of the dot is defined as $\mu(n) \equiv U(n) - U(n - 1)$ hence:

$$\mu(n) = (n - n_0 - \frac{1}{2})E_C - \frac{E_C}{|e|}(C_s V_s + C_d V_d + C_g V_g) + E_N, \quad (1.4)$$

where $E_C = \frac{e^2}{C_\Sigma}$ is the charging energy. The first two terms are electrostatic and the last is chemical. The electrochemical potential is convenient to work with as it depends linearly on voltages. Tunneling through the dot is dependent on the alignment of electrochemical potentials. A bias window that allows electrons to tunnel from the source reservoir to the drain reservoir is created by the voltage

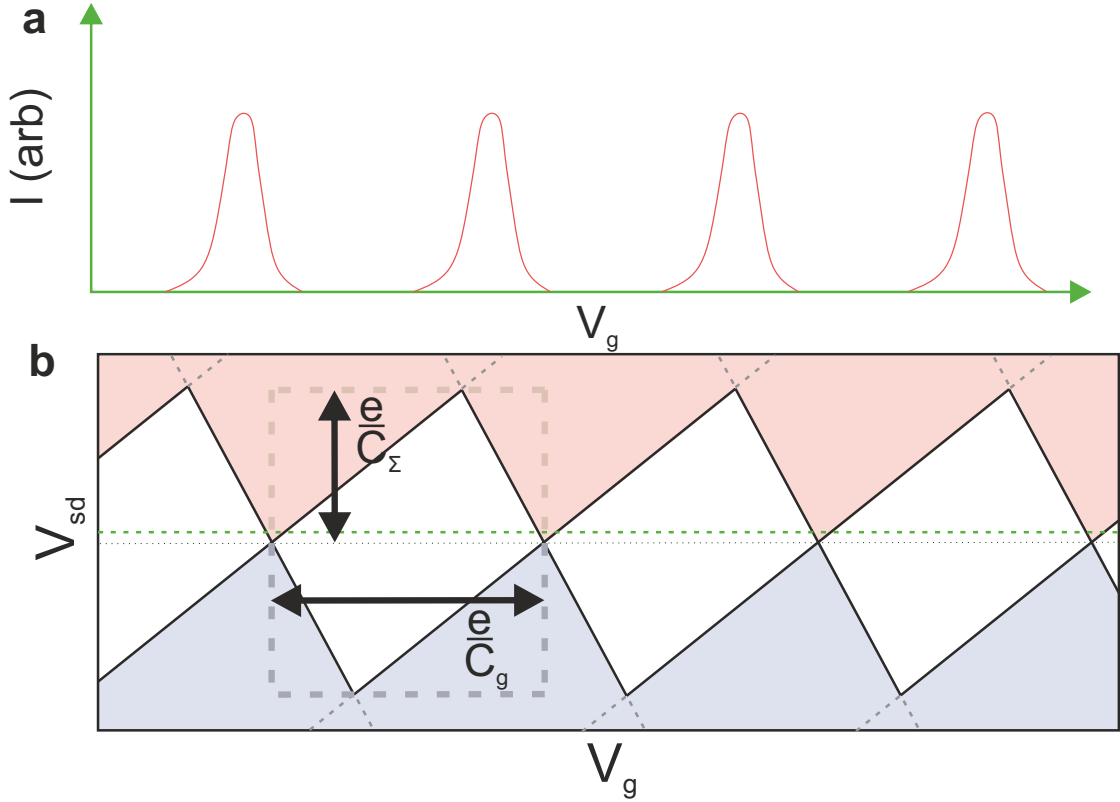


Figure 1.3: Schematics of coulomb blockade in single dots **a** Coulomb peaks at near zero V_{sd} . **b** Coulomb diamonds over the same domain of V_g . Coulomb diamonds with annotations showing cut responsible for **a**.

$V_{sd} = V_s - V_d$. This window has an associated energy $\mu_s - \mu_d = -|e|V_{sd}$. If an energy level of the dot sits within the window then single-electron transport is allowed. The energy levels of the dot are spaced apart by an energy $E_{\text{diff}} = E_C + \Delta E$, where ΔE is the difference between two discrete quantum energy levels. They can be shifted by altering the voltage V_g hence with our three gate voltages we can control the size of the bias window and whether an energy level sits within it.

For a small bias window $-|e|V_{sd} \ll E_C$ the majority of V_g values result in no energy level sitting within the bias window and thus no current is allowed to flow; this phenomena is referred to as Coulomb blockade. If an appropriate value of V_g is applied such that $\mu_s \approx \mu(n) \approx \mu_d$ then the Coulomb blockade is lifted and tunnelling is permitted. Only one energy level will be involved in electron transport if the bias window is small compared to ΔE (and small temperature).

We can vary V_g over some range of values while observing the transport current

I. Near-zero current is observed during Coulomb blockade. At the point where an energy level becomes aligned so that $\mu_s \approx \mu(n) \approx \mu_d$ for any given n the current rises and then falls sharply again with respect to V_g . This rise and fall of current forms a peak called a Coulomb peak (Fig 1.3a). The spacing between peaks provides information about the value of E_{diff} . The exact functional form $I(V_g)$ is dependent on many factors including tunnelling rates and temperature.

Increasing the applied bias voltage V_{sd} by some small amount relative to E_C will increase the width of peaks observed as a function of V_g . Increasing V_{sd} so that is of order E_C will increase the width of peaks until they merge.

Investigating I as a function of both V_{sd} and V_g produces a structure shown in Fig. 1.3b, this pattern is referred to as a Coulomb diamond. Within each diamond the dot is in Coulomb blockade so the number of electrons in the dot remains fixed. Each diamond corresponds to a different charge state. The transitions marked as black bold lines are predicted by the constant interaction model. The positively sloped transitions represent a solution to $\mu_s = \mu(n)$ giving a gradient of $\frac{C_g}{C_\Sigma}$ and the negatively sloped transitions represent the solution to $\mu_d = \mu(n)$ giving a gradient of $\frac{C_g}{C_s - C_\Sigma}$. For the same occupation number, intersection of transition lines takes place at a given positive value of V_{sd} ; this value corresponds to the charging energy in electron volts $\frac{E_C}{|e|}$. For a pair of solutions with occupation numbers that differ by one intersection takes place at $V_{sd} = 0$, the spacing in V_g between intersections of solutions of this type is $\frac{e}{C_g}$.

1.5.1 Double quantum dots

The constant interaction model can be simply extended to the case of two capacitively coupled quantum dots. In this case the electrochemical potential for dot 1 becomes

$$\mu_1(n_1, n_2) = U(n_1, n_2) - U(n_1 - 1, n_2) = \mathbf{E}^T \left(\frac{1}{|e|} \mathbf{C} \mathbf{V} + \mathbf{N} \right), \quad (1.5)$$

where

$$\mathbf{C} = \begin{bmatrix} C_{g11} & C_{g21} & C_s & 0 \\ C_{g12} & C_{g22} & 0 & C_d \end{bmatrix}, \mathbf{V} = \begin{bmatrix} V_{g1} \\ V_{g2} \\ V_s \\ V_d \end{bmatrix}, \mathbf{N} = \begin{bmatrix} n_1 - \frac{1}{2} \\ n_2 \end{bmatrix}, \mathbf{E} = \begin{bmatrix} E_C \\ E_{C_{id}} \end{bmatrix}. \quad (1.6)$$

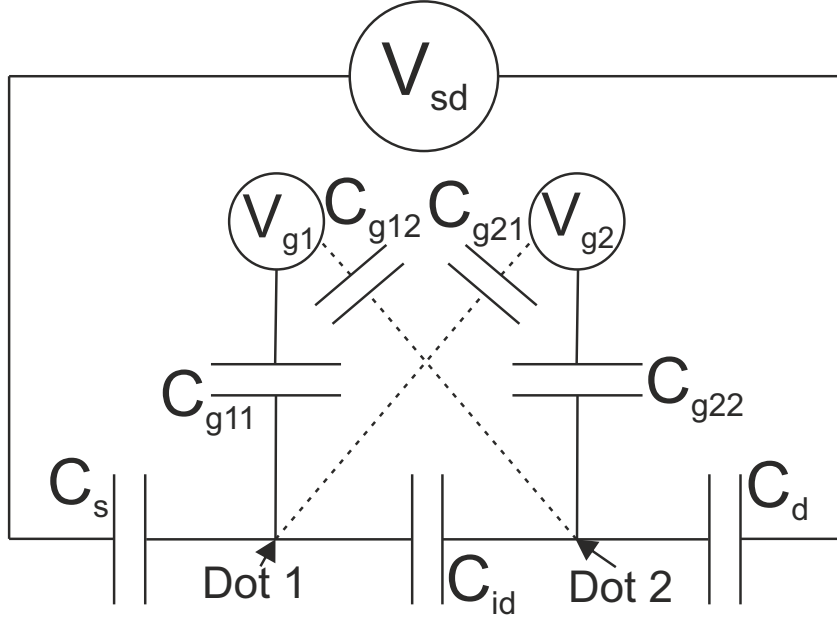


Figure 1.4: Circuit diagram of a double quantum dot Constant interaction model approximation of a double quantum dot. The double quantum dot is treated as a lumped element circuit with capacitors approximating the coupling of dot 1 and dot 2 to the applied voltages.

Values n_1 and n_2 are the charge occupation of dot 1 and dot 2. Matrix $\mathbf{C}$ is the full capacitance matrix containing: gate to dot capacitance terms C_{g11} and C_{g22} ; cross capacitance terms C_{g12} and C_{g21} ; and C_s and C_s source, drain capacitance terms. Where the first index of gate capacitance terms represents the gate and the second represents the dot. $\mathbf{V}$ contains gate voltages V_{g1} and V_{g2} . Finally, $\mathbf{E}$ contains the charging energy and the electrostatic coupling energy $E_{C_{id}}$.

Similarly to the first quantum dot, dot 2 $\mu_2(n_1, n_2) = U(n_1, n_2) - U(n_1, n_2 - 1)$ can be derived. If $C_{g11} = C_{g22} \neq 0$, $C_{g12} = C_{g21} = 0$, and $E_{C_{id}} = 0$ then for infinitesimally small V_{sd} the solution is a set of horizontal and vertical lines. If we make C_{g12} , C_{g21} , and $E_{C_{sd}}$ non zero, we instead observe a honeycomb pattern.

The constant interaction model can be further extended to systems of N coupled quantum dots by extending the matrix formalism used above.

1.6 Characteristics of laterally defined devices in GaAs/AlGaAs

I will briefly review the heterostructures. I select these types of devices to describe in detail as they feature prominently in this thesis.

The aim in these devices is to form an isolated region that contains a charge carrier. GaAs/AlGaAs heterostructures realise this through a combination of band-gap engineering and gating of the device. By stacking a layer of AlGaAs on top of a layer of GaAs and doping the AlGaAs layer with Si, free electrons are introduced. Due to the structure of the band-gap these free electrons accumulate at the interface with a ~ 10 nm width. Electrons in this plane can freely move and as such, this plane is typically referred to as a two dimensional electron gas (2DEG). Gate electrodes of Ti/Au are patterned lithographically on top of the heterostructures. By applying a negative DC voltages to the gate electrodes, regions of the 2DEG can be depleted of electrons. The architecture of gate electrodes that are patterned must be considered carefully so when voltages are applied quantum dots are formed. These quantum dots ideally should contain few electrons or even a single electron.

GaAs/AlGaAs heterostructures are a popular choice for such devices as they have a high mobility and low electron density giving electrons a large Fermi wavelength and a large screening length. The large screening length makes depletion of the 2DEG easier to achieve while the relatively large Fermi wavelength means larger quantum dots can still exhibit discrete energy levels.

Although a device might be designed to form a single or double quantum dot, that is not what happens for the majority of the applied gate voltage values. As long as no quantum dot is formed, the constant interaction model is not applicable; well defined tunnel barriers must first be formed. Forming tunnel barriers to form a quantum dot requires exploring the parameter space of gate voltage values.

1.7 Qubits in quantum dots

Multiple qubit implementations have been proposed for spins in quantum dots. I will detail the two that are relevant to this thesis.

1.7.1 Single spin qubit

Loss and DiVincenzo [1998] initially proposed a quantum computer based on isolated spins. This qubit, often referred to as the Loss-DiVincenzo qubit, uses a single spin to encode a qubit. The qubit basis states used are $|\uparrow\rangle$ and $|\downarrow\rangle$, corresponding to an alignment of the spin parallel or anti parallel to an applied magnetic field B_z . When applying a magnetic field, the energy level of the spin states split, with one energy level becoming lower in energy and the other becoming higher, this effect is called Zeeman splitting. The energy difference induced is proportional to the applied magnetic field $E_{split} = g\mu_B B_z$, where g is the g factor, and μ_B is the Bohr magneton. Operation of the qubit can be realised via electron spin resonance (ESR) or electron dipole spin resonance (EDSR). Operation via ESR [19] involves placing the qubit in an oscillating magnetic field with frequency $f = \frac{E_{split}}{h}$. Operation via EDSR [20] involves subjecting the qubit to an oscillating electric field while indirectly coupling the spin degree of freedom to the oscillating electric field - this can be achieved via spin-orbit interaction [20], hyperfine interaction [21], or on chip micromagnet [22]. By controlling the phase and duration of ESR/EDSR pulses, universal control of these spin qubits can be achieved.

Two methods of readout are employed. The first, referred to as Elzerman readout and demonstrated by [23], is based on energy-selective tunneling to a nearby reservoir. Readout can be performed by arranging the energy levels such that the higher energy electrochemical potential is greater than the reservoir's. The electron can then only tunnel if it is in the higher energy state, when tunneling takes place the charge state of the dot is altered. This change in charge state can be directly observed via a nearby charge sensor. An alternate form of readout utilises Pauli spin blockade, I will discuss this method in detail in the next section, as it is more commonly used with Singlet triplet qubits.

Using these techniques, single qubit gate, readout, and initialisation have all been demonstrated within the tolerances required for error correction [22–24]. Universal gate sets have been demonstrated in a two qubit processor proving the validity of the approach [25].

1.7.2 Singlet triplet qubit

The Singlet triplet (S-T) qubit is an alternative to the single spin qubit first proposed in a general form by Levy [2002] and demonstrated by Petta et al. [2005]. This implementation encodes a qubit in a pair of spin $\frac{1}{2}$ particles. The basis states for the qubit are those with spin projection $m_s = 0$, $|S\rangle = \frac{1}{\sqrt{2}}(|\uparrow\downarrow\rangle) - |\downarrow\uparrow\rangle$ and $|T_0\rangle = \frac{1}{\sqrt{2}}(|\uparrow\downarrow\rangle) + |\downarrow\uparrow\rangle$. Implementing a singlet-triplet qubit involves forming a double quantum dot. The qubit is typically operated close to the transition between the (2,0) - (1,1) or (0,2) - (1,1) charge states. In either the (2,0) or (0,2) charge states the Pauli exclusion principle plays an important role, and thus the exchange interaction $J(\epsilon)$ dominates the splitting of S and T_0 states. The detuning parameter controls the relative energy between the two charge states by the relative energy detuning ϵ between the two charge states ((2,0)-(1,1) or (0,2)-(1,1)). In the (1,1) state and when no inter-dot coupling is present the S state is degenerate with the T_0 state and the T_+ and T_- states (the triplet $m_s = 1, -1$ states). In the presence of inter-dot coupling, $J(\epsilon)$ removes the degeneracy. This effect becomes larger at small detuning (close to the inter-dot transition). For large detuning in (1,1), the states S and T become degenerate again.

These qubits can be operated all electrically by controlling the detuning ϵ . Changes in ϵ are applied at ns time scales, this process is referred to as pulsing. By entering the (2,0) or (0,2) charge state, an electron is loaded into the ground state, which corresponds to the $|S\rangle$ state. By pulsing along the detuning axis, the exchange interaction strength can be controlled, and even effectively turned off when the operation point is deep into the (1,1) transition. By positioning the operating point at specific values of detuning, $J(\epsilon)$ can be controlled in a manner that realises rotations about the qubits first axis. Rotations about the second

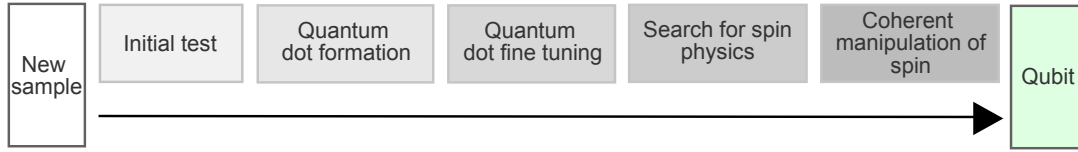


Figure 1.5: Graphic of steps Steps in the problem of device tuning and their approximate order.

control axis pose more of a challenge. The typical approach to gain control over the second axis is to place the dots in a magnetic field gradient either by fabricating an on chip micro magnet [27] or by relying on the intrinsic magnetic field supplied by nuclear spins in the host material [28].

For readout, Pauli spin blockade can be used for spin-to-charge conversion. This is achieved by pulsing gate voltages into the S-T splitting inside the (2,0) or (0,2) transition where transport is allowed for the $|S\rangle$ state but not the $|T_0\rangle$ state. This is because the $|T_0\rangle$ state would remain in the (1,1) charge state while a $|S\rangle$ would instead transition to a (2,0) or (0,2) charge state. This difference can be observed by a nearby charge sensor.

1.7.3 Tuning, the current paradigm

I will describe the general steps that comprise device tuning at the time of writing, with approximate times required for those steps to be completed by a human and a comment on how amenable to automation each step is. Device tuning is manual, relying on experimentalists who iterate between performing an experiment to obtain measurements, waiting for experiments to complete, reviewing measurements to infer the state of the device, and updating device parameters. When measurements are acquired using transport current, a full iteration of this process takes place on time scales of the order minutes to tens of minutes. Completely bringing a device into its intended operation requires many iterations of this process. It is not uncommon for it to take hundreds or even thousands of iterations. As so many iterations are required the full task of bringing a device into operation is separated into different steps (see Figure 1.5). These steps are: initial tests

(see section 1.7.3); quantum dot formation (see section 1.7.3); quantum dot fine tuning (see section 1.7.3); search for spin physics (see section 1.7.3); and coherent manipulation of spin (see section 1.7.3). Although I present these steps as being sequential this is not necessarily the case; it is quite common for an experimentalist to try quantum dot fine tuning if coherent manipulation is proving unsuccessful. It is also important to consider that any form of automation is valuable for quantum device tuning as it allows experimentalists to work on other problems.

Initial test

Once a device is ready for measurement initial tests are performed; these are used to quickly determine if the device can be operated. During this step the experimentalists first checks that the device behaves like a resistor and has appropriate resistance (appropriate resistance values depend heavily the type of device and experimental setup). They then test that there is no leakage between pairs of gates or the source/drain reservoirs. They finally check that the device is gateable, by demonstrating that there are regions of gate voltage space where transport current can flow and regions where it can not. For a human operator this process can take ten minuets to an hour depending on the number of gate electrodes that have to be checked. These tasks are simple to automate using linear regression and thresholding methods but are not described in this thesis.

Quantum dot formation

The second step is quantum dot formation; used to bring the device into the correct operating regime. During this step the experimentalists iterates between performing grid scans with specific gates and altering gate voltage parameters. The process required to achieve this step is very dependent on the type of device being used as demonstrated in chapter 7. For a human operator this process can take an hour to multiple days. This step is very difficult to automate because the parameter space is large and devices vary significantly, chapters 4 and 7 demonstrate that automation of this step is possible.

Quantum dot fine tuning

The third step is quantum dot fine tuning; used to bring the formed quantum dots into a regime where spin physics can be observed and manipulated. During this step the experimentalists iterates between performing grid scans with specific gates and altering gate voltage parameters. For a human operator this process can take tens of minutes to a day depending on how easy it is to control tunnel couplings without altering other device behavior. For example the device displayed in figure 4.1 is harder to fine tune than the one displayed in figure 5.1 because of its gate architecture. This step is challenging to automate and is covered in chapter 6.

Search for spin physics

The fourth step is a search for indications of spin physics. During this step the experimentalist iterates between performing grid scans with specific gates and altering gate voltage parameters whilst also adjusting the bias voltage and externally applied magnetic field. The process required to achieve this step depends on whether the device is capable of depleting its quantum dots down to the last charge carrier. In the situation where this is possible charge states can be counted making the process more reliable. In the situation where this is not possible many inter-dot transitions must be explored [29]. This process can take tens of minutes to tens of days depending on which approach has to be used. This step is challenging to automate because signals that are indicative of spin physics are very subtle and only occupy a small portion of device parameter space, chapter 8 introduced an approach that could be used for this purpose.

Coherent manipulation of spin

The final step that must be performed is the coherent manipulation of spin. During this step the experimentalist iterates between perform sequences of pulses while measuring the state of the device, and designing pulse sequences or finely adjusting gate voltages. This step is very dependant on the type of spin qubit that the experimentalist is aiming to achieve (see 1.7.1 and 1.7.2 for examples) and the

tuning achieved in previous steps. This process can take hours to days. This step is challenging to automate in part because it is so reliant on the successful completion of all prior steps.

1.8 Automation in spin qubits

Automation in spin qubits had been relatively unstudied until recent years. Its importance for scalability is becoming clear. Even with renewed interest few parts of the process of operating spin qubits have been automated due the complexity of various parts of the tuning process.

1.8.1 Grid scanning

Grid scanning techniques are the backbone of characterisation techniques, and while basic, they represent a form of automation. As touched upon in section 1.5, information about the state of a quantum dot is contained in the structure of observables like transport current I with respect to other device parameters like gate voltages. To interpret devices, uniformly spaced grids of measurements are acquired in a one or two dimensional slice of the parameter space.

A one dimensional grid scan often referred to as a trace, sweep, or ray, changes a variable while measuring changes in I . That is a single variables V_{fast} is selected and stepped by δv_{fast} ; this is performed N times. At each iteration of the stepping procedure n , the state of the observable I_n is stored.

The result of a one dimensional grid scan is an array shape N containing measurements of I . This can be plotted as a line plot or scatter plot on a graph.

A two dimensional grid scan requires two variables V_{fast} and V_{slow} , selected from the set of all variables. The grid scan selects a number of one dimensional grid scans to be made M and a value to step V_{slow} by each iteration δv_{slow} . The grid scan proceeds by following a raster pattern like that displayed in Fig. 1.6a.

A two dimensional grid scan produces an $N \times M$ grid of measurements of I . These are displayed or processed as single channel colour images on different colour-maps. These type of measurements feature heavily throughout the rest of this thesis.

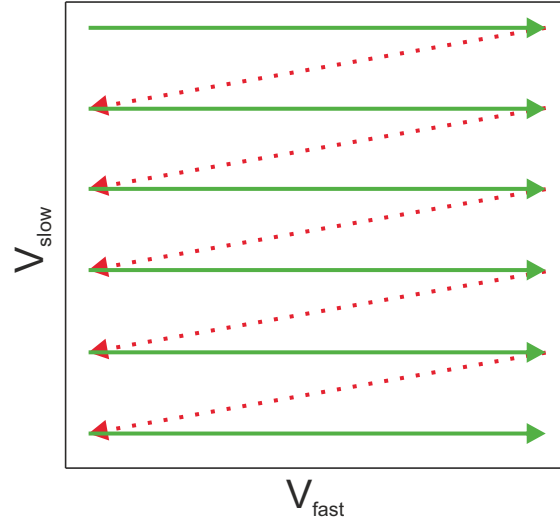


Figure 1.6: Grid scan sequence Grid scan sequence. Grid scan sequence marking how V_{fast} and V_{slow} evolve over the duration of a raster pattern grid scan. While traversing solid green arrows measurements are collected uniformly with respect to V_{fast} ; no measurements are acquired while traversing dashed red arrows.

1.8.2 Signal processing

Information about the charge state of the device is contained in measurements of a signal, such as current, as a function of device parameters. The use of signal processing techniques to extract useful information about a device has been demonstrated. The methods fall into two broad categories based on the computer science methods they employ, classical computer vision methods and machine learning methods.

Computer vision methods have been used to detect edges between charge states in stability diagrams corresponding to five coupled quantum dots [30]. This allowed the authors to match the measurements to the state of a simulator. The Hough transformation [31] has been applied multiple times [32, 33] to extract transition lines in stability diagrams, providing information about the gate capacitance matrix. Gabor filtering [34] has been applied to detect Coulomb peaks in stability diagrams [35]. Finally, custom convolutions have been designed to detect inter dot transitions [35].

Use of machine learning methods has focused on the detection of charge states. Implementations range from those based on convolutional neural net-

works [36–39] (CNN’s), to those based on classical machine learning methods, such as support vector machines, multi-layer perceptions, Gaussian processes, and k-nearest neighbor [40].

1.8.3 Virtual gates

Virtual gates are a powerful tool that is popular in the space of automating the control of spin qubits. The aim is to define a new set of gate voltages that are a linear combination of the applied gate voltages, creating in the process a virtual voltage space. In this virtual space, physical properties of the device are orthogonal, making the device easier to control algorithmically. The linear transform is a natural solution to this problem as off-diagonal elements in the gate capacitance matrix of equation 1.6 give rise to shear transformations in the gate voltage space. For the case of the two gates $V_g = [V_{g1}, V_{g2}]^T$ defined in section 1.5.1 the gate capacitance matrix is

$$\mathbf{C}_g = \begin{bmatrix} C_{g11} & C_{g21} \\ C_{g12} & C_{g22} \end{bmatrix}. \quad (1.7)$$

To approximate this matrix, two dimensional grid scans are acquired and parameters are extracted either by using the constant interaction model to fit measurements [41, 42] or simple linear fitting [43]. Other approaches to obtaining elements of the cross capacitance matrix from raster scans have also been demonstrated based upon signal processing and machine learning techniques [33]. Once the matrix has been populated it is then inverted to find inputs that make the gate capacitance matrix diagonal,

$$\mathbf{V}_g^\delta = \mathbf{C}_g \mathbf{V}_g. \quad (1.8)$$

Where $\mathbf{V}_g^\delta$ represents voltages in the virtual space. Controlling the physical parameters of the device is simpler in this virtual space and automation methods can be built directly upon this. Using this approach has enabled the fine control of large arrays for purposes such as quantum simulation [44].

The elements in the matrix $\mathbf{C}_g$ will scale $O(n^2)$ with the number of quantum dots in the system making the full characterisation of this matrix (especially through

traditional grid scanning) challenging. However, experiments have indicated that this matrix is sparse [43] in a linear chain of coupled quantum dots, so it is likely that it will have a far better scaling.

So far I have described virtual gates that are a combination of all physical gates that act to change the electrostatic potential of a dot. A gate that performs this role is referred to as a plunger gate, hence these virtual gates are called virtual plunger gates. Gates can also alter the tunnel barriers between neighboring quantum dots and source or drain. Gates of this type are referred to as barrier gates. Virtual barrier gates are a subject of research. Similarly to virtual plungers, virtual barriers involve learning linear transformations by performing grid scans and applying fits to extract tunnel couplings [45]. Virtual tunnel barriers remain harder to extract than virtual plunger gates as they require more measurements, more fitting, and are not directly linearly related to applied gate voltages.

The constant interaction model is only an approximation of a quantum dot device so this approach has significant limitations. The relationship between applied gate voltages and the electrostatic potentials of a set of quantum dots can vary none linearly due to the capacitance matrix approximation of the system also being a function of gate voltage. This limitation means virtual gates see heavy use in fine tuning applications where the approximation of the capacitance matrix holds.

1.8.4 Tuning

The process of tuning requires iteratively collecting information about the state of the device and then updating experimental parameters. The aim is to bring the device into a desirable state, typically one that renders it operational. Using a set of heuristic decision rules that alter parameters according to predetermined methods and classical computer vision signal processing methods mentioned in section 1.8.2, Baart et al. [2016] demonstrated the tuning of a device into a double dot regime and in particular to a (1,1) charge state, in 12 hours. This was the first demonstration of automated tuning. However, the process is stated to not be totally reliable, it requires some manual input, and makes many assumptions about

the specific gate architecture. Since this, Darulová et al. [2020] have been able to leverage different decision rules and machine learning signal processing methods to demonstrate this approach can be made faster and more reliable. However, significant assumptions about the gate architecture were still made. Finally, Volk et al. [2019] demonstrated that once a few quantum dots have been tuned, virtual gates can be applied to tune additional barrier gates one by one.

Kalantre et al. [2019] demonstrated that using a numerical optimiser and machine learning signal processing loss function, a device can be tuned from a single dot regime into a double dot regime or visa versa depending on the desire of the operator. This demonstration however operates on a subset of all gates voltages in a predetermined region of interest. Hence, it is a solution that can only be applied to a device operated close in gate voltage space to the single/double quantum dot regime.

Finally, using the Kalman filtering [46] method developed for classical control theory, it was demonstrated that tunnel rates in an already tuned double quantum dot can be adjusted according to the user's desire [47]. Other demonstrations of the tuning of tunnelling rates leveraging virtual gates exist [41–43, 45].

1.9 Relevant ML methods

I will briefly touch upon some methods used in this thesis.

1.9.1 Deep learning

The term Deep learning refers to a subset of machine learning methods that rely on Artificial Neural Network (ANN) based models. ‘Deep’ models consist of many network layers. Although ANN’s have been studied for decades [48] only recently have they seen extensive use.

Renewed interest was initiated by large improvements in state-of-the-art performance of ANN based models across many applications during the early 2010’s [49–52]. [49] presents the fundamental approach for deep learning over the last decade: large datasets to train on; stochastic gradient decent for learning; hardware

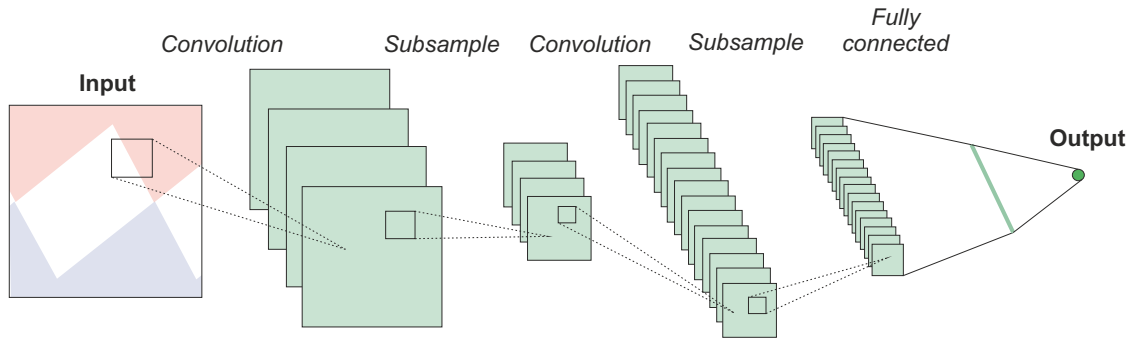


Figure 1.7: Illustration a simple convolutional neural network

acceleration via graphical processing unit (GPU) for faster training; and data augmentation techniques to improve model generalisation.

ANN based models come in different forms. Most relevant to this thesis is the convolutional neural network (CNN) [53]. The CNN architecture (Fig. 1.7) was inspired by the the structure of the visual cortex [53] and is a popular choice for computer vision tasks. The primary component of a CNN architecture is the *convolution* layer. This layer takes as input an array shape $C_d \times W_d \times H_d$ array where C_d is the number of channels, W_d is the width of the input, and H_d is the height of the input. The input is convolved with K_d kernels shape, $C_d \times N_d \times M_d$, where N_d is the width of the kernel and M_d is the height of the kernel. This produces an output array shape $K_d \times W_d - \frac{(N_d-1)}{2} \times H_d - \frac{(M_d-1)}{2}$ (in the case N_d and M_d are odd). This output is passed through a nonlinear activation function. The result of this becomes the input shape $C_{d+1} \times W_{d+1} \times H_{d+1}$ for the next layer. Layers that perform *sub-sampling* are applied to further reduce the dimensionality of the input. Finally, *fully connected* layers are employed to produce the desired output shape. During training, convolution parameters, fully connected parameters, and in some cases subsampling parameters are iteratively updated using gradient descent.

1.10 Multi-armed bandits

In machine learning multi-armed bandit refers to a fundamental class of problems. They have been the focus of research for over a century [54–56]. The problem considered is how an agent should allocate actions to gain maximum reward

considering fixed resources. The problem formulation is as follows: the agent is given $k \in \{1, \dots, K\}$ actions that can be performed, each action is associated with a random variable denoting the reward $\bar{R} = \{R_k\}_{k=1, \dots, K}$ that is unknown to the agent. At a given iteration i the agent performs some action k_i and receives a randomly sampled reward $r_i \leftarrow R_{k_i}$. The objective of the agent is to maximise the accumulated reward $\hat{r} = \sum_{i=1}^N r_i$ over N iterations. The difficulty of this problem arises from the trade-off between exploration and exploitation. The agent must trial different actions while also selecting intelligently so that not too much reward is lost.

There are some common solutions to these problems. Epsilon greedy methods which select by default the predicted best action and randomly select an action with small probability ϵ . Upper confidence bound methods [57, 58] which include predictions of action performance with a measure of uncertainty to make decisions. Probability matching or Thompson sampling [59, 60] builds a statistical model of rewards and selects an action according to a randomly drawn belief.

1.10.1 Bayesian statistics

Bayes rule is a formula that forms the foundation of Bayesian statistics. Formally it is:

$$P(A|B) = \frac{P(B|A)P(A)}{p(B)} \quad (1.9)$$

It can be observed that it provides a way of inverting a conditional probability distribution. If one has a model that produces predictions of B for a given A , $P(B|A)$ (often referred to as the likelihood) bayes rule can be used to find likely values of A given observations of B . Even more conveniently it offers a way to weight the possibilities by our belief of what values A can take, this is captured in $P(A)$ (often referred to as the prior). So that the distribution $P(A|B)$ is a valid probability distribution it is normalised by the term $P(B)$ (often referred to as the evidence). The $P(B)$ term is often unknown or hard to estimate as it requires the marginalisation of the model over all A this typically means methods like Bayesian inference must approximate $P(A|B)$ using the knowledge that it is

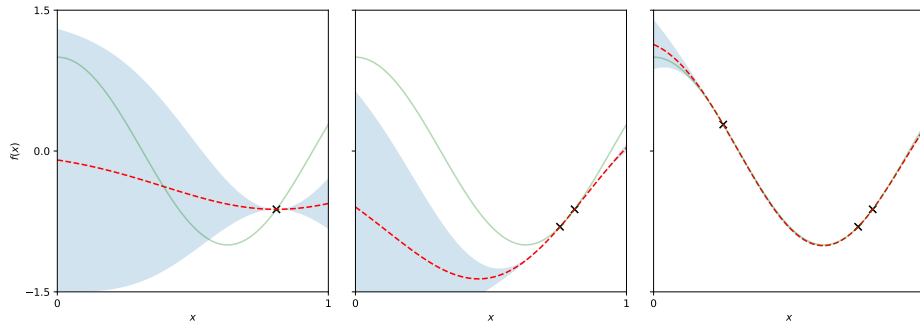


Figure 1.8: Example Gaussian process in 1D Plots shows Gaussian process prediction as a function of the number of points used for training data. Black crosses represent training data, red dashed line represents posterior mean prediction, blue shaded area represents GP uncertainty, and solid green line represents ground truth.

proportional to $P(B|A)P(A)$. Methods like Metropolis Hastings [61] or variational inference [62] are popular choices for solving this problem.

1.10.2 Gaussian processes

A Gaussian process [63] is a popular choice of machine learning model for Bayesian statistics that I use throughout this thesis. Gaussian process can be used for regression and classification tasks. They are a type of supervised, non-parametric, generative model that is capable of accurately quantifying its uncertainty for a given input (see figure 1.8 for example). They are mathematically tractable to work with allowing theoretical analysis. A Gaussian process is defined as a set of random variables, they model a collection of functions, so for a given input x the function $f(x)$ as a distribution.

$$f(x) \sim GP(m(x), K(x, x')) \quad (1.10)$$

This distribution is dependent on $m(x)$ the mean function and $K(x, x')$ the Gaussian processes kernel. The kernel $K(x, x')$ must be selected with care as it defines the correlation between different values of x . Common kernel choices are the exponentiated quadratic kernel, the rational quadratic kernel, or the matern family of kernels. Kernels introduce hyperparameters defining their length-scales. Large length scales result in high correlation over large distances. These hyperparameters are typically

inferred from data by finding the maximum a posteriori probability (MAP) estimate using a nonlinear optimiser and a prior over reasonable hyperparameter values.

2

Methodology

In this chapter I discuss the interface between devices and algorithms. Experiments in Chapters 3, 4, 7 and section 6.1 took place at the University of Oxford. Experiments in Chapter 5 took place at University of Basel in collaboration with the Zumbühl group and at IST Austria in collaboration with the Katsaros group. Experiments in section 6.2 took place at University College London during an internship at Quantum Motion Technologies. Experiments in Chapter 8 took place at QDev University of Copenhagen in collaboration with the Kuemmeth group. The focus of this chapter is on how algorithms were verified experimentally and so this section is not intended to describe the classical and quantum hardware extensively. Figure 2.1 provides a high level overview of the interface between devices and algorithms that I used.

2.1 Sample preparation

Samples must first be prepared before devices can be measured and controlled in automation experiments. The device is connected to a sample board (Fig. 2.2a). The sample board serves to interface the device with the signals that are sent to it. The sample is fixed in place with polymethyl methacrylate (PMMA). PMMA is strong enough to keep the sample in place, but weak enough for the sample to

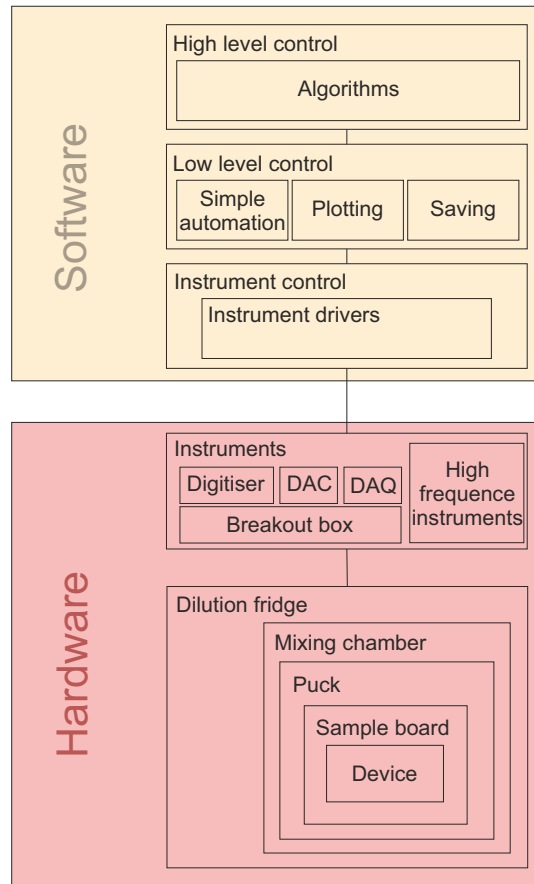


Figure 2.1: Interfacing algorithms and quantum devices. The interface between the algorithms and the device split into the main components of hardware and software.

be removed without damage. The device is then electrically connected to pads on the sample board using a wire bonder. Certain fridge lines and sample board lines have different capabilities, such as bias tees to combine high-frequency and DC signals. Bonding must be planned carefully so that inputs to the device can be electrically connected to classical hardware with the appropriate capabilities.

These experiments require mK temperatures to operate. At temperatures higher than this thermal noise can supply enough energy for electrons to enter a quantum dot. Experimentally these low temperatures are provided by a dilution refrigerator [64]. Once the device is bonded to the sample board, it can be loaded into a puck (Fig. 2.2b). The puck is a small container used to load the sample in and out of our cryo-free dilution refrigerator. It contains the sample board and it may contain other elements of the experimental setup that require mK temperatures

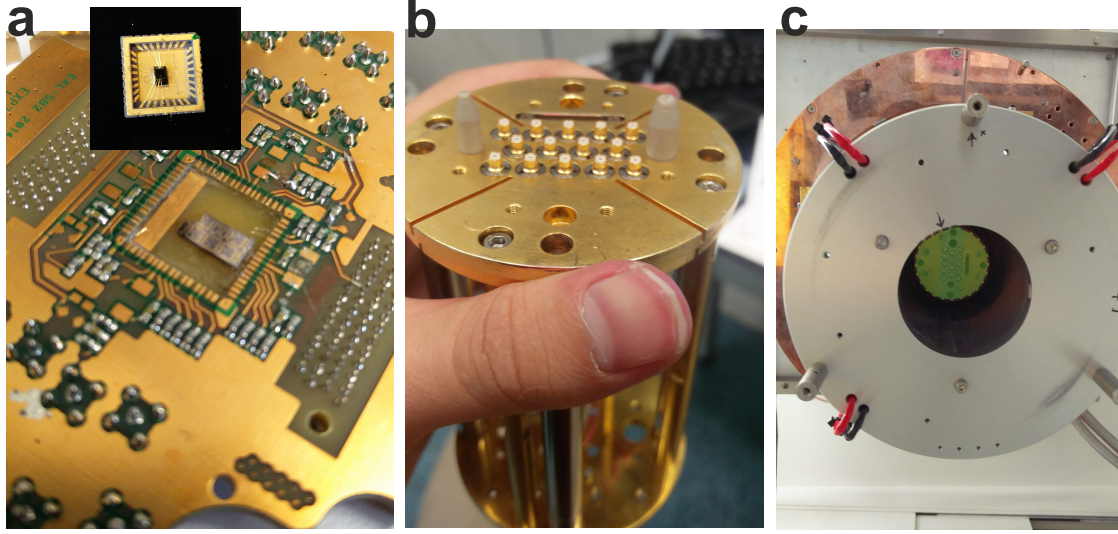


Figure 2.2: Sample preparation. **a** Sample placed in a sample board. The sample is fixed and bonded to a sample board ready for loading into a dilution refrigerator. **b** A puck containing a sample board. **c** A view of the cold finger. View from below the mixing chamber of the dilution refrigerator where the puck is connected (segmented green).

to operate. The face plate of the puck makes good contact with the face plate of the cold finger (Fig. 2.2c) which is attached to the dilution refrigerator's mixing chamber allowing it to thermalise to mK temperatures. The connections on the faceplate of the puck directly mate with those of the dilution refrigerator so that the puck is electrically connected to the dilution fridge lines.

2.2 Classical electronics

We must generate and readout signals to control and operate the device in the dilution refrigerator. All DC lines in the fridge exit out of the top of and are routed via a fisher cable to a 48 channel break out box (Fig. 2.3a). Each DC line has its own BNC connector on the breakout box which can be grounded or floating depending on the state of a switch.

To generate the voltage applied to a given DC line, we use a 16 channel 16-bit digital to analog converter (DAC) from TU Delft (Fig. 2.3b). The DACs have a 4V range with the capability to be run in an positive, negative, or bipolar mode. Throughout this work they are operated in the bipolar mode giving a range of -2V to 2V. As the DACs count with 16 bit, they have a voltage resolution of

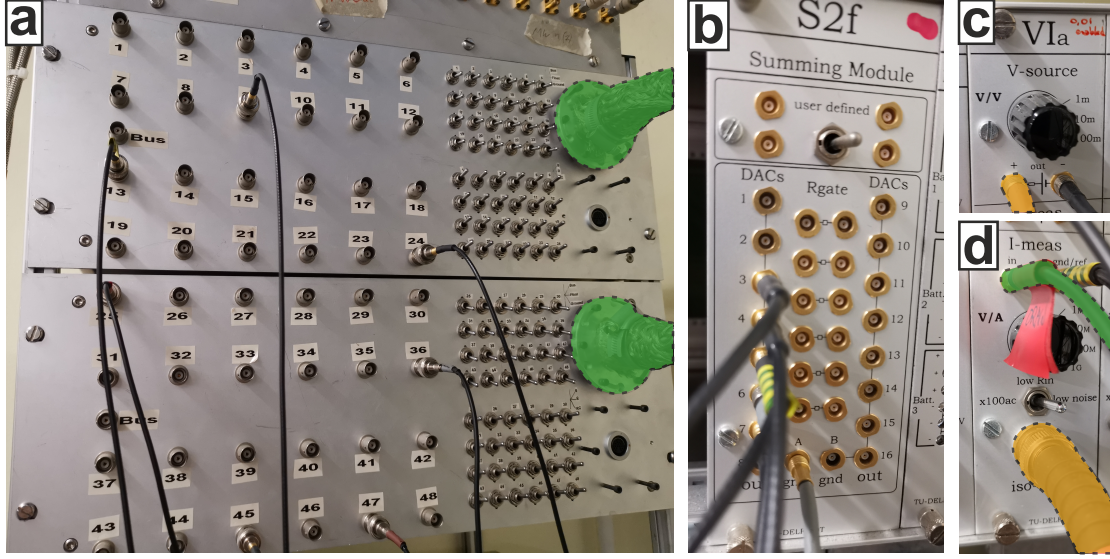


Figure 2.3: Electronics. **a** 48 line break out box. A 48-line breakout box with BNC connectors for each line, is connected via fisher cables (segmented in green) to the top of the fridge. Each line can be grounded or floating conditional on a switch. **b** DAC outputs. A 16-channel DAC with 2 ground ports and test resistors. **c** Voltage divider. A voltage divider for DAC channel 1 in **b**. The output is segmented in orange. **d** Current amplifier. A current amplifier for the signals received from the breakout box (segmented in green). The output to the digitiser is segmented in orange.

$60 \mu V$. We use a voltage divider (Fig. 2.3c) to achieve smaller voltage resolutions. Voltage division factors typically used in this work range from $\frac{1}{1000} \frac{V}{V}$ to $\frac{1}{100} \frac{V}{V}$. The DAC is controlled via a laboratory PC that communicates over a fiber optic cable. Serial commands are sent and received from the DAC allowing relatively fast changes ($\sim 10ms$) to be made to the output voltages.

Finally, to measure the current flowing through the device, the drain is connected to ground with an in line low noise current amplifier (Fig. 2.3d). The current amplifiers gain is adjustable but for the majority of experiments in this thesis $1 \times 10^{-9} \frac{A}{V}$ was used. Measurements of the current amplifier output were performed by a PCIe digitiser card. This digitiser card has a maximum sample rate of $150 \frac{MSP}{s}$. When performing transport measurements, we acquire a buffer of 1028 samples at a sampling rate of $1 \frac{MSP}{s}$ and the digitiser card returns the average value of the buffer at the end of sampling. To readout a signal in real-time with the digitiser, we configure it to measure a buffer of the appropriate size. When acquisition is desired, a trigger signal is generated indicating to the digitiser to fill the buffer. The buffer

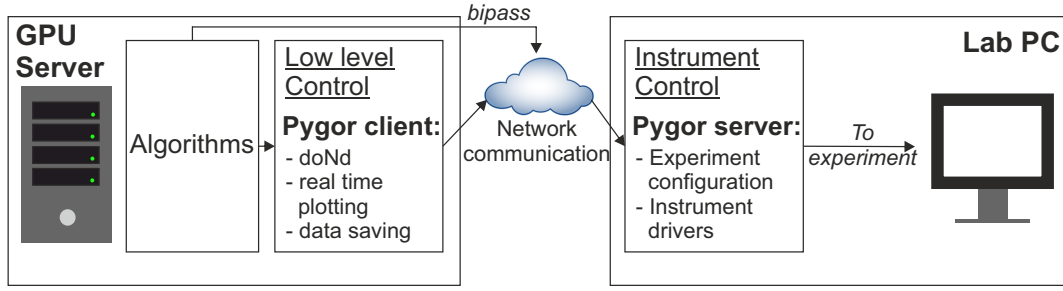


Figure 2.4: Pygor framework A diagram detailing the role of the *Pygor* framework and how it sits in the software part of the interfacing stack shown in Fig. 2.1.

is then transferred from the digitisers on chip memory to the PC's RAM for further manipulation such as averaging. The digitiser is controlled through an SDK library.

2.3 Classical electronics and low level control

The classical electronics is controlled programatically using the python programming language. For experiments in Chapters 3, 4, 7, 6, and the first half of 5, I developed a custom experiment control library called *Pygor*. An overview of where *Pygor* sits in respect to the software interface is shown in Fig. 2.4. The purpose of the *Pygor* framework is to separate the bare-bones instrument level control from the low level, human parsable, control of the experiment. The framework allows for these two control instances to be performed on different computers. *Pygor-client* allows any PC such as a work desktop, personal laptop, or powerful server, to control the experiment at the level of executing grid scans. *Pygor-server*, however, runs on the lab PC, executing the instructions.

So that the following can be understood in context I will define some programming related terminology. A class is a predefined structure defined by a programmer, it defines methods and properties, commonly (but not always) this will represent some concept in the real world like a person for example. Properties are attributes that a class can have, with regards to our example this could be weight of a person. A method is a functions that a class is able to perform, in our example a person could have an eat method that increments a weight property and reduces a hunger property. An instance is a specific instance of a class, for example

this would represent a specific person with a specific weight. Finally, inheritance allows a class to take on properties and methods of another class while adding to it or altering it in some way, for example the person class could inherit from an animal class which defines the eat method.

Instrument level control, such as that performed by *Pygor-server*, involves communicating with instruments. Instrument drivers are responsible for this. They take the native language that instruments understand and wrap it inside useful programming methods. *Pygor* implements its own drivers. The instrument drivers utilise *pyvisa*, *pyserial*, and the AlazarTech *ATS-SDK* (for the digitiser card) to interface with experiments. All drivers inherit either the *acquisition* or *control* class which themselves inherit from the *instrument* class. All acquisition instruments implement a *measure* method which returns an array of data. This method is called when acquiring measurements of current through the device from the digitiser. All *control* instruments implement a *pushparams* method that takes a list of values and sends these to the instruments. This method is called when setting voltages to the DAC, for example. All instruments implement a *config* method allowing for instrument configuration. The digitiser can be configured to use a trigger signal using this method. An experiment might require several instruments. To avoid having to reconfigure all instruments every time *Pygor-server* is closed, the desired configuration of instruments is stored in a json file that is referenced every time *Pygor-server* is started. With all instrument drivers configured, they can be abstracted together into an *experiment* class. At runtime, *Pygor-server* instantiates all instrument classes according to the json files. All instruments are then passed to the constructor of the *experiment* class returning an instance. Methods of the experiment instance are then registered to a simpleXMLRPC server.

The low level control performed by *Pygor-client* involves setting specific variables, performing simple grid scans, plotting measurements, and saving data. *Pygor-client* uses an ip address and port to connect to *Pygor-server*. Once this is done, communication with the exposed methods of the experiment object becomes possible. *Pygor-client* implements the key methods: *getval*, gets value of a parameter; *setval*,

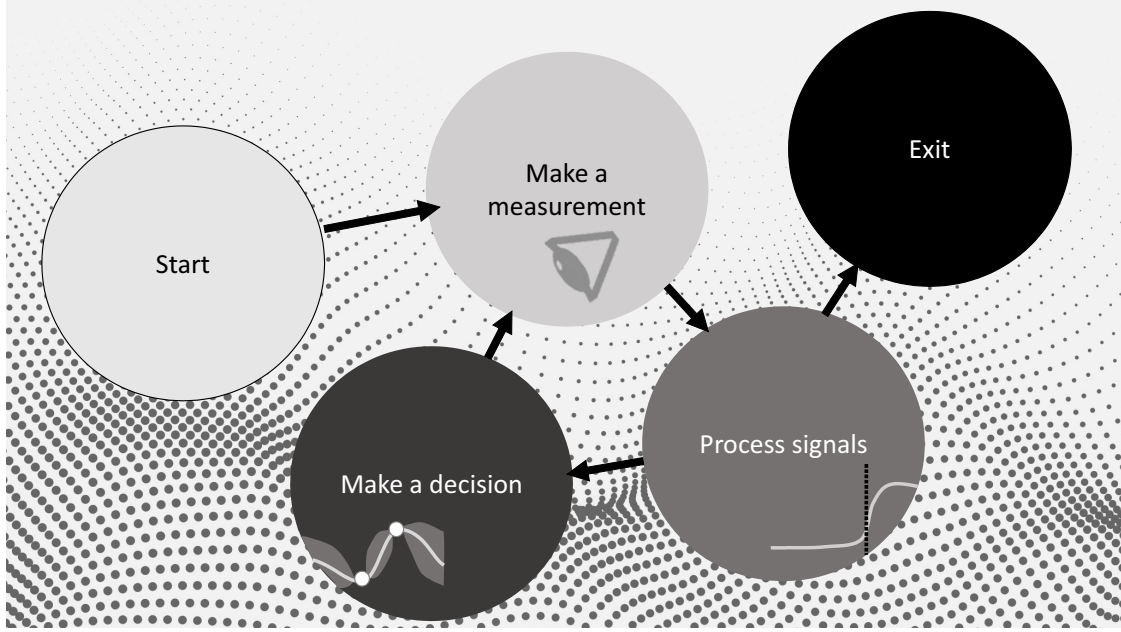


Figure 2.5: Algorithms workflow. A diagram outlining the workflow of automation algorithms in this thesis.

sets value of a parameter; *getvals*, gets the value of many parameters; *setvals* sets the value of many parameters; *get_params*, gets the value of all parameters; *set_params*, sets the value of all parameters; *do0d*, measures all active acquisition instruments; *do1d*, performs 1D grid scan for a given parameters over a given range; and *do2d*, performs a grid scan in a raster pattern for given parameters over given ranges. Other more specific methods are also implemented but these are the most notable.

In this thesis *Pygor* is used so that the lab PC runs *Pygor-server* while an onsite GPU server runs *Pygor-client*. The GPU server uses *Pygor-client* to control the experiment. In some cases the client is bypassed allowing for finer control of the experiment. The benefit of this setup is that powerful computational hardware can be used for automation during an experiment without requiring a server next to the dilution fridge. It also allows cloud computing services to control experiments.

2.4 Algorithms

The final element that is most important to this thesis are the algorithms that are used to automate the control of the device. The algorithms employed vary

significantly in their goals, methodology, and implementation, but they all directly control the experiment without human intervention. Algorithms are typically iterative, iterating between the collection of measurements, the signal processing of these measurements, and the decision about what to measure next (Fig. 2.5). This process is normally initiated by a human operator, and terminated after some condition is met; typically a fixed number of iterations is reached or a stopping criterion is satisfied.

Making a measurement typically involves acquiring data sets of measurements like that of the grid scan (section 1.6).

Processing signals is the first way in which machine learning techniques can prove to be invaluable. By signal I refer to any arbitrarily structured set of measurements. Often, information is contained in the patterns and structures observed in measurements. Methods that can be leveraged here fall into the camp of classical computer vision or deep-learning techniques (section 1.8.2). Computer vision techniques are challenging to apply successfully, they require some degree of hand crafting in order to work. Their benefit is that they can be developed with the knowledge of relevant measurement features in mind. This means no training data is required and thus there is a lower risk of over-fitting on some spurious insignificant detail in training data sets. Deep-learning (section 1.9.1), conversely, allows little user knowledge to be imposed, as methods are completely defined by their data-sets. This approach can only be as good as the data-sets it is trained on; a data-set of cat images is of no use for scoring bias triangles.

Finally, the aspect that differentiates the content of this thesis from prior literature is the use of machine learning algorithms for making decisions in quantum device experiments. Prior work uses simple numerical optimisers and heuristic decision rules. In this work I make heavy use of statistical methodology employing techniques such as active learning (Chapter 3), and Thompson sampling (Chapter 4). The common element in these methods is that they all build a model to represent beliefs about what the experiment will show. These models are then exploited to generate new decisions either by optimising an acquisition function or by sampling

from a probability distribution. This approach fundamentally differs from all prior work that does not attempt to predict device behavior.

3

Efficient measurement of quantum devices

Grid scanning (section 1.8.1) is used throughout device characterisation. Some examples of where they see use are: the characterisation of tunnel barriers; the characterisation of charge states; bias spectroscopy; spin funnel measurements; Rabi experiments; and Ramsey experiments. Despite being key to device characterisation, they can take tens of minuets or more to perform.

The information contained in a scan is not uniform, flat regions of the scan provide little information. In this chapter, I introduce a method for optimising the order in which measurements are acquired in a grid scan in order to increase the rate of information gain about the device.

3.1 Device and experiment

To demonstrate the algorithm a GaAs/AlGaAs heterostructure device was used (chapter 1.6). The device was fabricated by patterning five Ti/Au gate electrodes over the heterostructure (Fig. 3.1a). Gate voltages V_1 , V_2 , V_3 , V_4 and V_G are applied to create the confinement potential. Gate voltages V_1 to V_4 tune the tunneling rates while V_G mainly shifts the electrical potential of the quantum dot. The gate architecture is designed to form a single quantum dot. Two ohmic contacts are

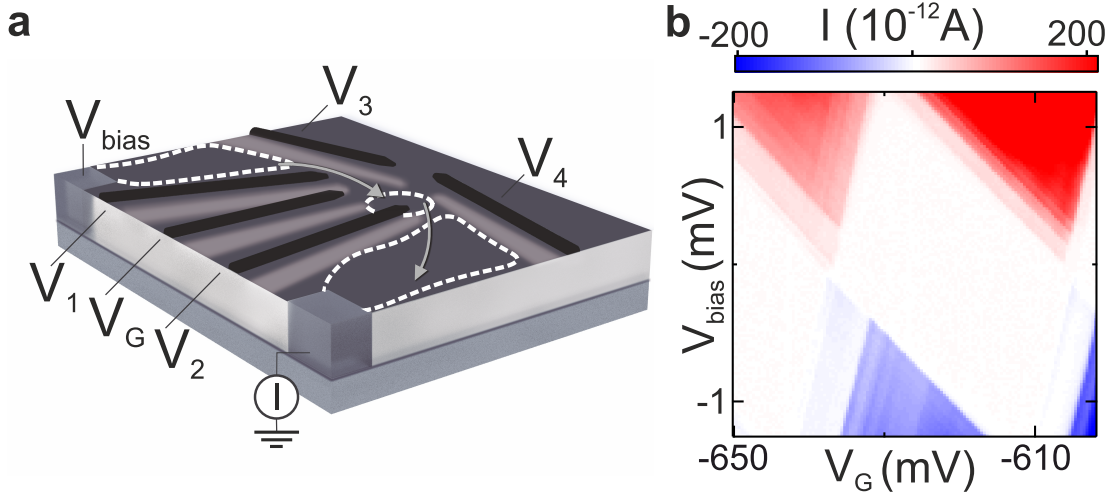


Figure 3.1: Device architecture and example measurement **a** Schematic of the device. A bias voltage V_{bias} is applied between ohmic contacts to the two-dimensional electron gas. I apply gate voltages labelled V_1 to V_4 and V_G . **b** A measured current map showing the current flowing through the device I as a function of V_{bias} and V_G . The Coulomb diamonds are the white regions where electron transport is suppressed, and most of the information necessary to characterise a device is contained just outside these diamonds.

made to the 2DEG and a bias voltage V_{bias} is applied. The current through the device I is measured. Measurements were performed at 30 mK.

I first consider bias spectroscopy measurements for which the quantum dot is characterised by varying V_G and V_{bias} for fixed values of V_1 to V_4 . As discussed in section 1.5, I expect to observe Coulomb diamonds. Figure 3.1b shows an example of such measurement. Coulomb diamonds contain large regions in which the current is almost constant with respect to V_G and V_{bias} . Taking measurements in these regions slows down informative data acquisition. The algorithm I developed, would be expected to take measurements close to the diamond edges.

3.2 Algorithmic framework

I will introduce the algorithmic framework used in this chapter (Fig. 3.2). The algorithm attempts to acquire the most informative measurements. This is done through the use of a model and an acquisition function. Given some initial measurements the model is used to produce predictions about what the full current map could look like, these are referred to as reconstructions. Disagreement between

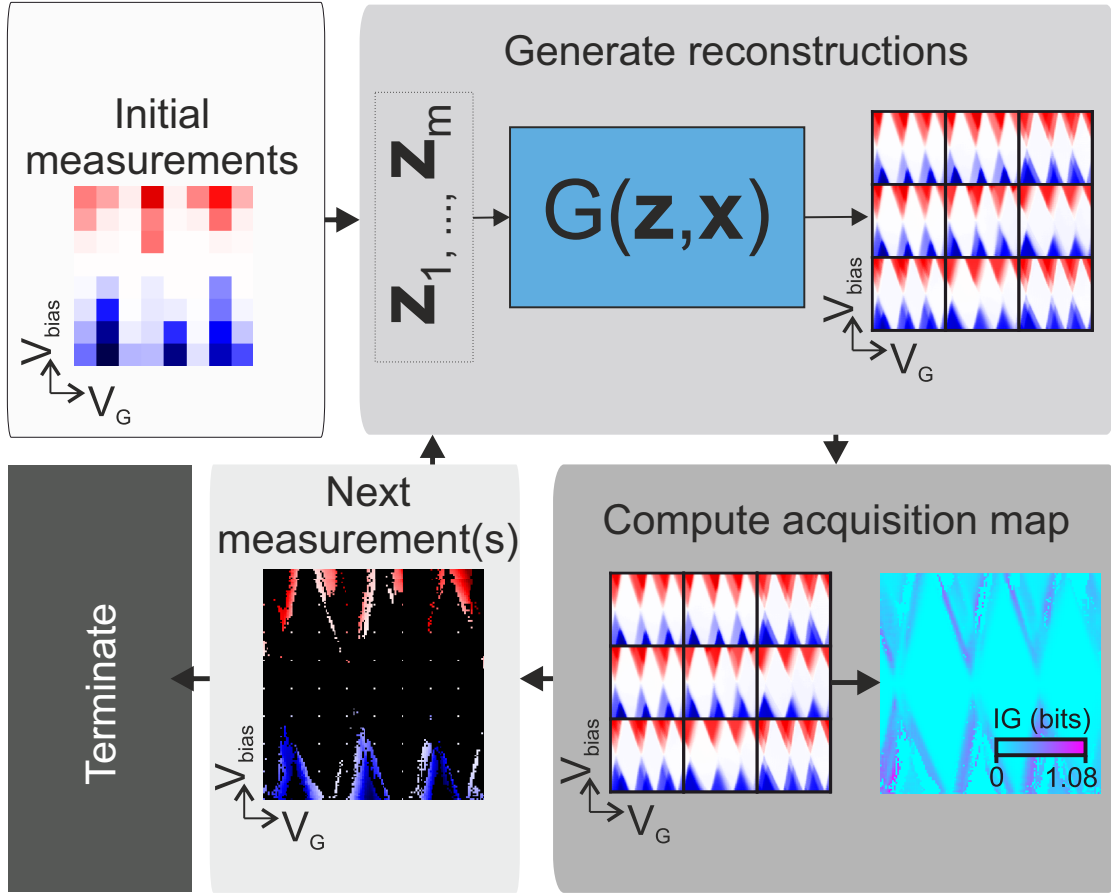


Figure 3.2: Algorithmic framework. A flow diagram of the algorithm framework is displayed. The algorithm adheres loosely to the general workflow detailed in section 2.4. Measurements are taken, signals are interpreted computationally as reconstructions, and decisions are made by processing the reconstructions.

reconstructions is used to produce the acquisition map. Using the acquisition map new measurements are performed and the process repeats.

A measurement consists of selecting a value $\mathbf{x}$ and acquiring a value y . When performing bias spectroscopy $\mathbf{x} = (V_G, V_{\text{bias}})$ and y is a measurement of current I . The algorithm iteratively collects measurements. At iteration n the algorithm collects measurement $(\mathbf{x}_n, y_n)$ and includes it in the set of all acquired measurements, which I will call a partial current map $Y_n = \{(\mathbf{x}_i, y_i)\}_{i=1, \dots, n}$. The full current map will be $Y = Y_N$ for brevity.

The model used by the algorithm is of the form $\hat{y} = G(\mathbf{z}, \mathbf{x})$. $\mathbf{z}$ is a vector of parameters I refer to as a latent space, and $\hat{y}$ is a prediction about the value of y at $\mathbf{x}$. The predicted partial current map is $\hat{Y}_n = \{(\mathbf{x}_i, \hat{y}_i)\}_{i=1, \dots, n}$, and a predicted

full current map, referred to as a reconstruction, $\hat{Y} = \hat{Y}_N$.

The algorithm starts by producing $m = 1, \dots, M$ values of $\mathbf{z}$, denoted $\mathbf{z}_m$ with are used to produce reconstructions $\hat{Y}_{1:M} = \{\hat{Y}_m\}_{m=1,\dots,M}$. An acquisition function is applied to $\hat{Y}_{1:M}$ as a function of $\mathbf{x}$, producing an acquisition map. New measurements $(y_n, \mathbf{x}_n)$ are selected according to this acquisition map. Restrictions are placed over possible values of $\mathbf{z}$ to take into account new measurements. This process is iterated.

3.3 Predicting unobserved measurements with a deep generative model

A model G is used to make predictions. The typical approach for this task would be to use a Gaussian process [63] or a physics model. I used a deep generative model instead because of their ability to produce accurate reconstructions. A deep generative model aims to produce new data that is indistinguishable from the set of training examples it was given. Deep generative models have been used for: speech synthesis [65]; generating images of digits and human faces [66, 67]; transferring image style [68, 69]; and inpainting missing regions of images [70]. Recently, variational auto-encoder (VAE) models have been used in scientific research to optimise molecular structures [71, 72]. In spite of their suitability, these models have not previously been applied to efficient data acquisition.

The specific deep generative model selected is a conditional variational auto-encoder (CVAE) [73] which is capable of generating high-resolution reconstructions given partial information and is fast enough for real-time decisions. Deep generative models such as generative adversarial networks (GAN) [74], the variational auto-encoder (VAE) [75] and its extensions, such as CVAE, have shown great success in multi-modal distributions and complex non-stationary patterns of data, similar to those of observed in quantum device measurements.

The CVAE characterises likely patterns in a training data set, which in this case was composed of a mixture of measured and simulated current maps (section 3.4). Deep generative models encode this pattern characterisation in a low-dimensional

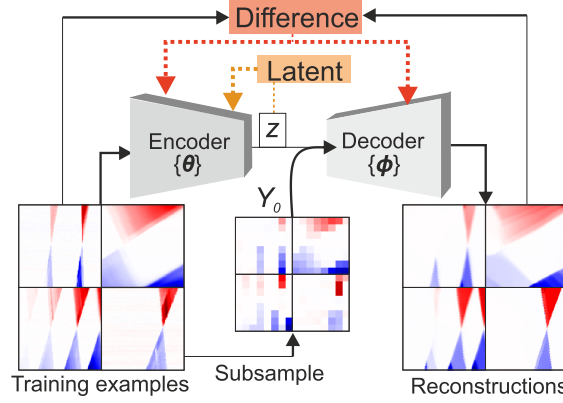


Figure 3.3: CVAE training A diagram of the training procedure of the CVAE. Two CNN's, encoder and decoder, are parameterised by θ and ϕ . The encoder and decoder are trained to reproduce the original input training examples with a difference loss. The encoder is trained to encode in a manner dictated by the latent loss.

real-valued vector, the latent vector $\mathbf{z}$. Each latent vector can be decoded to produce a full-resolution reconstruction of the data.

The latent space representation and the decoding are learned during training (Fig. 3.3). The CVAE consists of two convolutional neural networks (section 1.9.1), an encoder and a decoder. The encoder is trained to map full-resolution examples of current maps Y to the latent space representation $\mathbf{z}$. The encoder also enforces that the distribution $p(\mathbf{z})$ of training examples in latent space is Gaussian.

The decoder is trained to reconstruct Y , from the representation $\mathbf{z}$ combined with an 8×8 subsample of Y . As a result, $\mathbf{z}$ attempts to represent all the information that is missing from the subsampled data. In a plain VAE, the input of a decoder is only $\mathbf{z}$.

Training aims to minimise user-defined loss terms by adjusting the decoder and encoder parameters θ and ϕ using a gradient-descent based method. The two loss terms that are minimised to train the encoder and decoder networks are referred to as the difference loss and the latent loss. The difference loss consists of two metrics. The first is a sum of the pixel-wise difference between the reconstruction and the training example. The second is a contextual difference which is similar in concept to GAN; the contextual loss is taken from another convolutional neural network called a discriminator. The discriminator is trained in tandem with the encoder and decoder and is trained to distinguish between reconstructions and training examples. The

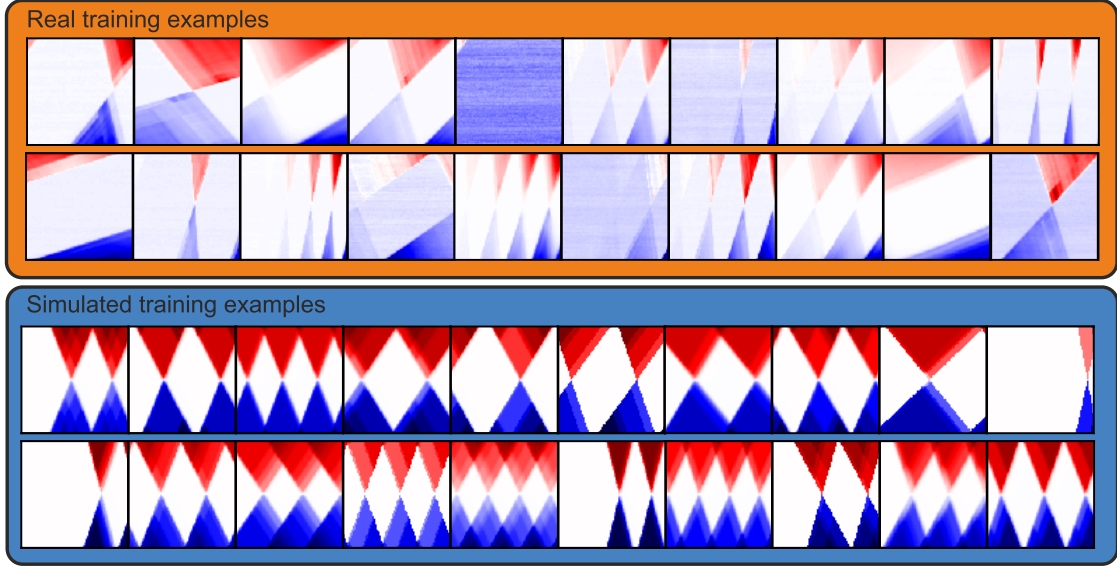


Figure 3.4: Examples of real and simulated training data. Samples from the real and simulated training data-sets.

input to the discriminator is a training example Y or reconstruction $\hat{Y}$ and the output is a continuous value between 0 and 1, representing the probability the input is a training example or a reconstruction. As the discriminator is trained to distinguish between training examples and reconstructions, it learns to decode contextual features that distinguish reconstructions from training examples. The difference between representations of the training example in different layers (layers as defined in section 1.7) of the discriminator and representations of its reconstruction (for the same layers) are used for the contextual loss. The contextual loss is required to avoid blurry, unrealistic, reconstructions.

The latent loss is applied only to the encoder and forces the set of encoded training examples to be normally distributed with mean $\mu = \mathbf{0}$ and covariance of $\Sigma = \mathbb{1}$. This can be achieved by minimising the Kullback-Leibler (KL) divergence between the output distribution of the encoder and the target zero-mean distribution.

3.4 Training data

The model is trained using both simulated and measured current maps. I choose to work with current maps of resolution 128×128 . The simulation is based on a

constant-interaction model using equations mentioned in section 1.5. To generate the simulated maps, random values of C_s , C_d , and C_g are chosen from a uniform distribution. Electron energy level transitions are generated randomly to provide the CVAE with a sufficiently diverse set of data. Finally C_s , C_d , and C_g are scaled by a randomly generated value as a function of N , producing current maps with diamonds that increase in charging energy as the occupation reduces. This characteristic is often seen in experimental data.

Measured current maps for training were generated by selecting gate voltages, bias ranges and gate voltage ranges randomly from a uniform distribution. The total training data set (Fig. 3.4) consists of 25,000 simulations and 25,000 real examples generated by randomly cropping 750 measured current maps. The current maps were subjected to data augmentation (random offsets, rescaling, and added noise) to increase the variability of the training set.

3.5 Leveraging the trained Deep Generative model

After training, only the decoder network is used for the algorithm's model G . The decoder outputs full reconstructions $\hat{Y}$. While measuring, the partial current map denoted as Y_n , where $n \leq N = 16,384$, is saved. To generate a reconstruction, the decoder takes as input the initial 8×8 grid scan Y_{64} , together with a latent vector $\mathbf{z}_m$ sampled from the posterior distribution $p(\mathbf{z}|Y_n)$.

Samples from the posterior distribution are generated using the prior density $p(\mathbf{z})$ and a likelihood function. I approximate the likelihood function using approximate Bayesian computation [76]. This is necessary as the CVAE only outputs a deterministic prediction $\hat{Y}$ per sample of $\mathbf{z}$. The approximation used is given by the following expression:

$$p(Y_n|\hat{Y}) \propto \exp(-\lambda \sum_{(\mathbf{x},y) \in Y_n} |y - \hat{Y}(\mathbf{x})|), \quad (3.1)$$

which allows for a likelihood over $\mathbf{z}$

$$p(\mathbf{z}|Y_n) \propto p(Y_n|\hat{Y} = \hat{Y}_z)p(\mathbf{z}). \quad (3.2)$$

The latter equation can be derived by considering that $Y_n \perp\!\!\!\perp \mathbf{z}$.

Multiple posterior samples are drawn from $p(\mathbf{z}|Y_n)$ by the Metropolis-Hastings (MH) method to approximate $p(\mathbf{z}|Y_n)$. These samples are represented by $\mathbf{z}_m$, with corresponding reconstructions $\hat{Y}_m$. In this work $m = 1, \dots, M$ with $M = 100$. The continuous posterior $p(\mathbf{z}|Y_n)$ is approximated by a discrete posterior of samples $P_n(m)$, which denotes how probable $\hat{Y}_m$ is. This approximation is necessary as $p(\mathbf{z}|Y_n)$ has no closed form and sampling it with MH is computationally expensive. I refer to $P_n(m)$ as the posterior distribution of reconstructions.

3.6 Choosing optimally what to measure next

With each iteration of the decision algorithm, an acquisition map is computed using samples from the posterior distribution of the model. The acquisition map assigns to each potential measurement location (i.e. to each pixel location in the current map) an information value (Fig. 3.5). The criterion used is

$$IG(\mathbf{x}) = \mathbb{E}[KL(p_n(\mathbf{z}|(y_{n+1}, \mathbf{x}_{n+1}))||p_n(\mathbf{z}))], \quad (3.3)$$

which is the expected information about $\mathbf{z}$ to be gained by performing a measurement of y_{n+1} . KL is the Kullback-Leibler divergence and $p_n(\cdot) = p(\cdot|Y_n)$. This criterion is equivalent both to Bayesian active learning by disagreement [77] which can be seen by considering

$$\mathbb{E}[KL(p_n(\mathbf{z}|(y_{n+1}, \mathbf{x}_{n+1}))||p_n(\mathbf{z}))] = I(\mathbf{z}; (y_{n+1}, \mathbf{x}_{n+1})|Y_n) \quad (3.4)$$

From this I see that $IG(\mathbf{x})$ is equivalent to conditional mutual information of a pair of observations and the latent space conditional on observations that have already been made. Appendix .2 provides further details of how equation 3.3 is applied to the CVAE.

To make decisions based on the acquisition map, I follow two approaches; a pixel-wise method and a batch method. The pixel-wise method selects the single

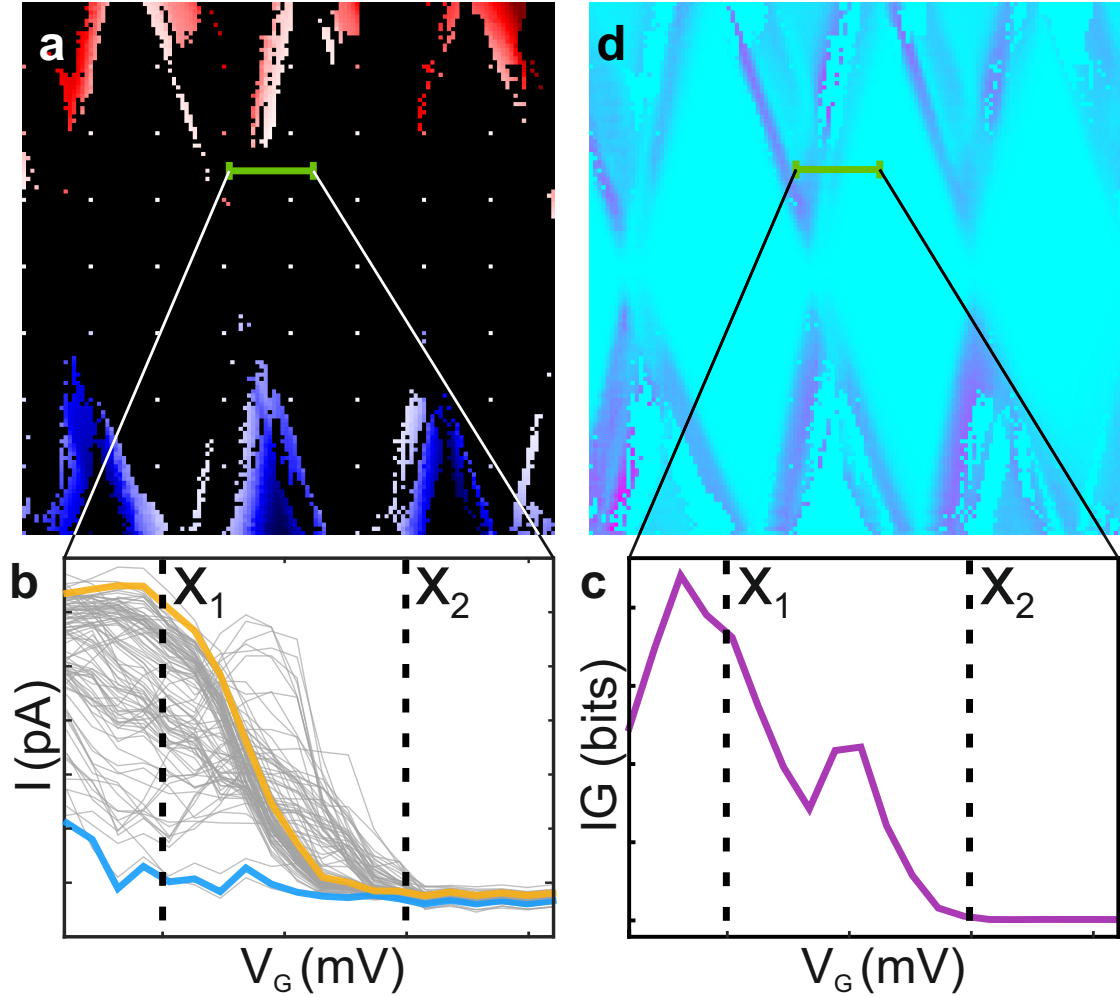


Figure 3.5: **a** Partial current map. To illustrate the first step in the computation of the acquisition map, a trace is considered (green) through an unmeasured region of the map. **b** For the unmeasured trace in **a**, reconstructions provide 100 different predictions. Blue and yellow traces highlight two of these predictions. The objective is to determine the most informative measurement location. At X_2 , all predictions are similar, so measuring here will have little impact on the posterior distribution of reconstructions. At X_1 , predictions are dissimilar and, therefore, X_1 is a more informative measurement location, with a larger effect on the posterior distribution of reconstructions. **c** Information gain computed for the unmeasured trace in **(a)**. **d** Acquisition map of information gain computed from the partial measurements in **(a)**, and plotted over the entire current map range.

best pixel in the acquisition map. In practice, this is often not optimal in terms of measurement time, because it does not take into account the time cost of sweeping gate voltage to move between measurement locations or the computational cost of recalculating the acquisition map. To overcome this limitation, I also devised a batch method, which selects multiple locations from the acquisition map, and then acquires measurements by taking a fast route between them. This approach reduces the measurement time compared with the pixel-wise method.

3.7 Efficient measurement of single quantum dot devices

To test the algorithm, I used it to acquire a series of current maps in real time. The device was thermally cycled to randomise the charge distribution and therefore present the algorithm with a configuration not represented in its training data. Gate voltages V_1 to V_4 were set to a combination of values, and the algorithm was tasked to measure the corresponding current map (I as a function of V_{bias} and V_G) using both the batch and the pixel-wise methods. Ten different combinations of bias and gate voltages were considered. Fig. 3.6 presents data acquired by the algorithm at selected acquisition stages, together with selected reconstructions. As expected, reconstructions become less diverse as more measurements are acquired. The reconstructions do not necessarily replicate the measured current map for large n . This is because reconstructions present a variability given by the training data set. Decisions are based on learned patterns from the training data and thus all possible features in a current map might not be considered, but common patterns most certainly are. As it can be observed in Fig. 3.6, the algorithm gives high priority to regions of the current map where the current is rapidly varying and avoids regions of nearly constant current, such as the interiors of Coulomb diamonds. This strategy is an emergent property of the algorithm and is wise; little information about the device characteristics can be found in low-current gradient regions of the current map. This preference derives from the comparison between reconstructions, which exhibits the greatest disagreement outside Coulomb diamonds.

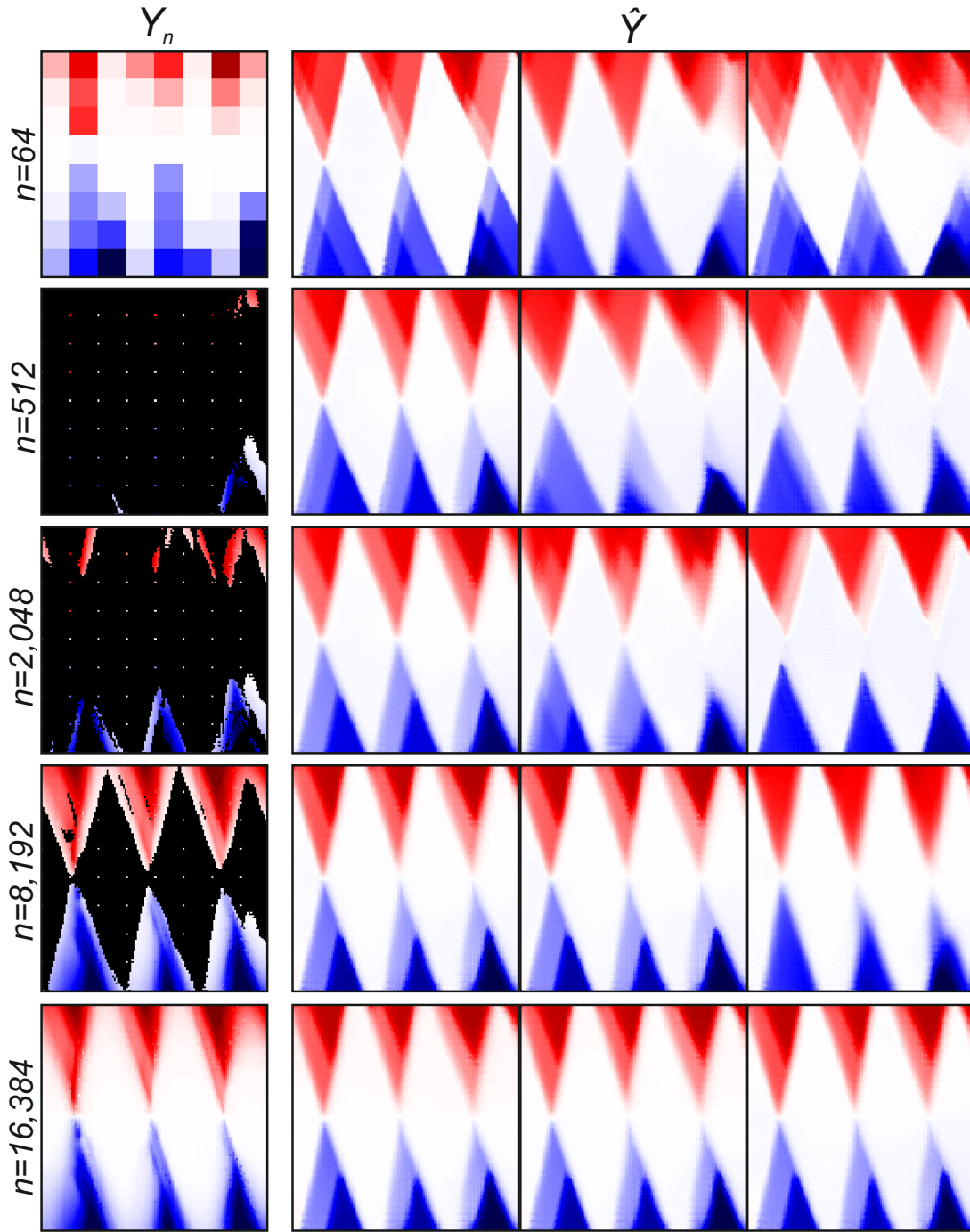


Figure 3.6: Updating reconstructions using information from new measurements In each row, the first column shows the algorithm-assisted measurements, using the batch method, for a given n . The remaining three columns contain example reconstructions given the corresponding n measurements. As n increases, the diversity of the reconstructions is reduced and their accuracy increased. As expected, the uncertainty is almost eliminated in the last row. The residual remaining variance is because slightly different reconstructions are nearly equally consistent with the data.

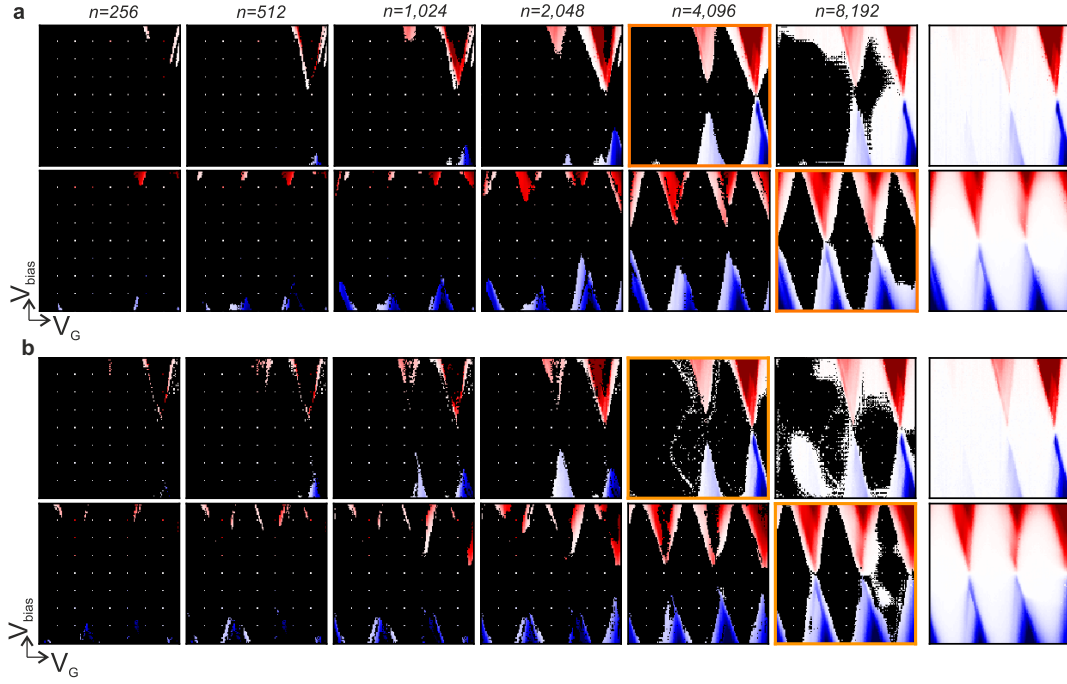


Figure 3.7: Measurements of Coulomb diamonds performed by the algorithm a and b Sequential measurements for two different current maps. Each row displays algorithm-assisted measurements of I as a function of V_{bias} and V_G for different values of n . The last plot in each row is the full-resolution current map. Panel **a** corresponds to results for the batch method. Panel **b** corresponds to results for the pixelwise method.

In Fig. 3.7, two representative measurement sequences using the batch method and the pixel-wise method are displayed. I observed that the batch method collects grouped measurements while the pixel-wise method distributes measurements more uniformly, given that in the latter case the acquisition map is more frequently updated to take into account recently acquired information. To advance the batch method, the information gain of the batch could be considered jointly [78].

The performance of the algorithm is bench-marked against an implementation of the alternating grid scan method. This type of grid scan starts with 8×8 measurements and alternately increases the vertical and the horizontal grid size by 2 (i.e. 16×8 , 16×16 , 32×16 , etc.), without performing the same measurement twice. Over the ten different current maps, the average time to acquire a full-resolution current map with the alternating grid scan method is 554 seconds. This time is limited by the bias and gate voltage ramp rates and by the settling time chosen. The batch method can be implemented with any batch size. For direct comparison

with the alternating grid scan, increasing batches of 32×2^b are selected, where b is the batch number starting from 1.

Two types of computation are required to make a measurement decision: the sampling reconstructions using the Metropolis Hastings (MH) method, and the construction of the acquisition map. One MH sampling iteration takes 63 ms. Multiple of these sampling iterations are performed after each batch decision and measurement. Since sampling can be performed simultaneously with measurement acquisition, in this chapter I will refer to measurement times excluding the time required for sampling. To compute a single acquisition map takes approximately 50 ms using a NVIDIA GTX 1080 Ti graphics card and a Tensorflow [79] implementation. The acquisition map must be computed for every batch or every pixel measurement, except for the initial 8×8 grid scan and the final acquisition step (for which there is no choice of which pixel(s) to measure next). To acquire a full resolution current map thus requires 7 computations (350 ms) for the batch method, and 16,319 computations (816 s) for the pixel-wise method. For the batch method, the computation time is negligible compared to the measurement time, but for the pixel-wise method it is a limiting factor in the measurement rate.

To quantify the algorithm's performance, a new measure is introduced based on the observation that the most informative regions of the current map are those where the current varies strongly with V_G and V_{bias} . I define the local gradient of the current map at each location $\mathbf{x}$ as

$$v(\mathbf{x}) \equiv \|\nabla Y(\mathbf{x})\|_2 = \sqrt{\left(\frac{\partial I(\mathbf{x})}{\partial V_G}\right)^2 + \left(\frac{\partial I(\mathbf{x})}{\partial V_{\text{bias}}}\right)^2}, \quad (3.5)$$

where $I(\mathbf{x})$ is current measurement at $\mathbf{x}$, and the derivatives are calculated numerically. The error measure $r(n)$ of a partial current map is the fraction of the total gradient that remains uncaptured, i.e.

$$r(n) \equiv 1 - \frac{V(n)}{V(N)} \quad (3.6)$$

where $V(n) \equiv \sum_{i=1}^n v(\mathbf{x}_i)$ is the total acquired gradient and $\mathbf{x}_i$ is the location of the i th measurement.

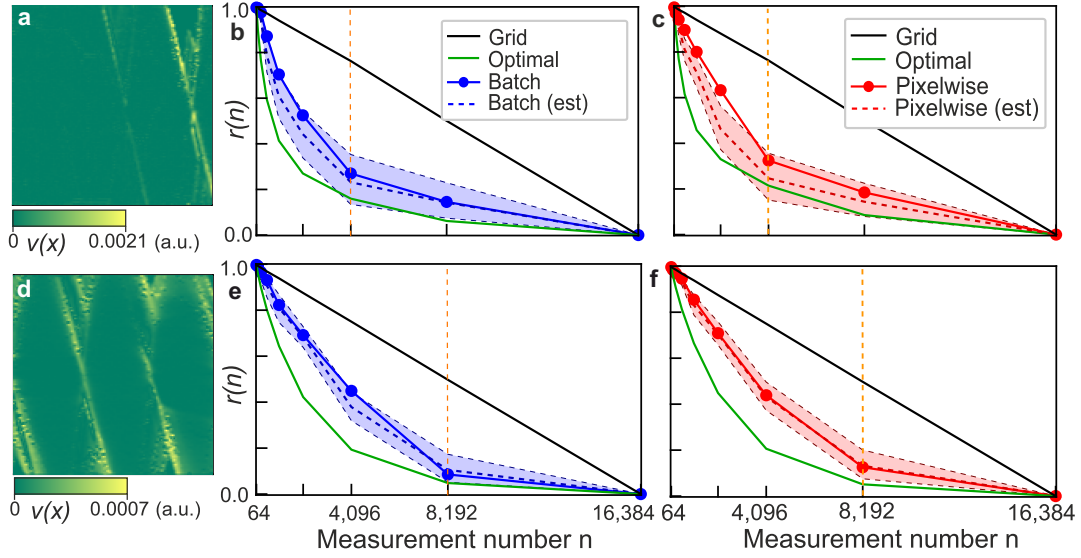


Figure 3.8: Algorithm's performance. **a, d** Current gradient map (defined by Eq. 3.5) for each current map in Fig. 3.7. **b, e** (**c, f**) Measure of the algorithm's performance $r(n)$ for the batch (pixelwise) method. Real-time estimate of $r(n)$ across reconstructions with credible 90% interval shaded and optimal $r(n)$ for both current maps in Fig. 3.7a (Fig. 3.7b). The black line is the value of $r(n)$ corresponding to the alternating grid scan method. The vertical orange line indicates the value of n determined by the stopping criterion. The corresponding current map is highlighted in orange in Fig. 3.7a (Fig. 3.7b).

This error can only be calculated after all measurements have been performed. However, I can utilise the m th reconstruction to generate an estimate $\tilde{r}_m(n)$ in real time by replacing $\|\nabla Y(\mathbf{x})\|_2$ with $\|\nabla \hat{Y}_m(\mathbf{x})\|_2$. The estimates from multiple reconstructions yield a credibility interval for $r(n)$. For the real time estimates of $r(n)$, the number of reconstructions is augmented up to 3,000 to include different noise patterns typically present in current maps, this is detailed in appendix .3. For an optimal algorithm, the error would be $\bar{r}(n) = 1.0 - \frac{V^*(n)}{V(N)}$, where $V^*(n)$ is the sum of the largest n values of $v(\mathbf{x})$. This would be achieved if each measurement location were chosen knowing the full-resolution current map, and thus the location of the the highest unmeasured current gradient. No decision method can exceed this bound.

Algorithm performances for two representative current maps are shown in Fig. 3.8. Grid scans reduce $r(n)$ linearly with increasing n . Both the pixel-wise and batch decision algorithms outperform the simple grid scan and are nearly optimal. When most of the current gradient is localised, the grid scan is far from optimal and even

the decision algorithm has room for improvement. In this case, the performance of the algorithm is determined by how representative the training data is.

When should the algorithm stop measuring a given current map? To answer this question, I considered a simple stopping criterion that uses the estimated reduction of the error $r(n)$ to determine when to stop measuring in a scenario in which more experiments are waiting to be conducted. For a given current map containing n measured pixels, the error after the next measurement batch is estimated for reconstruction m to be $\tilde{r}_m(n + \Delta)$, where Δ is the size of the batch. Thus the estimated rate at which the error decreases is $\beta_m \equiv |\tilde{r}_m(n + \Delta) - \tilde{r}_m(n)|/\Delta$. In the worst case among the candidate reconstructions, this rate is $\beta \equiv \min_m \beta_m$. If the algorithm begins to measure a new current map, for which no reconstructions are yet available, the error will decrease at a rate of at least $\alpha \equiv 1/N$; this is the slope achieved by a grid scan and the worst case of the decision algorithm (black lines in Fig. 3.8b, c, e, and f). Hence whenever $\beta < \alpha$, it is beneficial to halt measurement and move onto a new current map that is awaiting measurement. Since α and β correspond to the worst-case estimates for each measurement method, the criterion is conservative in its decision to stop. The stopping points corresponding to this criterion are shown in Fig. 3.8b, c, e, and f, with orange dashed lines. The total average time (measurement time plus decision time) to reach the stopping criterion was 237 s, compared with the 554 s required to measure the complete current map by grid scan. In this way, I reduce the measurement time by a factor between 1.84 and 3.70 across all 10 test cases.

3.7.1 Generalising the algorithm

The algorithm described here does not require assumptions about the physics of the experiment. Provided that training data is available, the algorithm is thus applicable to other kinds of measurements. To confirm this, the algorithm was applied to a different measurement configuration also encountered in quantum dot tuning. In this case, the current flowing through the device is measured as a function of two gate voltages (V_1 and V_2), while keeping other voltages fixed (V_G ,

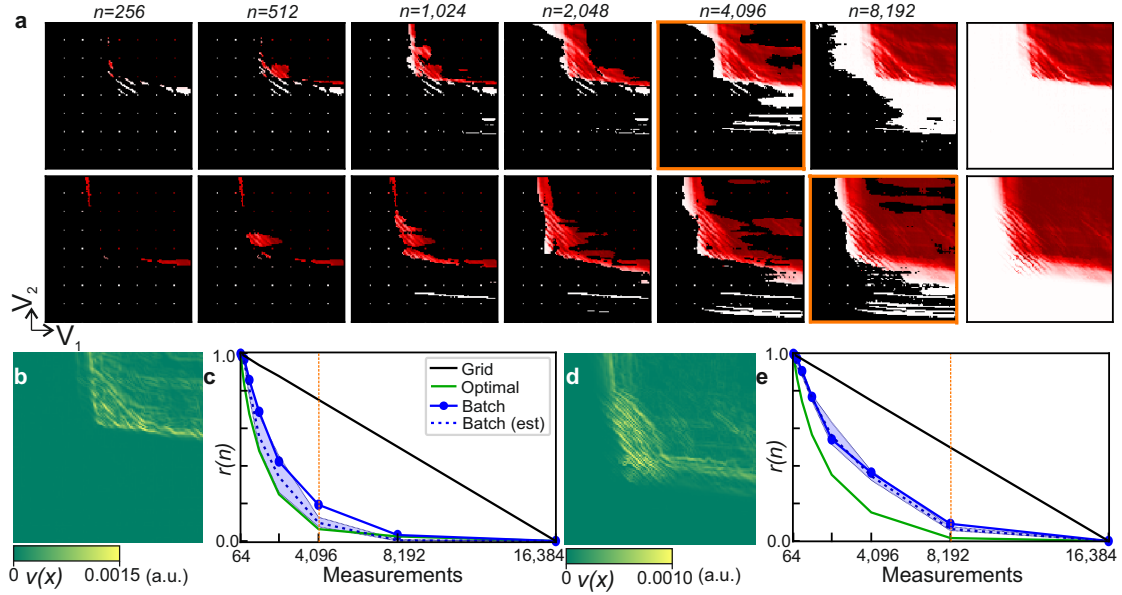


Figure 3.9: Measuring a different type of current map. **a** Each row displays the algorithm-assisted measurements of a current map as a function of V_1 and V_2 for different values of n . Sequential batch measurements were performed. The last plot in each row is the full-resolution current map. **b**, **d** Current gradient map for the current maps in the top and low rows in **(a)**, respectively. **c**, **e** Measure of the algorithm's performance $r(n)$, average real-time estimate of $r(n)$ with 90% credible interval, and optimal $r(n)$ for the current maps in the top and low rows in **(a)**, respectively. The black line is the value of $r(n)$ corresponding to the alternating grid scan method. The dashed orange line indicates the value of n determined by the stopping criterion. The corresponding current maps in **a** are highlighted in orange. The alternating grid scans took 2267 s and 2333 s to acquire the full current maps in the top and low rows in **(a)**, respectively. The batch method took 673 s and 1552 s respectively to reach the stopping criterion.

V_{bias} , V_3 and V_4). These current maps evidence the transistor-like behavior of the device, i.e. a boundary separates large high and low current areas in which the current scarcely changes. At specific gate voltage locations close to this boundary, Coulomb peaks are present. For the training set, 382 current maps were measured with a resolution of 251×251 , as in section 3.4 gate voltage values and ranges were randomly sampled from a uniform distribution to produce each current map. These current maps were then randomly cropped to a resolution of 128×128 and subjected to the same image augmentation techniques.

The performance of the algorithm in this new scenario was tested by considering two different combinations of V_G , V_{bias} , V_3 , V_4 , and measuring the corresponding current maps in real time (Fig. 3.9). The device was thermally cycled after the

training set was acquired and in between the acquisition of the two current maps in Fig. 3.9. As expected, the algorithm focuses on measuring regions of high current gradient, the boundary between high and low current areas and, in particular, the Coulomb peaks close to it.

In the top row of Fig. 3.9a, $n = 4,096$ was the end condition set by the stopping criterion. In the bottom row, the boundary extends further in the current map and the stopping criterion indicated a stop point corresponding to $n = 8,192$. The algorithm reduced the time needed to measure the current maps by 3.36 and 1.50 respectively when compared with the alternating grid scans.

Memorisation is a known issue in deep generative models [80]. What this means in this context is that the CVAE used for reconstructions simply memorises current maps in the training data-set instead of generating new unique current maps. In this application the distinction is not as important we do not need to generate unique samples we are focused on the performance of decisions. The primary reason the CVAE is effective in this problem is it allows the user to create a function G that is fast essentially acting as a surrogate physics model. However, future work should aim to produce generative models that are capable of generating more complex current maps or capable of working on many different types of current map without retraining. In these instances more care should be taken to verify that models are capable of generalising.

3.8 Altering acquisition with segmentation for context aware decisions

By converting reconstructions to maps that convey context about them, decisions can be made relying on such context instead of on the raw measurements. To demonstrate this, a segmentation method that produces a segmentation map of Coulomb diamonds was developed. A segmentation map is a classification method that takes as an input a complete current map Y . From this current map a new map S_N , which contains N pixels, is produced. Each pixel takes a continuous value in the domain $[0, 1]$ and represents the probability that a pixel belongs to a given class. In

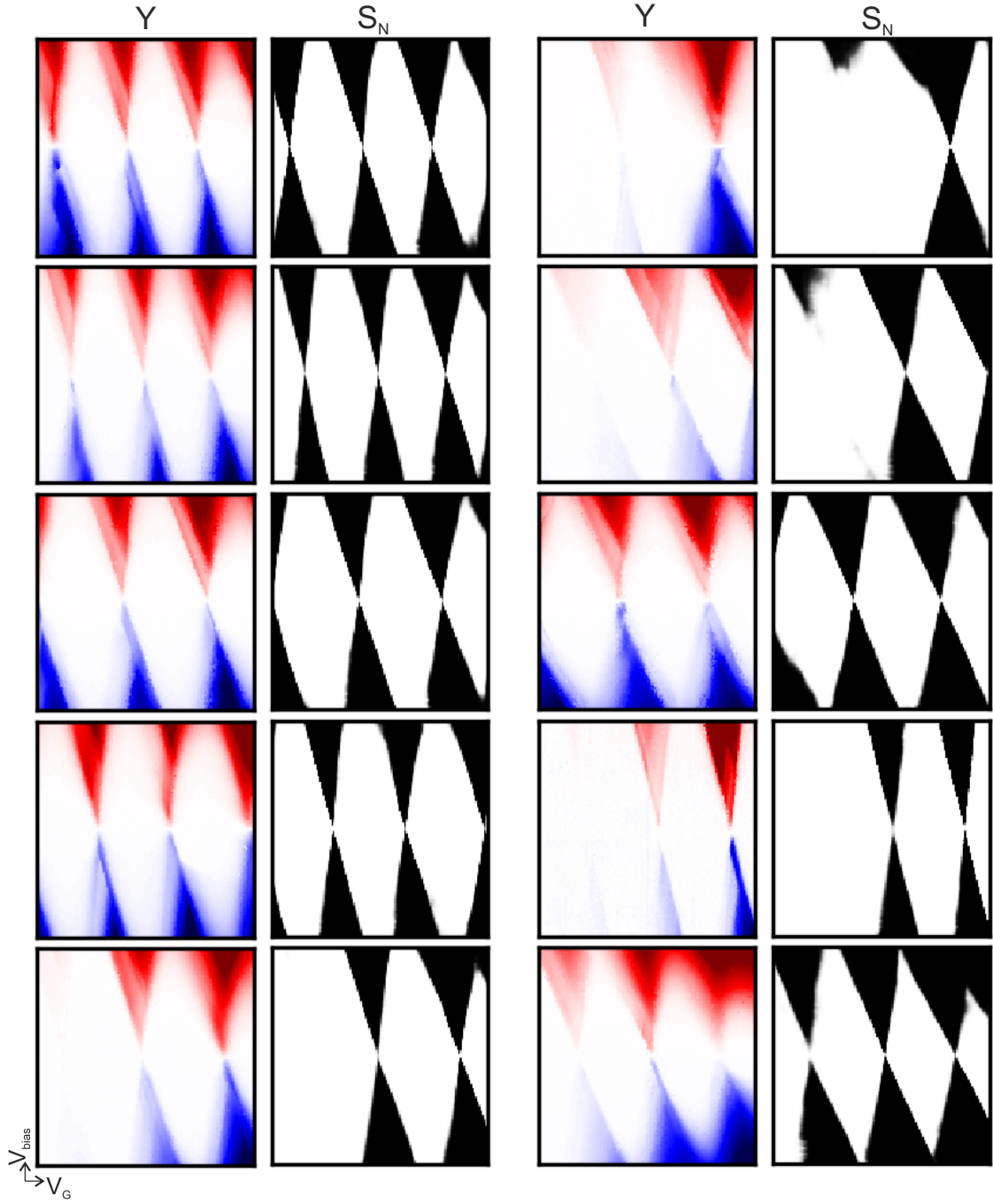


Figure 3.10: Segmentation results. Full resolution current maps and segmented results from the segmentation network.

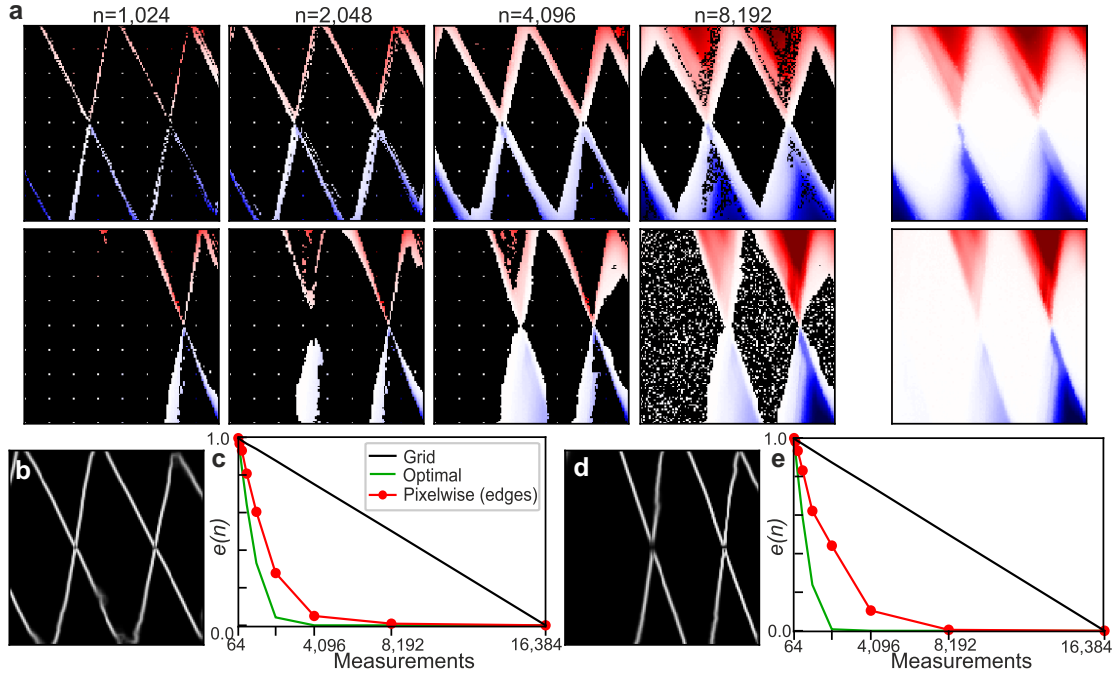


Figure 3.11: Measurements of current maps performed by the context-aware implementation of the algorithm **a** Each row displays the algorithm assisted measurements of a current map as a function of V_{bias} and V_G for different values of n . Sequential pixel-wise measurements were performed. The last plot in each row is the full-resolution current map. **b**, **d**, Sobel filter applied to the segmentation map of the full current map for both examples in **a**. **c**, **e**, Measure of the algorithm's performance $e(n)$ and optimal $e(n)$ for current maps in **a**, respectively. The black line is the value of $e(n)$ corresponding to the alternating grid scan method.

this case, the class is 1 for a pixel outside and 0 for a pixel inside a Coulomb diamond. This segmentation method is based on a deep neural network called a U-net [81]. Training data for the segmentation network are simulated current maps produced by the simulator in section 3.4, and segmentation maps also produced by the simulator. Figure 3.10 shows the segmentation result for 10 measured current maps.

By producing segmentation maps of reconstructions, their segmentation disagreement can be calculated. Because pixels can only be segmented by the U-Net when a full current map exists (i.e. the U-Net can only be applied to Y or $\hat{Y}$) equation 3.1 can not be used to approximate the likelihood and the information gain criterion can not be applied (equation 3.3). Instead I use a different criterion

$$IS(\mathbf{x}) = \text{Var}_m(S_{N,m}(\mathbf{x})), \quad (3.7)$$

the variance of all segmentation map values for a given value of $\mathbf{x}$. Although this criterion is more heuristic it produces large disagreement along the edges of reconstructions resulting in measurements that focus on diamond edges as shown in Fig. 3.11a. Perturbations are added to the outside of diamond segmentation maps to increase disagreement outside the diamonds. This leading to the method first prioritising edges, and then measurements outside the Coulomb diamonds once edges are measured. Reconstructions are still periodically updated considering current values Y_n using equation 3.2.

In order to calculate the success measure $e(n)$ in Fig. 3.11 c and e, I first applied the segmentation model to the fully measured current maps. I then applied a Sobel filter [82] to the resulting segmented maps. The Sobel filter is a popular computer vision edge detection method. In this way, edge maps are obtained. The errors and optimal performances are then calculated in the same way as equation 3.6 except with the Sobel filters output in place of $V(n)$ and $V(N)$.

3.9 Physics aware active learning

The model G does not necessarily have to be device agnostic. Instead, a physics model can be used to inform decisions. In this case, a latent space can be constructed over physical parameters in the model. I use this approach to efficiently measure Rabi oscillations. Data was obtained from a device similar to that explored in Ref. [15].

3.9.1 Rabi oscillations

Rabi oscillations occur when a qubit is exposed to a resonant external drive. In this work, the qubit is encoded in a single spin, as discussed in section 1.7.1. Qubit operation is achieved by EDSR. When an oscillating signal at the qubit's Larmor frequency $f = f_{\text{Larmor}}$ is applied to a gate electrode, the qubit oscillates between states $|\uparrow\rangle$ and $|\downarrow\rangle$ at a rate f_{Rabi} known as the Rabi frequency (fig 3.12a). These oscillations take place as long as the drive is applied and thus a longer drive duration t_{burst} produces a larger rotation. If the qubit is driven off resonance such that $|f - f_{\text{Larmor}}| > 0$, then oscillation occur at the generalised Rabi frequency

$\sqrt{f_{\text{Rabi}}^2 + (f_{\text{Larmor}} - f)^2}$. The amplitude of these oscillations is smaller, since the state $|\downarrow\rangle$ is never reached (fig 3.12b). A common grid scan performed while characterising a qubit is a Rabi chevron, an example of which is shown in fig 3.12c. This grid scan consists of current measured as a function of t_{burst} and f . Current is measured directly through spin-to-charge conversion over many applications of the same pulse cycle in a way identical to [15].

3.9.2 Predicting unobserved measurements with a physical model

The model selected consists of 6 unknowns, 2 which will form the space of $\mathbf{x} = (t_{\text{burst}}, f)$ and 4 which will form the latent variable $\mathbf{z} = (f_{\text{Rabi}}, f_{\text{Larmor}}, C_a, C_b)$. The model is

$$\hat{y} = C_a + C_b \frac{f_{\text{Rabi}}}{f_{\Omega}} \sin(\pi f_{\Omega} t_{\text{burst}})^2, \quad f_{\Omega} = \sqrt{f_{\text{Rabi}}^2 + (f_{\text{Larmor}} - f)^2}. \quad (3.8)$$

Where C_a and C_b are an offset and scaling factor accounting for an unknown linear transformation in the readout. A predicted partial map can be constructed as before $\hat{Y}_n = \{\hat{y}(\mathbf{x}_i, \mathbf{z})\}_{i=1, \dots, n}$, with $\hat{Y} = \hat{Y}_N$ representing a full map, where $N = 2,601$.

This simple numerical model predicts a noiseless $\hat{Y}$ and is capable of predicting individual pixel values hence it can be used to directly construct the likelihood of individual measurements under the assumption they are subject to Gaussian noise,

$$y|\mathbf{z}, \mathbf{x} \sim N(\hat{y}(\mathbf{z}, \mathbf{x}), \sigma^2) \quad (3.9)$$

σ introduces an additional latent variable to $\mathbf{z}$. The likelihood of a set of measurements can be constructed by considering

$$p(Y_n|\mathbf{z}) = \prod_{i=1}^n N(\hat{y}(\mathbf{x}_i, \mathbf{z}), \sigma^2) \quad (3.10)$$

and likewise the likelihood of $\mathbf{z}$ can be expressed as

$$p(\mathbf{z}|Y_n) \propto p(Y_n|\mathbf{z})p(\mathbf{z}). \quad (3.11)$$

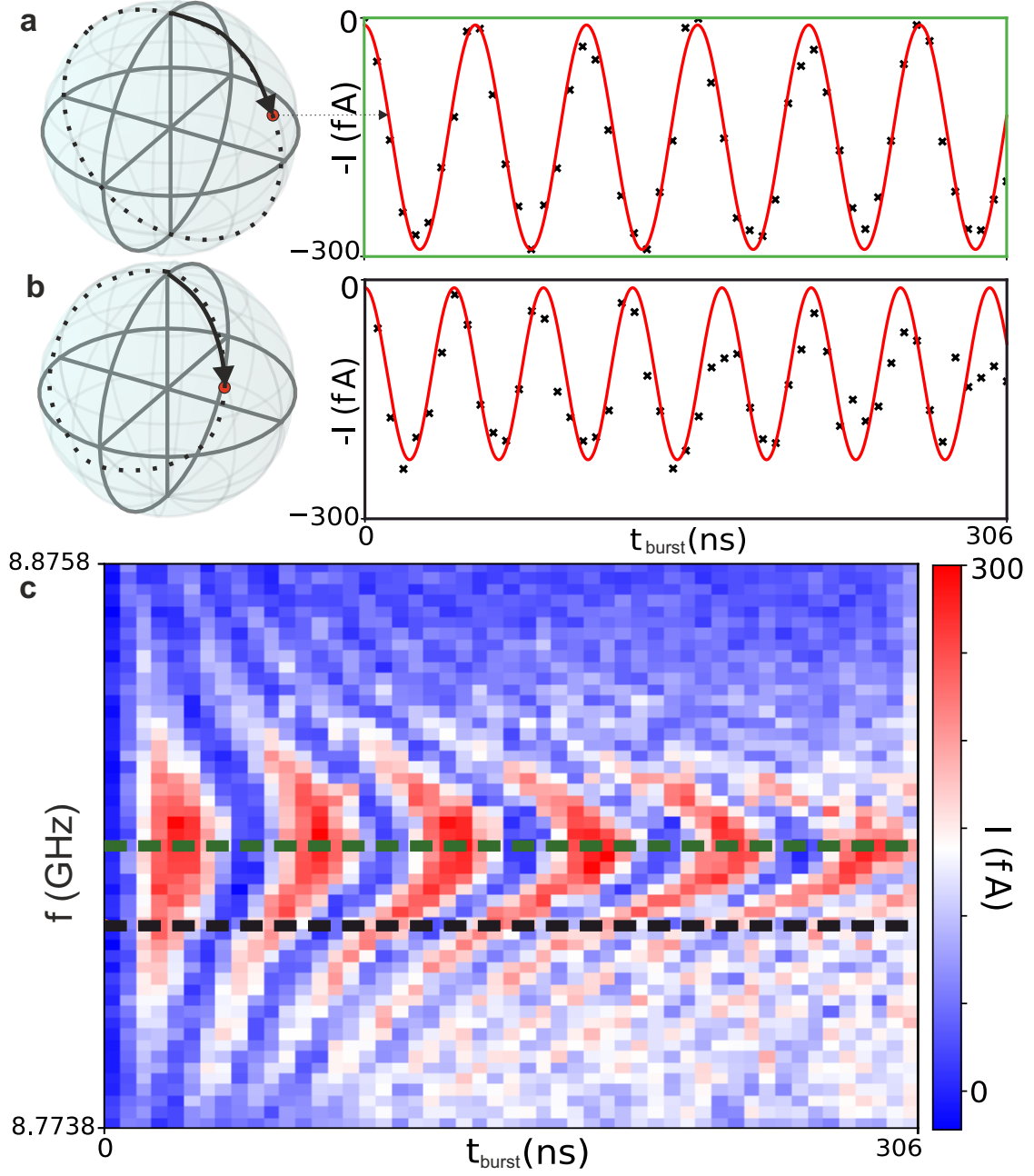


Figure 3.12: Rabi oscillations. **a** and **b** Rabi oscillations when driving at f_{Larmor} and $|f - f_{\text{Larmor}}| > 0$, respectively. Left: Schematic Bloch sphere shows the effect of driving. Right: Resulting current as a function of t_{burst} . **c** Rabi chevron pattern. Current is plotted as a function of f and t_{burst} . Slices of chevron corresponding to panels **a** and **b** are highlighted as colored dashed lines.

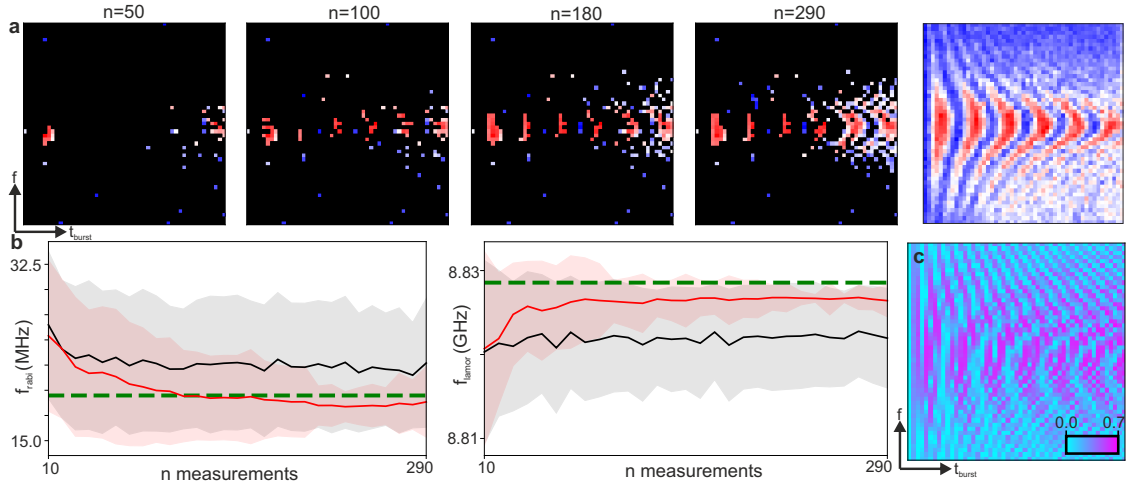


Figure 3.13: Performance of algorithm on a Rabi Chevron. **a** Sequential batch measurements for a single run of the algorithm. Row displays algorithm assisted measurements of current as function of f and t_{burst} for different values of n . The last plot in the row is the full-resolution current map. Note this experiment was performed offline, i.e. on saved data. **b, d** Algorithm's performance in inferring model parameters f_{Rabi} and f_{Larmor} . Inferred values of f_{Rabi} and f_{Larmor} as a function of the number of measurements n . The solid red line indicate the average performance of the algorithm, and the standard errors are indicated with a shaded red area. The average performance of random decisions is plotted for comparison (solid black line) and standard errors are indicated with a shaded grey area. The green dashed lines highlight the expected values for the inference of $p(\mathbf{z}|Y_n)$ estimated from the full-resolution current map. **c** Example information gain map for iteration $n = 100$ in panel **a**.

This expression is the analog of equation 3.2. Using this approach the posterior of $\mathbf{z}$ can be computed given a few measurements. The condition in equation 3.3 can now be applied to select where to measure next. Appendix .4 provides further details of how equation 3.3 is applied to the physics model.

3.10 Efficient measurement of Rabi oscillations

I apply the active learning method to the Rabi Chevron in Fig 3.12c using the probabilistic model detailed in section 3.9.2. To benchmark the performance of the algorithm, the convergence of model parameters f_{Rabi} and f_{Larmor} are considered. To provide a baseline, I compare the performance of the active learning method against selecting measurements at random. This experiment takes place offline. I apply both the algorithm and the random decision method to the current map displayed in Fig. 3.12 25 times and analyse their convergence. I utilise the batch

decision method with small batches of 10 measurements for 30 iterations. The reasoning for selecting only $n = 290$ observations is that I am interested in early convergence of the posterior distribution.

Figure 3.13a displays partial current maps from a single representative run. It is observed that measurements are concentrated on high current fringes in the Rabi Chevron pattern. This strategy is wise as it allows for efficient learning of f_{Larmor} and f_{Rabi} . Figure 3.13b confirms this showing that the active learning algorithm reduces its uncertainty quickly, i.e. for small values of n . Figure 3.13c shows the information gain map. Two sets of fringes can be identified in Fig. 3.13c, one with frequency $f_1 \approx f_{\text{Rabi}}$ and a second with $f_2 \approx 2f_1$. The most informative measurement locations correspond to those that distinguish between $f_{\text{Rabi}} = f_1$ and $f_{\text{Rabi}} = f_2$.

3.11 Discussion

I introduce an approach for performing measurements efficiently. The approach shows great promise for speeding up the measurement of slow grid scans which take tens of minutes or even hours. A model is used to produce predictions about unmeasured candidate locations in a 2D map. An active learning approach is used to determine which candidate locations are most informative given such predictions. New measurements are acquired and predictions are updated in light of this. This method can be run in real time as an alternative to conventional grid scanning, outperforming it by over 3.7 times. I demonstrated that the method is of broad applicability by applying it to different current maps; those described in section 3.7.1 and section 3.8. I have also demonstrated another capability of the algorithm by replacing the CVAE with a physical model (section 3.9.2).

A significant limitation of this approach is that no speedup should be expected when performing measurements that do not match the predictions made by the model. If a CVAE is used care should be taken to verify that its training data set is representative of measurements that will be made. If using a physics model care should be taken when designing this to ensure its predictions are an accurate representation of real measurements. This also limits the utility of this approach

in its current form for making efficient measurements of novel or unexpected device behaviour. Additionally, care should be taken when extending this approach to more complex current maps to verify that the CVAE isn't just memorising members of its training data set.

This work could be extended by considering more than just two parameters in the experimental design space. Parameters like the number of shots or integration time could be considered - this would be particularly useful in the case of measuring Rabi chevrons in section 3.9 where each individual measurement takes ~ 1 s.

4

Automatic coarse tuning of a quantum device

Device parameters define a high-dimensional space. This parameter space is typically explored manually by taking grid scan measurements (section 1.8.1). However, only in a small region of this space will a device be rendered operational. Device parameters are explored until such region is found. This process takes hours [83] even for the most experienced operators.

In this chapter I analyse the gate voltage parameter space of electrostatically defined quantum dot devices. I introduce a statistical search algorithm that efficiently explores this space with little prior knowledge of the device. The algorithm is designed to efficiently find double quantum dot regimes.

4.1 Devices

The quantum dot devices used for this chapter are defined in a 2-dimensional electron gas in a GaAs/AlGaAs heterostructure by Ti/Au gate electrodes. DC voltages are applied to these gate electrodes, V_1 to V_8 , to create a lateral confinement potential for electrons. Of particular importance are gate voltages V_3 and V_7 , which mainly tune the electron occupation of the left and right dot. A bias voltage V_{bias}

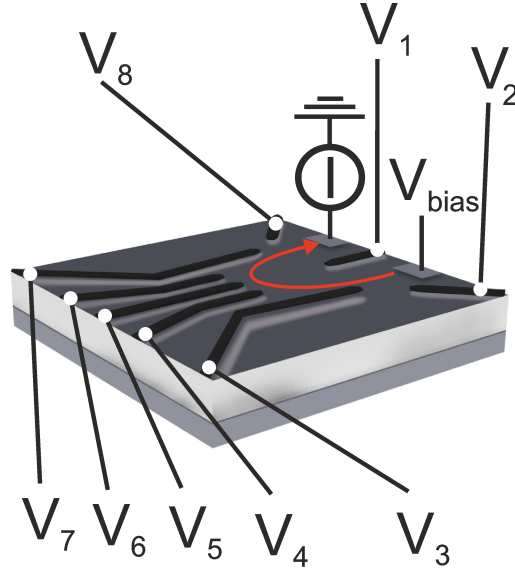


Figure 4.1: Device architecture Schematic of the devices. A bias voltage V_{bias} is applied between ohmic contacts to the two-dimensional electron gas. Voltages labelled V_1 to V_8 are applied to the gates.

is applied to ohmic contacts to drive a current (I) through the device. The device schematic is shown in Fig. 4.1. Measurements were performed at 50 mK.

To demonstrate the algorithm’s generality and to better understand its performance, the algorithm was run several times on two different devices with nominally identical gate architecture, Devices 1 and 2. While for Device 1 I considered the eight-dimensional parameter space defined by all its gate electrodes, for Device 2 I excluded gate electrode 6 by setting $V_6 = 0$ mV due to observed leakage currents associated with this gate which indicate the gate cannot be used (see section 1.7.3 for background).

4.2 The gate voltage space

To design a flexible tuning algorithm the structure of the gate voltage space is considered. For the majority of the gate voltage space these devices operate as transistors. They allow current to flow for values of $V = (V_1, \dots, V_8)$ close to 0 V, and block the flow of current for values close to $V_{\text{min}} = -2$ V. V_{min} is the minimum voltage (i.e. maximum negative voltage) that can be applied to the device safely

avoiding leakage currents. Device characterisation relies on monitoring current measurements while input gate voltages are varied. In the regime where current cannot flow, the device can not be characterised by transport measurements. I refer to this regime as pinch off or the ‘pinched’ regime.

A confinement potential requires the formation of well defined tunnel barriers. These tunnel barriers will increase the resistance from source to drain and thus the current flow through the device will diminish for a given bias voltage. As the magnitude of the gate voltages is increased, the tunnel barriers might become taller and thicker and the device resistance would grow significantly until the current is unobservable and the pinched regime is entered.

These observations lead to the following points - transport features indicating quantum dot formation are expected to be found near the boundary that separates the pinched regime from the high current regime. I refer to this boundary as the hypersurface and mathematically define it in section 4.3. I plot a three dimensional slice of the hypersurface in Fig. 4.2. Current maps at different locations near the surface show that both single and double quantum dot regimes are present. This surface was acquired by performing a three dimensional $100 \times 100 \times 20$ grid scan (current as a function of V_2 and V_8 measured as a function of V_5). The hypersurface definition (see section 4.3) was applied to the $100 \times 100 \times 20$ array. The resulting segmentation map has a size $100 \times 100 \times 20$. This map was turned into a mesh by applying the marching cubes algorithm [84]. The current threshold used to define this hypersurface was 20% of the maximum measured current.

The structure of the hypersurface is cube-like, with each face corresponding to a different gate electrode dominating pinch-off. The hypersurface can be thought of as containing information about which gate electrodes contribute to forming tunnel barriers in the hypersurface normal vectors. I developed a method for tuning of tunnel barriers which I call *simple tuner* (see appendix .5). This method is simple and efficient in high dimensional systems with close to orthogonal barrier gates.

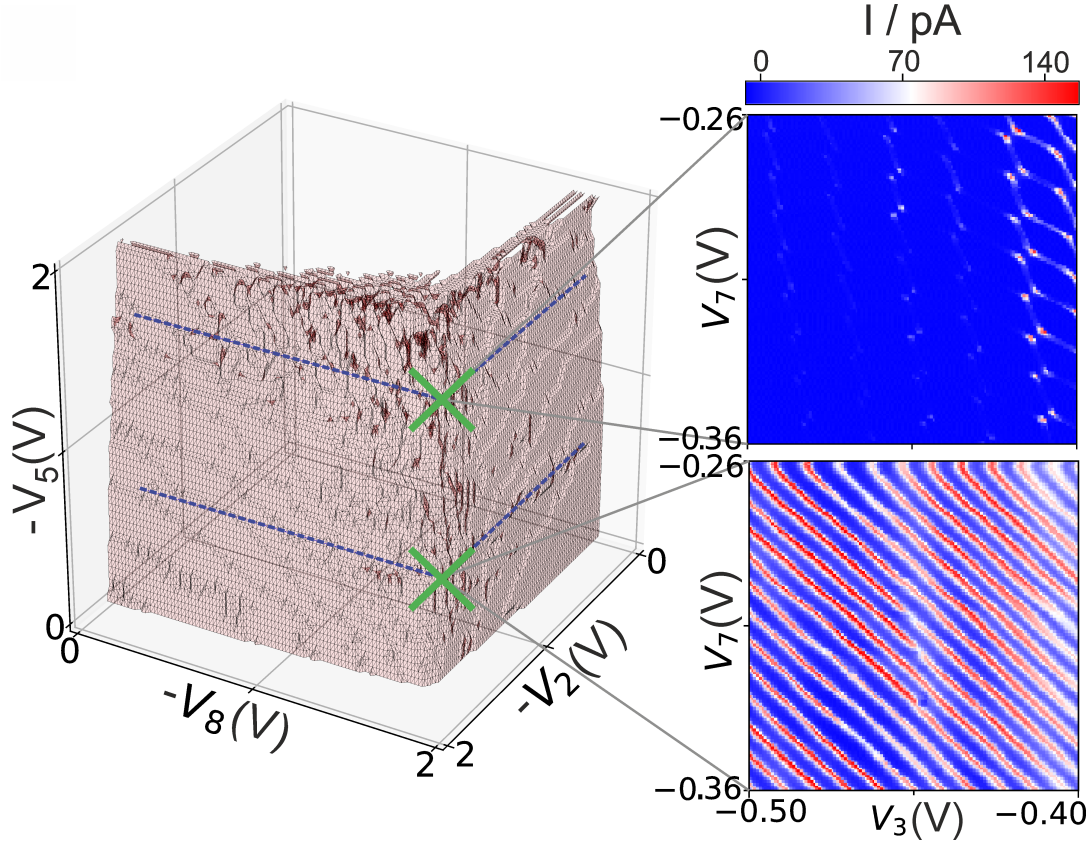


Figure 4.2: The hypersurface. Left: Boundary hypersurface measured as a function of V_2 , V_5 , and V_8 , for fixed values of V_1 , V_3 , V_4 , V_6 , and V_7 . The gate voltage parameter space, restricted to 3D for illustration, contains small regions in which double and single quantum dot transport features can be found. These regions typically appear darker in this representation because they produce complex boundaries. Right: For particular gate voltage locations marked with green crosses, the current as a function of V_7 and V_3 is displayed. The top and bottom current maps display double and single quantum dot transport features, respectively.

4.3 Hypersurface definition

The definition of a hypersurface is used extensively throughout this work so I shall mathematically formulate it. The hypersurface is a $d - 1$ dimensional sub-manifold in gate voltage space (where d is the number of gate voltages). Voltages $\mathbf{v}_s$ can be applied to these gates where $\mathbf{v}_s \in \mathcal{V}$. The set $\mathcal{V}$ contains all safe gate voltages that can be applied to the device (this safety is put in place in order to avoid leakage currents). The hypersurface can be found by projecting a set of unit vectors U_s from an origin $\mathbf{o}$ onto the hypersurface, having set a fixed bias voltage V_{bias} .

In this context, the current through the device can be expressed as a function of all gate voltages; $I(\mathbf{v}_s)$. For a given unit vector $\mathbf{u} \in U_s$, I will consider a one dimensional ray from the origin $I_{\mathbf{u}}(r) = I(\mathbf{u}r + \mathbf{o})$ where r is the distance from that origin. Now to identify if the device is in the pinched regime or not, I introduce a current threshold I_{thresh} defining a simple classification function

$$I_c(\mathbf{v}_s) = \begin{cases} 1 & I(\mathbf{v}_s) > I_{\text{thresh}} \\ 0 & I(\mathbf{v}_s) \leq I_{\text{thresh}} \end{cases}. \quad (4.1)$$

The value of r that intersects the hypersurface can thus be expressed as

$$r_{\text{max}}(\mathbf{u}) = \max\{r \in \mathbb{R}^+ : I_c(\mathbf{u}r + \mathbf{o}) = 1\}, \quad (4.2)$$

and the location of gate voltage space that sits on the hypersurface would be $s(\mathbf{u}) = r_{\text{max}}(\mathbf{u})\mathbf{u} + \mathbf{o}$. The hypersurface is found by considering

$$S \equiv \{s(\mathbf{u})\}_{\mathbf{u} \in U_s}, \quad (4.3)$$

where the set U_s contains all unit vectors that intersect the hypersurface, sampled so that intersections with the surface are uniformly distributed.

4.4 A search based algorithm for double quantum dot tuning

I introduce a statistical search algorithm with the goal of finding a region in the device parameter space that contains double quantum dot features. An active learning approach like that used in chapter 3 cannot be used because to specify the model G and make real time decisions we either require training data that is too costly to acquire or a physics model that is too computationally expensive and inaccurate for making decisions [85]. Instead the algorithm approaches device tuning as a multi armed bandit problem (section 1.10), using heuristics and Thompson sampling to improve the probability of reaching its goal. The algorithm only leverages knowledge about the structure of the gate voltage space. This knowledge is incorporated as modeling assumptions in two ways: (i) single and double quantum

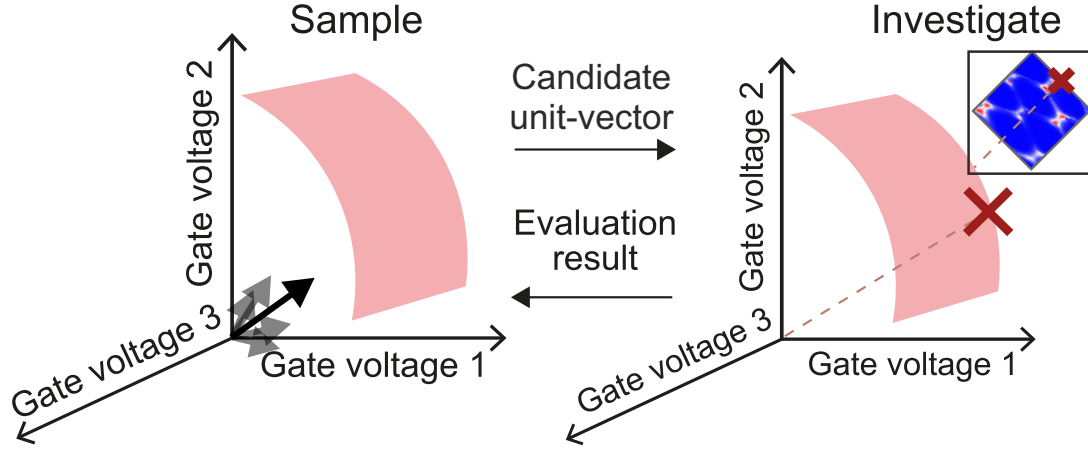


Figure 4.3: Algorithm overview. The sampling stage produces candidate unit vectors and selects one. The investigation stage evaluates the candidate by, for example, measuring current maps (see inset) which are evaluated by a score function. The current map displayed is an example of a measurement performed by the algorithm. It uses a colour scale running from red, the highest current measured, to blue, the lowest current. Evaluation results are fed back to the sampling stage. The algorithm adheres loosely to the general workflow detailed in section 2.4; the roles of making measurements and processing signals are combined in the investigation stage.

dot transport features are embedded near a boundary hypersurface, which separates regions in which a measurable current flows, from regions in which the current vanishes; (ii) large regions of this hypersurface do not display transport features.

The algorithm consists of two main components: a sampling stage that generates candidate unit-vectors (section 4.6), and an investigation stage in which transport measurements associated with a given unit-vector are performed (section 4.5). Based on the measurements performed, the investigation stage attempts to determine if the device is in a desirable location in gate voltage space (Fig. 4.3). The results of the investigation stage are feed back into the sampler, which chooses a new candidate unit vector in the light of this information. Finally to improve the quality of observations made by the investigation stage, a pruning heuristic is applied (section 4.7).

4.5 Investigation stage

The investigation stage consists of a sequence of parametrised experiments. Its role is twofold: firstly, each experiment provides information about the device regime so

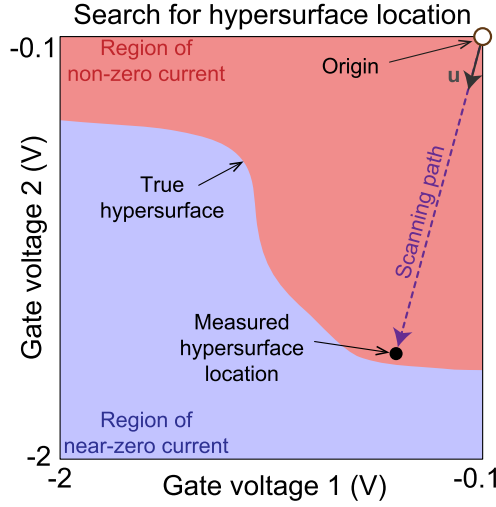


Figure 4.4: Search for the hypersurface. The gate voltage space, restricted to two dimensions for illustration, is divided into regions of near-zero (blue) and non-zero current (pink), separated by a boundary hypersurface. The gate voltages are scanned along a ray (dashed violet arrow) starting at the origin (white circle) and defined by a direction $\mathbf{u}$. By monitoring the current, the intersection with the hypersurface is detected.

the sampler can learn; secondly, it reduces the number of measurements required to determine if the device is in a double quantum dot regime. If an experiment evidences that the device is not in a promising regime, the sequence of experiments is terminated and new gate voltage parameters are selected to be explored by the sampler, reducing the amount of time required to investigate the device. The specific sequence of experiments used are referred to as: search for the hypersurface (section 4.5.1), Coulomb peak check (section 4.5.2), low resolution scan (section 4.5.3), and high resolution scan (section 4.5.4).

4.5.1 Search for the hypersurface

For a given unit vector provided by the sampling stage, this search (Fig. 4.4) aims to efficiently identify the location in gate voltage space of the hypersurface as defined in section 4.3. For each candidate unit-vector $\mathbf{u}$, a linear search is used to find $s(\mathbf{u})$.

From the origin $\mathbf{o}$ gate voltages are incremented by δ and the condition $I_{c,j} = I_c(j\delta\mathbf{u} + \mathbf{o})$ is checked for $j = 0, \dots, J$. The condition is considered inside a sliding window of $K_s\delta$ where K_s is an integer specified by the user. This sliding window

is $W_k = (I_{c,j})_{j=k,\dots,k+K_s}$ for step k in the search. Pinch-off is identified as the point $s(\mathbf{u}) \approx K\delta\mathbf{u} + \mathbf{o}$ where $K = \min\{k \in \mathbb{Z}^+ : W_k = 0\}$. In essence, the search moves along a ray, checking the current every δ . It halts when the search finds the current has fallen below the threshold I_{thresh} for $K_s\delta$. In this chapter, I_{thresh} is set to 20% the maximum current, $\delta = 10 \text{ mV}$, and $K_s = 5$. These parameters were fixed but can also be subject to optimisation.

If the hypersurface is not found within the safe bounds of $[V_{\text{min}}, \mathbf{o}]$ then the investigation stage exits and the sampler suggests a new unit-vector. Otherwise, the investigation stage continues with the Coulomb peak check.

4.5.2 Coulomb peak check

The Coulomb peak check is paramaterised by a location in gate voltage space $\mathbf{v}$. The algorithm sets $\mathbf{v} = s(\mathbf{u})$. The nearby region of gate voltage space is investigated to determine whether it corresponds to the single-electron tunnelling regime (Fig. 4.5). This measurement is carried out in the plane containing $s(\mathbf{u})$, defined by sweeping the two plunger gate voltages $\hat{V}_3$ and $\hat{V}_7$. These gate axes are selected before running the algorithm, as they correspond to gate electrodes designed to predominantly shift the electrochemical potential in the left and right dots. A one-dimensional grid scan is performed in this investigation plane, starting at $s(\mathbf{u})$ and running along the diagonal axis $\hat{V}_e \equiv \frac{1}{\sqrt{2}}(\hat{V}_3 + \hat{V}_7)$. This scan is chosen to have a length of 128 mV and a resolution of 1 mV.

A peak detection routine is applied to the 128 current measurements that constitute a current trace in order to detect the presence of Coulomb peaks. The method used is a simple local maxima approach implemented in scipy's [\[86\]](#) `find_peaks` method. The method detects the location of peaks by inspecting the magnitude of neighboring measurements in parameter space. This method alone is highly susceptible to noise fluctuations, which cause false positives. To mitigate this, the prominence of peaks is estimated and used to reject peaks below a certain prominence threshold. The prominence quantifies the height of a peak relative to

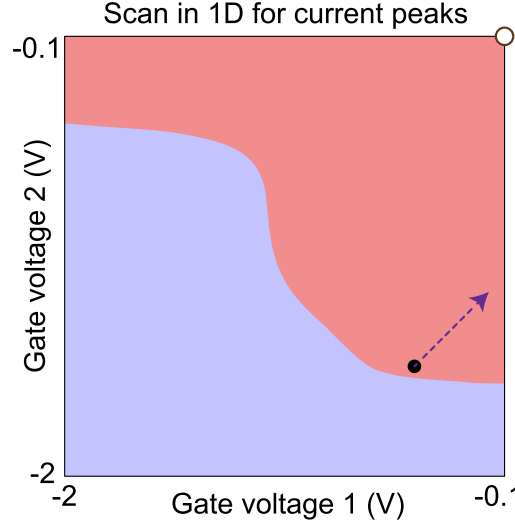


Figure 4.5: Coulomb peak check. Gate voltage space illustrated as before (Fig. 4.4) for the same $\mathbf{u}$. Based on a short 1D scan, the gate voltage location is classified according to whether it shows current peaks indicating Coulomb blockade.

its surroundings. Other settings to further improve peak detection performance were implemented but had little impact in the algorithm’s performance.

The peak detection returns the locations in gate voltage space of all detected peaks. If no peaks are detected, the investigation stage exits and the sampler suggests a new unit-vector. Otherwise, the investigation stage continues to a low resolution scan.

4.5.3 Low resolution scan

If Coulomb peaks are detected during the Coulomb peak check, a two-dimensional grid scan is performed (Fig. 4.6). The scanning region is a square oriented along $\hat{V}_e$ and its orthogonal axis $\hat{V}_a \equiv \frac{1}{2}\sqrt{(\hat{V}_3 - \hat{V}_7)}$. The square is delimited so that one side is centered in $s(\mathbf{u})$, and its side length is chosen to be 3.5 times the average peak spacing identified during the Coulomb peak check. If the Coulomb peak check shows less than 3 peaks, the side length defaults to 100 mV.

After performing the grid scan, a score (section 4.5.3) is assigned to the resulting current map. If the score exceeds a threshold γ_{score} then the investigation stage continues to the high resolution scan. If this condition is not met, the investigation

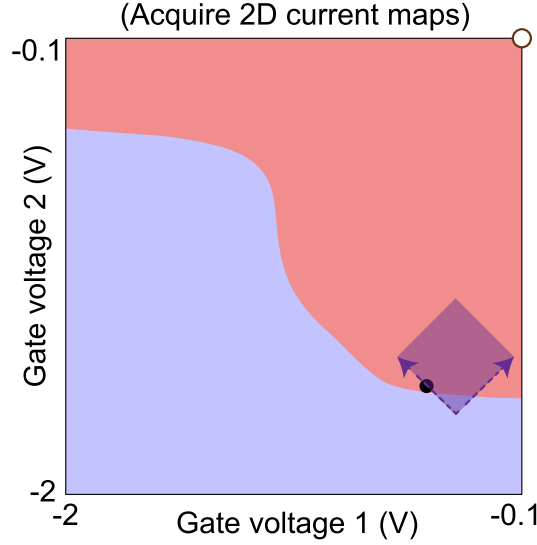


Figure 4.6: Low and high resolution scans Gate voltage space illustrated as before (Fig. 4.4) for the same $\mathbf{u}$. If Coulomb peaks are found, a two-dimensional scan (violet square) is performed in the plane of V_3 and V_7 , and is possibly repeated at higher resolution.

stage exits and the sampler suggests a new unit-vector. The score function threshold dynamically adjusts so that only the top 15% of low resolution scans successfully reach the high resolution scan step. The dynamic threshold ensures that high resolution scanning does not become a bottleneck for the algorithm's run time. The dynamic threshold is γ_{85} , the 0.85 quantile of all previously observed scores. A minimum cap on the score threshold is applied:

$$\gamma_{\text{score}} = \max(\gamma_{85}, \gamma_{\min}), \quad (4.4)$$

where $\gamma_{\min}$ is the minimum allowable value, set to 0.001 throughout the experiments in this chapter. The role of the score function is that of a flexible classification method; I do not rank, optimise, or prefer scans based on their score as long as they pass the threshold γ_{score} .

Score function

The score function is designed to distinguish between measurements containing double quantum dot features, single quantum dot features, and measurements for which the confinement potential is not well defined. This could be achieved using

modern machine learning techniques [36–39]. However, the score function I use is not reliant on training data, and hence is less device and labeller specific. The score consists of three separate scores multiplied: the orientation of gradients score, the sharpness score, and the fit direction score.

The orientation of gradient score (og score) aims to capture if Coulomb peaks appear as straight lines in a current map or if a honeycomb pattern is present. This is achieved by taking the numerical derivative of the current map and producing gradient unit vectors. These gradient vectors will exhibit a distribution of angles. If Coulomb peaks appear as straight lines, then the distribution of angles will reveal two high-density collections of unit vectors separated by π rad from each other. In this case, the high-density collections will have a small variance and a single line can fit the two distributions they define. For a honeycomb pattern, this variance will be larger or the high-density collections will not be well defined, thus two lines will better describe the distributions. The value of the gradient orientation score is defined as the difference between the average of residuals corresponding to a fit through the high-density collections. Gradients smaller than a certain value are not considered for fitting. This threshold was inferred from the current noise.

The sharpness score aims to prioritise ‘sharp’ Coulomb peaks. To compute the sharpness score the current map is first split into 16 smaller tiles. Each tile is segmented into regions of high current and low current using a simple threshold. Then, the second order derivative of the current as a function of gate voltage is computed. This second order derivative is averaged over the high current regions and its standard deviation is calculated. The score for each tile is defined as the product of these average and standard deviation values. The scores for the 16 tiles are then averaged together giving the final sharpness score.

The fit direction score (direction score) aims to further differentiate between single and double quantum dot behavior. The current map is first split into 16 smaller tiles, in the same way as for the sharpness score. For each of these tiles, a line is fit to the high-density collections of gradient vectors, in the same way as for the orientation of gradient score. From this fit, a gradient vector angle is assigned

to the tile. For each row of tiles (in direction $\hat{V}_e$), the standard deviation of gradient vector angles is calculated. In this way, the array of 4×4 tiles is reduced to a 1×4 vector. The average of the elements in this vector defines the fit direction score.

4.5.4 High resolution scan

If the score function threshold is passed a new two-dimensional grid scan is performed. This scan is acquired over the same gate voltage region than the low resolution scan but with a higher resolution; 48×48 pixels. All high resolution scans are stored for future reference. These scans can either be processed algorithmically or a human user can look at them in order to identify promising scans. After the scan is complete, the investigation stage exits and the sampler suggests a new unit-vector.

4.6 Sampling stage

The sampling stage or sampler aims to efficiently search candidate gate voltage locations, increasing the probability that a double quantum dot regime is investigated. It does this by modelling the behavior of the device given information provided by the investigation stage. Models are then sampled to propose candidate unit-vectors that are more likely to contain double quantum dot features.

4.6.1 Modelling the behavior of the device with a Gaussian process

At a given iteration j , the investigation stage produces a sequence of results $Y_j = (r_{\max,j}, Y_{\text{Surf},j}, Y_{\text{CP},j}, Y_{\text{SF},j})$, where $r_{\max,j}$ is the distance from the origin found by the search for the surface, Y_{Surf} is a label denoting whether the surface was found during the search for the surface, Y_{CP} is a label denoting whether any peaks were found in the Coulomb peak check, and Y_{SF} is a label denoting whether the score function threshold was passed in the low resolution scan. The high resolution scan does not return any information. All information gathered up until iteration i is stored by the algorithm as $Y_{1:i} = (Y_j)_{j=1,\dots,i}$ along with all proposed unit vectors $X_{1:i} = (\mathbf{u}_j)_{j=1,\dots,i}$.

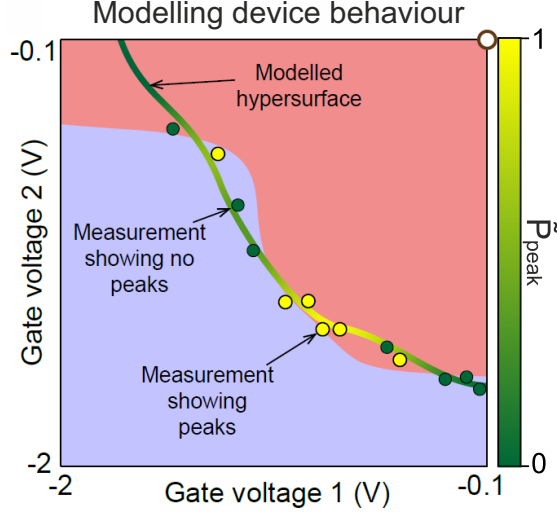


Figure 4.7: Modelling the device. Gate voltage space illustrated as before (Fig. 4.4). Based on the first 30 measurements (green and yellow circles), the algorithm builds a model of the hypersurface and assigns a probability $\tilde{P}_{\text{peak}}$ that Coulomb peaks will be found.

At the end of a given iteration i , the sampler models the device using a set of models $\tilde{P} = \{\tilde{p}(r|\mathbf{u}), \tilde{P}_{\text{valid}}(\mathbf{v}), \tilde{P}_{\text{peak}|\text{valid}}(\mathbf{v})\}$, which predict outputs of the investigation stage $\hat{Y} = \{\hat{r}_{\text{max}}, \hat{Y}_{\text{Surf}}, \hat{Y}_{\text{CP}}\}$ (quantities are the same as defined in Y_j , hats denote that they are predictions) given an input X by using $Y_{1:i}$ and $X_{1:i}$ as training data (Fig. 4.7). The algorithm does not attempt to predict if the score function is passed ($\hat{Y}_{\text{SF}}$) because limited data is available and because the scoring threshold is dynamic.

All models in $\tilde{P}$ rely on Gaussian processes: $\tilde{p}(r|\mathbf{u})$ uses a Gaussian process regression model to predict the hypersurface, while $\tilde{P}_{\text{valid}}(\mathbf{v})$ and $\tilde{P}_{\text{peak}|\text{valid}}(\mathbf{v})$ use Gaussian process classification models which predict if pinch off will be found and if Coulomb peaks will be found. Each Gaussian process requires a prior distribution, which is defined by a prior mean function and a covariance kernel. Details about the kernel, mean function, priors, and optimisation of hyperparameters are provided in appendix .6.

The model $\tilde{p}(r|\mathbf{u})$ is used to improve the efficiency of the search for the hypersurface (section 4.5.1). Instead of beginning at the origin (as illustrated in Fig. 4.4), the origin of the search is adjusted to a temporary origin $\mathbf{o}_{\text{adj}} = (m_r(\mathbf{u}) - 2\sigma_r(\mathbf{u}))\mathbf{u} + \mathbf{o}$,

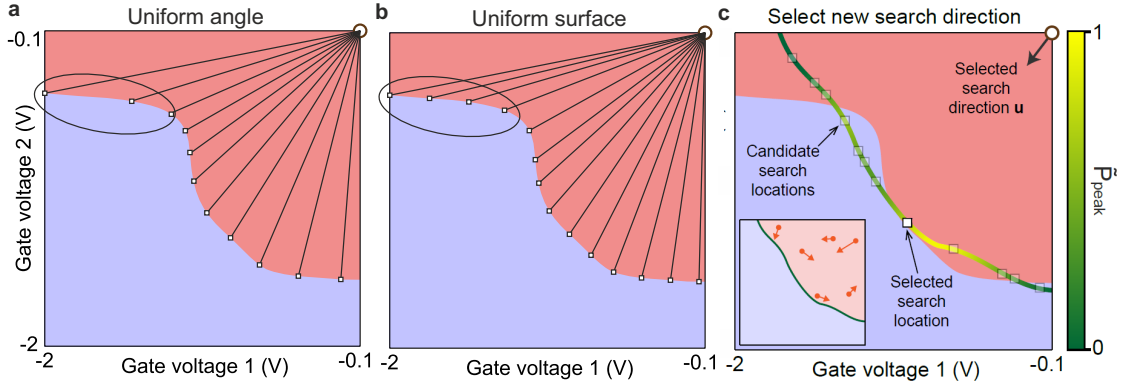


Figure 4.8: Sampling a unit-vector. Gate voltage space illustrated as before (Fig. 4.4). **a** Uniform angles. Uniform angles are projected onto the hypersurface. The oval highlights an under sampled region of the hypersurface. **b** Uniform hypersurface. Uniform locations on the hypersurface are projected back to the origin. The oval highlights a uniform sample density. **c** Selecting a unit vector. The algorithm generates a set of candidate search locations (squares), each one weighted by its corresponding value of $\tilde{P}_{\text{peak}}$, and selected at random. The unit-vector $\mathbf{u}$ corresponding to that location is then passed to the investigation stage. Inset: Numerically sampling the hypersurface using simulated Brownian motion. Each point represents a simulated particle moving inside an enclosed volume defined by the hypersurface. The collisions between the particles and the modelled hypersurface generate a set of candidate search locations.

where $m(\mathbf{u})$ is the mean and $\sigma(\mathbf{u})$ the standard deviation of the distribution $\tilde{p}(r|\mathbf{u})$. The new search begins at $\mathbf{o}_{\text{adj}}$ instead of $\mathbf{o}$ saving a significant amount of measurement time. Occasionally, the measured current at the beginning of the scan is below the current threshold, indicating that $\mathbf{o}_{\text{adj}}$ is already in the pinched-off region. In these cases, the algorithm scans towards the origin, along $-\mathbf{u}$. Once the measured current increases above 80% of the value at $\mathbf{o}$, the algorithm reverts to measuring in the $\mathbf{u}$ direction to locate the hypersurface in the usual way.

4.6.2 Generating samples using the models

The sampler produces a new $\mathbf{u}$ every iteration for the investigation stage to explore. The sampler has no information when the algorithm is started, so it begins with an initialisation state. During the initialisation state, unit vectors on a unit hypersphere are drawn uniformly in angle (this step does not require the models from the previous section). The initialisation lasts for the first 30 iterations of the investigation stage.

After the initialisation state is over, the sampler generates values of $\mathbf{u}$ associated with high $\tilde{P}_{\text{peak}}$, while occasionally exploring less promising, unexplored regions of the hypersurface. To achieve this exploration versus exploitation trade-off, the algorithm first generates a set of candidate search locations on the hypersurface. Generating a set of uniform samples on a complicated, arbitrary shaped hypersurface is challenging. The naive approach of sampling unit-vectors at random from a unit hyper-sphere results in oversampling and under-sampling of regions of the hypersurface, as shown in Fig. 4.8a. I used a numerical MCMC approach to sample from the predicted hypersurface [87]. To generate a set of samples that is approximately uniform despite the convoluted shape of the hypersurface, the Brownian motion of ‘particles’ inside a volume defined by the hypersurface is simulated. Each particle encounter with the hypersurface contributes one candidate location (Fig. 4.8c). A full description of the hypersurface sampling approach is detailed in section 4.6.2.

To each candidate location, the algorithm assigns a weight proportional to the corresponding value of $\tilde{P}_{\text{peak}}(\mathbf{v})$, and selects one location at random. This process is referred to as Thompson sampling or probability matching. A description of how this process relates to Thompson sampling is in appendix .8. In essence, this algorithm attempts to draw samples from $P(\mathbf{s}|\tilde{P}_{\text{peak}} = \text{True})$, where $\mathbf{s} \in S$ (see section 4.3). Because models $\tilde{P}_{\text{valid}}$ and $\tilde{P}_{\text{peak}|\text{valid}}(\mathbf{v})$ are used to build $\tilde{P}_{\text{peak}}(\mathbf{v})$ and they have uniform priors, the algorithm will start by choosing candidate locations on the hypersurface approximately uniformly. However, as observations are made, regions of the parameter space are excluded, either because no pinch off was identified or because no Coulomb peaks were detected.

Brownian motion for hypersurface sampling

The procedure used to sample the surface implements the method in [87]. The Gaussian process model of the hypersurface $\tilde{p}(r|\mathbf{u})$ is used to approximate the real hypersurface during sampling. To start, N_p particles are initialised inside the predicted (high current) volume of the hypersurface at the origin $\mathbf{o}$. After the volume of the hypersurface has been populated by the set of particles $V_p = \{\mathbf{v}_{p,j}\}_{j=1,\dots,N_p}$,

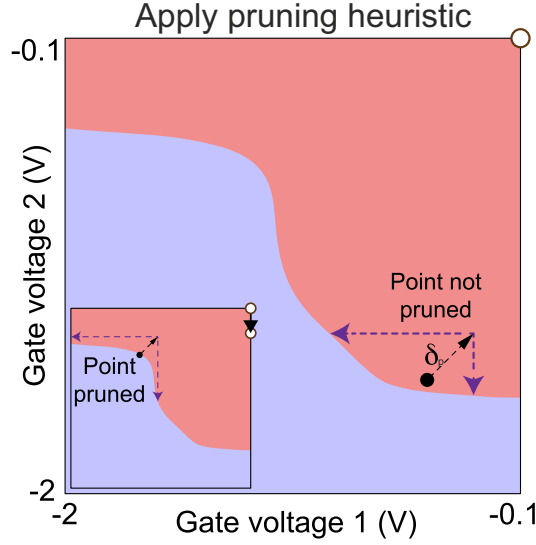


Figure 4.9: Pruning heuristic Gate voltage space illustrated as before (Fig. 4.4). To determine whether a region should be pruned, the algorithm scans each gate voltage individually toward the maximum of its range from a location back stepped from the hypersurface. If only one scan intersects the hypersurface (as shown in the inset), future exploration of that region is avoided by displacing the origin as indicated with a black arrow.

their Brownian motion is simulated. For this simulation, a displacement vector $\Delta \mathbf{V}_p \sim \mathcal{N}(\mathbf{0}, \sigma_b^2)$ is defined and N_p samples are drawn from it $\{\Delta \mathbf{v}_{p,j}\}_{j=1,\dots,N_p}$. For the j th particle if the location $\mathbf{p}_j = \mathbf{v}_{p,j} + \Delta \mathbf{v}_{p,j}$ is inside the volume the move is accepted and $\mathbf{v}_{p,j}$ will move to $\mathbf{p}_j$, however if $\mathbf{p}_j$ is outside the volume the move is rejected, $\mathbf{p}_j$ is added to a set $\mathcal{P} \cup \{\mathbf{p}_j\}$, and $\mathbf{v}_{p,j}$ will not be moved. If the j th particle $\mathbf{p}_j \notin \mathcal{V}$, the safe gate voltage space, then the particle is reset to the origin $\mathbf{o}$. This process is repeated, adding more elements to the set $\mathcal{P}$ (this is performed in parallel to $\mathcal{P}$). Elements in $\mathcal{P}$ are approximately uniform samples from S . In this chapter $N_p = 200$ and $\sigma_b = 25$ mV.

4.7 The pruning heuristic

Finally, to improve the quality of the search in the initialisation stage, a simple heuristic is applied. The pruning heuristic aims to remove regions of parameter space where only one gate dominates the pinch off behavior. From an hypersurface location $s(\mathbf{u})$, all voltages are stepped upwards to a location $s^{\delta_p}(\mathbf{u}) \equiv s(\mathbf{u}) + \delta_p$,

where δ_p is a step-back vector with each component chosen to be +100 mV. Each voltage in turn is then swept downwards towards the maximum of its range or until the hypersurface is encountered. If the hypersurface is encountered only along one voltage axis η , then the origin for subsequent iterations is moved so that its η -component is equal to the η -component of s^{δ_p} (Fig. 4.9 inset). Over several iterations, this process prunes away search paths for which the hypersurface is not present within the chosen gate voltage range.

4.8 Algorithm Hyperparameters

This algorithm contains many components, some of which require the user to specify hyperparameters that define the algorithms operation. In this section I list hyperparameters and describe what the hyperparameters are responsible for. I also comment on the hyperparameters sensitivity, known failure modes, and how reasonable values can be estimated:

- Origin (**o** see section 4.5.1)

Description: The starting origin of the algorithm and the highest gate voltage values that can be applied by the algorithm to the device.

Sensitivity: Medium sensitivity.

Failure modes: When incorrectly specified the main failure mode is that the search space of interest is excluded from the search. Usually evidenced by measurements containing only low current transport features or noise.

Estimation: Can be motivated by setting to maximum safe gate voltage which can be deduced from knowledge about the device material system, design and architecture.

- Bound ($V_{\min}$ see section 4.5.1)

Description: The lowest gate voltage values that can be applied by the algorithm to the device.

Sensitivity: Medium sensitivity.

Failure modes: When incorrectly specified the main failure mode is the search space of interest is excluded from the search. Usually evidenced by measurements containing only high current transport features.

Estimation: Can be easily motivated by setting to minimum safe gate voltage which can be deduced from knowledge about the device material system, design and architecture.

- Pinch threshold (I_{thresh} see section 4.3)

Description: The lowest gate voltage values that can be applied by the algorithm to the device.

Sensitivity: High sensitivity.

Failure modes: When incorrectly specified the hypersurface is not correctly specified and does not exist or is not correlated with desirable transport features. Evidenced by failure to observe pinch off when transistor behavior is clearly visible in the search for the hypersurface.

Estimation: A reasonable value can be easily deduced by considering $I_{\text{high}} = I(\mathbf{o})$ and $I_{\text{low}} = I(V_{\text{min}})$ and taking $I_{\text{thresh}} = I_{\text{low}} + a(I_{\text{high}} - I_{\text{low}})$. I select $a \approx 0.2$ through out this chapter.

- Delta (δ see section 4.5.1)

Description: The distance in gate voltage space stepped after every measurement when searching for the hypersurface.

Sensitivity: High sensitivity.

Failure modes: When incorrectly specified there are two failure modes. When delta is too small locating the hypersurface will take a disproportionate amount of measurement time and the algorithm will be slow. When delta is too large locating the hypersurface will be inaccurate, evidenced by pinch off curves that switch from on to off in one step.

Estimation: A value of delta can be selected by considering the expected size of desired features in gate voltage space given expected device characteristics like charging energy and lever arms. I recommend a value of order $\frac{e}{C_g}$ (see section 1.5) or lower.

- Length after (K_s see section 4.5.1)

Description: The number of measurements made by the search for the hypersurface to confirm the surface has been identified.

Sensitivity: Low sensitivity.

Failure modes: No major failure modes.

Estimation: A value of length after can be selected by considering the expected size of desired features in gate voltage space given expected device characteristics like charging energy and lever arms.

- Investigation gates (see section 4.5)

Description: The gate electrodes used for investigation the gate voltage space.

Sensitivity: High sensitivity.

Failure modes: When incorrectly selected investigation stage will not measure desirable features. Evidenced by little to no none-monotonic measurements being made by the coulomb peak check.

Estimation: Investigation gates can be selected by considering the gate architecture design of the device. Gates that operate as plungers should be selected.

- Length of coulomb peak scan (see section 4.5.2).

Description: Length of scan in gate voltage space.

Sensitivity: Low sensitivity.

Failure modes: When incorrectly specified scan can be too short or too long to capture coulomb oscillations. Evidenced by measurements containing only low current transport features or noise or by signals that are not smoothly varying.

Estimation: A value can be selected by considering the expected size spacing between coulomb oscillations in gate voltage space given the devices expected charging energy and lever arms. It should be set so that the coulomb peak scan captures multiple coulomb oscillations. I recommend a value of order $10 \times \frac{e}{C_g}$ (see section 1.5) or lower.

- Number of measurements in coulomb peak scan (see section 4.5.2).

Description: Number of measurements in coulomb peak scan.

Sensitivity: Low sensitivity.

Failure modes: When incorrectly specified scan can be too sparse to observe quickly changing signals like coulomb oscillations. Evidenced by measurements containing only low current transport features or noise.

Estimation: A value can be selected by considering the expected size spacing between coulomb oscillations in gate voltage space given the devices expected charging energy and lever arms. It should be set so that the coulomb peak scan captures multiple coulomb oscillations. I recommend a value of order $\frac{1}{10} \times \frac{e}{C_g}$ (see section 1.5) or lower.

- Peak detection prominence (see section 4.5.2).

Description: Prominence required for positive detection of peak.

Sensitivity: Medium sensitivity.

Failure modes: When incorrectly specified peak detection becomes unreliable, either returning false positives or false negatives depending on prominence setting.

Estimation: A reasonable value can be easily deduced by considering $I_{\text{high}} = I(\mathbf{o})$ and $I_{\text{low}} = I(V_{\text{min}})$ and taking $I_{\text{thresh}} = I_{\text{low}} + a(I_{\text{high}} - I_{\text{low}})$. I select $a \approx 0.02$ through out this chapter.

- Side length of low resolution scan (see section 4.5.3).

Description: Side length in gate voltage space of scan window.

Sensitivity: Low sensitivity.

Failure modes: When incorrectly specified scan can be too short to capture coulomb oscillations. Evidenced by measurements containing only low current transport features or noise or by signals that are not smoothly varying.

Estimation: A value can be selected by considering the expected size spacing between coulomb oscillations in gate voltage space given the devices expected charging energy and lever arms. It should be set so that the scan captures multiple coulomb oscillations. I recommend a value of order $3.5 \times \frac{e}{C_g}$ (see section 1.5).

- Number of measurements in low resolution scan (see section 4.5.3).

Description: Number of measurements in low resolution scan.

Sensitivity: Medium sensitivity.

Failure modes: When incorrectly specified if the number of measurements is too low the scan can be too sparse to observe quickly changing signals like coulomb oscillations. Evidenced by measurements containing only low current transport features or noise. If the number of measurements is too high the scan can take too long to provide a time saving benefit.

Estimation: A value can be selected by considering the expected size spacing between coulomb oscillations in gate voltage space given the devices expected charging energy and lever arms. It should be set so that the window captures multiple coulomb oscillations. I recommend a value of order 20 giving a resolution of $0.29 \frac{\text{pixels}}{\text{mV}}$ when $\frac{e}{C_g} = 20 \text{ mV}$.

- Score threshold min ($\gamma_{\min}$ see section 4.5.3).

Description: Minimum value of the score threshold.

Sensitivity: Low sensitivity.

Failure modes: No major failure modes.

Estimation: Can be set to zero, or a very small none zero value can be selected considering that sharpness score is typically very low for poor stability diagrams ($\sim 1e-8$).

- Score threshold quantile (see section 4.5.3).

Description: Quantile used to compute the dynamic threshold.

Sensitivity: Low sensitivity.

Failure modes: No major failure modes.

Estimation: A value that allows some high scoring stability diagrams through should be selected. I recommend a value of 0.85.

- Segmentation threshold (see section 4.5.3).

Description: Threshold used to compute sharpness score.

Sensitivity: Medium sensitivity.

Failure modes: If the value is set too high or too low coulomb oscillations will not be segmented and the sharpness score will always return zero. Evidenced by score measure always returning zero.

Estimation: A reasonable value can be easily deduced by considering $I_{\text{high}} = I(\mathbf{o})$ and $I_{\text{low}} = I(V_{\min})$ and taking $I_{\text{thresh}} = I_{\text{low}} + a(I_{\text{high}} - I_{\text{low}})$. I select $a \approx 0.05$ through out this chapter.

- Side length of high resolution scan (see section 4.5.4).

Description: Side length in gate voltage space of scan window.

Sensitivity: Low sensitivity.

Failure modes: When incorrectly specified scan can be too short to capture coulomb oscillations. Evidenced by measurements containing only low current transport features or noise or by signals that are not smoothly varying.

Estimation: Can be set to same or similar value as low resolution scan.

- Number of measurements in high resolution scan (see section 4.5.4).

Description: Number of measurements in high resolution scan.

Sensitivity: Low sensitivity.

Failure modes: When incorrectly specified if the number of measurements is too low the scan can be too sparse to observe quickly changing signals like coulomb oscillations. Evidenced by measurements containing only low current transport features or noise.

Estimation: A value can be selected by considering the expected size spacing between coulomb oscillations in gate voltage space given the devices expected charging energy and lever arms. It should be set so that the window captures multiple coulomb oscillations $0.86 \frac{\text{pixels}}{\text{mV}}$ when $\frac{e}{C_g} = 20 \text{ mV}$.

- Number of initialisation iterations (see section 4.6.2).

Description: The number of iterations to take before the sampler is used for decisions and the pruning is turned off.

Sensitivity: Low sensitivity.

Failure modes: When specified incorrectly if the value is too low then MAP estimate for GP hyperparameters might be poor leading to overconfident decisions. If set too large no time benefit is given by the sampler.

Estimation: A value of order $\sim 10\%$ the total number of iterations provides reasonable performance.

- GP priors(see appendix .6).

Description: The prior distribution used for the GP's

Sensitivity: Medium sensitivity.

Failure modes: If incorrectly specified decisions will suffer from overconfidence.

Estimation: Appendix .6 details how these can be derived from other algorithm hyperparameters.

- Displacement standard deviation (σ_b see section 4.6.2).

Description: The standard deviation of the transition kernel used in Brownian motion.

Sensitivity: Low sensitivity.

Failure modes: No major failure modes.

Estimation: A value of can be selected by considering the expected size of desired features in gate voltage space given expected device characteristics like charging energy and lever arms. I recommend a value of 25 mV .

- Number of Brownian motion particles (N_p see section 4.6.2).

Description: The number of particles undergoing Brownian motion.

Sensitivity: Low sensitivity.

Failure modes: No major failure modes.

Estimation: A value of can be selected considering the computational limitations that the user has. I used 200 but far larger values could be considered.

- Step back vector (δ_p see section 4.7).

Description: The step back vector used in the pruning rule.

Sensitivity: Low sensitivity.

Failure modes: No major failure modes.

Estimation: A value of μ_t can be selected by considering the expected size of desired features in gate voltage space given expected device characteristics like charging energy and lever arms. I recommend a value of order $10 \times \frac{e}{C_g}$ (see section 1.5).

4.9 Analysis of expected tuning times and ablation study

The performance of the algorithm is assessed by a statistical analysis of the expected time to investigate a successful candidate unit-vector μ_t . This is the time it takes the algorithm to acquire a high-resolution current map that is confirmed a posteriori by humans as containing double quantum dot features. This confirmation is only needed to assess the performance of the algorithm. Because human labelling is subjective, three different researchers labelled all current maps, deciding in each case if they could identify features corresponding to the double quantum dot regime, with no other information available.

4.9.1 Labelling procedure

Two separate data sets were labelled. The first contained the 2048 current maps gathered from all the experiments performed in this chapter, except for the pure random experiment in section 4.9.4. The second data set contained 151 current maps, 51 of which were taken from the pure random experiment in section 4.9.4 and 100 were randomly selected from the first data set to increase the number of current maps. The current maps in each data set were randomly shuffled so that labellers could not identify which experiment produced a given current map. The labellers were the same for both data sets.

4.9.2 Inferring the algorithm's performance

To quantify the algorithm's performance, I infer the expected tuning times μ_t , and the probabilities $P(\text{peaks})$, and $P(\text{success}|\text{peaks})$. With this aim, I used Bayesian inference with a Jeffreys prior.

Inferring probabilities

Let p denote the probability of either $P(\text{peaks})$ or $P(\text{success}|\text{peaks})$ that I want to estimate. The Jeffreys prior for a binomial distribution is $p \sim \text{Beta}(0.5, 0.5)$. If I observe k events over n trials, then the posterior distribution of p is

$$p|k, n \sim \text{Beta}(0.5 + k, 0.5 + n - k). \quad (4.5)$$

Inferring the expected tuning times

To infer μ_t , the rate of events over time is modelled as a Poisson distribution $k \sim \text{Poisson}(\lambda t_{\text{tot}})$, where t_{tot} is the total time required for an algorithm run, and λ is a rate parameter. The expected time between two consecutive successes (i.e. the acquisition of currents maps displaying double quantum dot transport features) is $\mu_t = 1/\lambda$. The Jeffreys prior for a Poisson distribution is $\lambda \sim \text{Gamma}(0.5, 0)$. Consequently,

$$\mu_t|k, t_{\text{tot}} \sim \text{Inv-Gamma}(0.5 + k, t_{\text{tot}}). \quad (4.6)$$

Combining multiple labellers

Multiple labellers are used for inferring μ_t , $P(\text{peaks})$, and $P(\text{success}|\text{peaks})$. Labellers are presented with data $\mathcal{D} = \{C_i\}_{i=1, \dots, n}$, where C_i is a current map produced by the high resolution scan step (section 4.5.4). If no high-resolution scan was acquired for iteration i of the algorithm then C_i is an empty array. The data $\mathcal{D}$ requires a labelling function ψ_j : $\psi_j(C_i) = 0$ if there is no recognized double quantum dot transport features, and $\psi_j(C_i) = 1$ otherwise. The set of all labellers is $\Psi = \{\psi_j\}_{j=1, \dots, 3}$.

Labellers might not agree and so must be marginalised out. Let θ denote a parameter to be inferred (for examples μ_t , $P(\text{peaks})$, or $P(\text{success}|\text{peaks})$), then $p(\theta|\mathcal{D}, \psi) = \frac{p(\mathcal{D}|\theta, \psi)p(\theta)}{p(\mathcal{D})}$. Marginalising the likelihood, I obtain

$$p(\theta|\mathcal{D}) = \mathbb{E}_\psi[p(\theta|\mathcal{D}, \psi)] \approx \frac{1}{|\Psi|} \sum_{\psi \in \Psi} p(\theta|\mathcal{D}, \psi),$$

and a cumulative distribution

$$P(\theta < z|\mathcal{D}) \approx \frac{1}{|\Psi|} \sum_{\psi \in \Psi} P(\theta < z|\mathcal{D}, \psi).$$

4.9.3 The Pure random baseline

I introduce a naive search-based algorithm which I refer to as *Pure random*. This algorithm will be used as a baseline when determining the algorithm's performance. *Pure random* has an investigation stage. However, in this investigation stage only the Coulomb peak check is retained, the search for the hypersurface and the score function threshold are not used. This means the algorithm will perform a high resolution scan whenever a Coulomb peak is identified. All other elements of the investigation stage remain the same.

The sampler is replaced with one that draws gate voltage locations as $\mathbf{v} \sim U(V_{\min}, \mathbf{o})$, where U is a uniform distribution. The pruning rule is not applied.

4.9.4 Device tuning

I tested the performance of the tuning algorithms in devices 1 and 2. To benchmark the algorithms' performance, I define the average count $\bar{C}$ as the number of current maps labelled by humans as displaying double quantum dot features divided by the number of labellers. For a single run of the *Pure random* algorithm in Device 2 and five runs of the algorithm in Devices 1 and 2, I calculate $\bar{C}$ as a function of laboratory time (Fig. 4.10a,b). It is observed that $\bar{C}$ is vastly superior for the full algorithm compared with *Pure random*, evidencing the magnitude of the gate voltage space.

The time μ_t is estimated using the multi-labeller statistics. It is found from this distribution that the median and 80% (equal-tailed) credible interval of μ_t is 2.8 hr and (1.9, 7.3) hr for Device 1 and 1.1 hr and (0.9, 1.6) hr for Device 2. Experienced humans require approximately 3 hr to tune a device of similar characteristics into exhibiting double quantum dot features [83], demonstrating the approach is competitive. Due to device variability, the hypersurfaces of these two

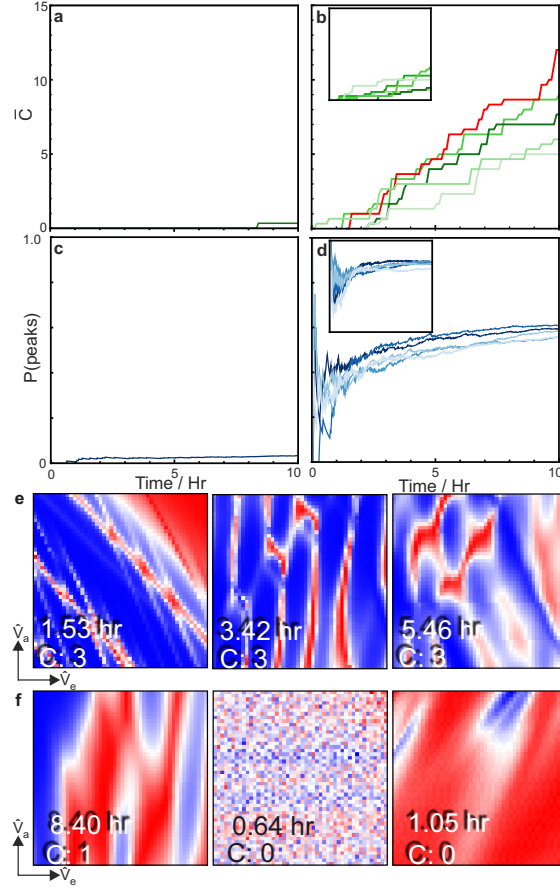


Figure 4.10: Algorithm's performance a-d Average number of current maps displaying double quantum dot features, $\bar{C}$, and $P(\text{peaks})$ as a function of laboratory time. Current maps are labelled by humans a posteriori, i.e., after the algorithm is stopped. **a, c**, correspond to one run of *Pure random* while **b, d** correspond to five runs of the full algorithm. All algorithm runs displayed in main panels were performed in Device 2, while insets show runs of the algorithm in Device 1. **e, f** High-resolution current maps produced in Device 2 by *Pure random* and by one of the full algorithm runs, respectively. I indicate the time the algorithm had been running for before they were acquired and the number of labellers, C , that identified them as displaying double quantum dot features. Current maps are ordered from left to right in decreasing order of C , and maps that have the same values of C are displayed in the order at which they were sampled. Each panel uses an independent colour scale running from red (highest current measured) to blue (lowest current). All measured high resolution stability diagrams for all tuning runs are displayed in Appendix .9.

devices are significantly different, showing our algorithm is capable of dealing with such variability.

Fig. 4.10 c,d, compares the probability of measuring Coulomb peaks in the vicinity of a given $s(\mathbf{u})$, $P(\text{peaks})$, for *Pure random* and for different runs of the full algorithm. This is calculated by taking the empirical probability, i.e. the number of observed Coulomb peaks at iteration i divided by i . $P(\text{peaks})$ is significantly increased by the full algorithm. This increase is observed both during and immediately after the investigation stage, followed by saturation.

Figure 4.10 e,f shows the high-resolution current maps obtained in Device 2 by *Pure random* and one of the full algorithm runs. It is observed that, within 1.53 hr, the algorithm produces high-resolution current maps that are recognized by all labellers as displaying double quantum dot features. The three current maps in Fig. 4.10 f correspond to double quantum dot regimes found by the algorithm in different regions of the gate voltage space. The number of labellers C who identified the current maps produced by *Pure random* as corresponding to the double quantum dot regime, C , is 0 or 1.

4.9.5 Tuning by grouping gates

To significantly reduce tuning times, I modified the algorithm to group gate electrodes that perform similar functions. The algorithm assigns equal gate voltages to gate electrodes in the same group. For Device 1, the eight gate electrodes were organised into four groups: $G_1 = (V_1)$, $G_2 = (V_2, V_8)$, $G_3 = (V_3, V_7)$, and $G_4 = (V_4, V_5, V_6)$. In this case, the median and 80% credible interval of μ_t improve to 0.6 hr and (0.4, 1.1) hr. This approach demonstrates that exploiting knowledge of the device architecture reduces μ_t dramatically, in this case by more than four times.

4.9.6 Ablation study

The proposed algorithm consists of many components which contribute differently to the performance of the algorithm. I attempt to verify the components of the algorithm through an ablation study, in which various modules of the algorithm

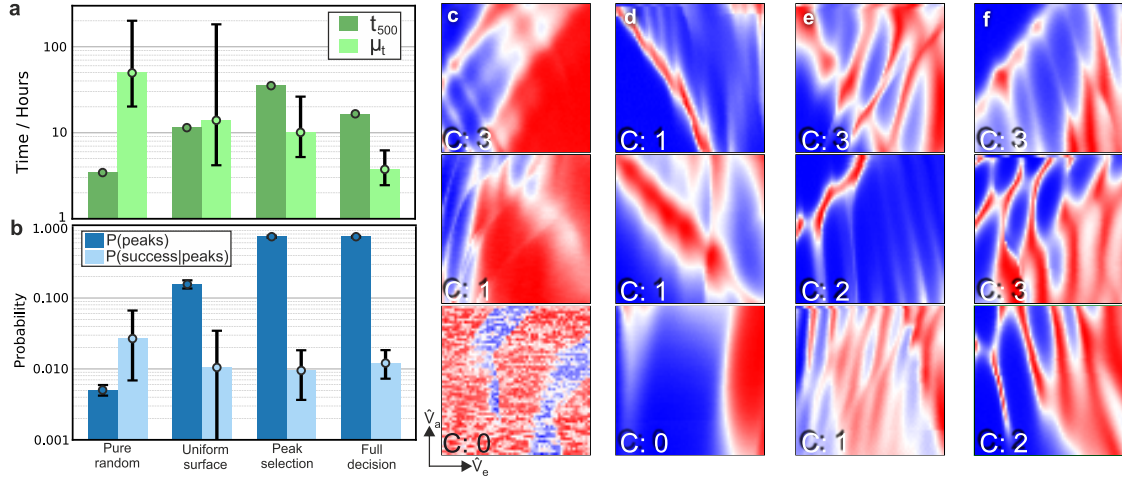


Figure 4.11: Ablation study **a, b** Bar charts and corresponding data points comparing μ_t (light green), t_{500} (dark green), $P(\text{peaks})$ (dark blue) and $P(\text{success}|\text{peaks})$ (light blue) for the different algorithms considered. Error bars represent 80% (equal-tailed) credible intervals. Due to a measurement problem, 459 sampling iterations instead of 500 were considered for the *Full decision* algorithm. **c-f** High-resolution current maps sampled by *pure random*, *uniform surface*, *peak weighting*, and *full decision*, respectively. In each panel, I indicate C , the number of human labellers that identified a map as displaying double quantum dot features. Current maps with identical values of C are displayed in the order in which they were sampled, from top to bottom. Current maps with $C: 0$ were randomly selected. Each panel uses an independent colour scale running from red (highest current measured) to blue (lowest current).

are removed. For the ablation study I chose to compare the full algorithm which I refer to as *Full decision*, with three reduced versions that combine different components. These version are; *Pure random*, *Uniform surface* and *Peak weighting*. The *Pure random* algorithm is defined in section 4.9.3. *Uniform surface* makes use only of the hypersurface sampling module of the sampler (section 4.6.2). No weighting is considered and the pruning heuristic is not applied (section 4.7). The investigation stage uses the search for the hypersurface. *Peak weighting* uses the full sampler and applies the pruning heuristic. All algorithms use the Coulomb peak check (section 4.5.2) in their investigation stage, and none use the score function threshold (section 4.5.3). During the ablation study, the low (high) resolution scans contained 20×20 (60×60) pixels.

To analyse the algorithms' performance, I estimate $P(\text{peaks})$ and the probability of success, i.e. the probability of acquiring a high-resolution current map labelled as containing double quantum dot features, given Coulomb peak measurements

$P(\text{success}|\text{peaks})$. To take measurement times into consideration, I define t_{500} as the time to sample and investigate 500 locations in gate voltage space. The ablation study was performed in Device 1 keeping the investigation stage parameters fixed. The cool down cycle was the same as in section 4.9.4. Device tuning was performed in a given cool down (cool down 1), except for *Pure random*, which was run in a different thermal cycle (cool down 2). Results are displayed in Fig. 4.11.

Figure 4.11 a shows that the introduction of hypersurface sampling, weighting and pruning increases t_{500} . This is because $P(\text{peaks})$ increases with the inclusion of these components (Fig. 4.11b), and thus the number of low and high resolution current maps acquired by the investigation stage is larger. Within uncertainty, $P(\text{success}|\text{peaks})$ remains mostly constant for the different algorithms considered. The result is a decreasing μ_t from *Pure random* to *Peak weighting* within experimental uncertainty.

The reason behind the use of Coulomb peak detection in all the algorithms considered for this ablation study is the vast amount of measurement time that would have been required otherwise. Without Coulomb peak detection, the posterior median estimate of μ_t for *Pure random* is 680 hr.

To complete the ablation study, the grouped gates approach described in the previous section was considered, keeping parameters such as the current map resolutions equal. It was found that $\mu_t = 80.5$ mins.

Comparing *Pure random* and *Uniform surface*, the importance of hypersurface sampling is clear. The difference between *Uniform surface* and *Peak weighting* highlights the importance of weighting/Thompson sampling, and pruning. The improved performance of *Full decision* with respect to *Peak weighting* evidences the tuning speedup achieved by the introduction of the score function. These results demonstrate *Full decision* exhibits the shortest μ_t and imply an improvement over *Pure random* without Coulomb peak detection of approximately 180 times.

4.10 Discussion

I investigated the gate voltage parameter space of electrostatically-defined quantum dot devices. I proposed a statistical search algorithm that uses a small amount of device knowledge to make efficient decisions. The sampling algorithm approaches tuning as a multi armed bandit problem using Thompson sampling to manage the exploration vs exploitation trade-off. All the algorithm modules either serve to increase the probability of observing a double quantum dot, or reduce the time cost associated with evaluating the transport regime of the device.

The proposed algorithm was demonstrated experimentally and displays performances superior to that of a typical human operator with median tuning times of under 70 minutes. This was further improved to 36 minutes when gates are grouped to reduce the dimensionality of the search space. The algorithm's design was analysed using an ablation study and found to be justified. A 179 times improvement was shown over the naive approach; pure random search without a Coulomb peak check. This demonstrates that knowledge about the device behavior can have a large impacts on the algorithm's performance.

The approach in its current form is limited to making measurements using transport through the device. In order to readout spin states and accurately count charge states alternate measurement methods are required such as charge sensing. Therefore, extending this work to include the tuneup of a charge sensor would be valuable. This could be achieved by extending the investigation stage with a charge sensor tuneup step after the high or low resolution scan completes or by simultaneously tuning the charge sensor while investigation is being performed.

5

Automatic coarse tuning of silicon devices

In this chapter I experimentally demonstrate that the algorithm proposed in Chapter 4 can be generalised to other quantum dot devices with minimal changes. These include devices with different host materials, device architectures, and charge carriers. The results in this chapter further emphasise the efficacy of the automated tuning approach I developed and demonstrates that general solutions to the problem of device tuning can be developed.

5.1 Devices

To demonstrate the generality of the approach developed in Chapter 4, the algorithm was run in three quantum dot devices of entirely different design. Throughout this chapter the devices are referred to as: FinFET, Nanowire, and Heterostructure.

5.1.1 The FinFET

The FinFET (Fig. 5.1) is a non-planar Si device, capable of am-bipolar operation developed using industry standard equipment and processes [15, 88, 89]. In this work I operate the device in a regime in which holes are dominant. The device consists of a Si channel, which is raised and makes contact with a source and drain at either end. As the charge carries (holes in this case) are already restricted to a

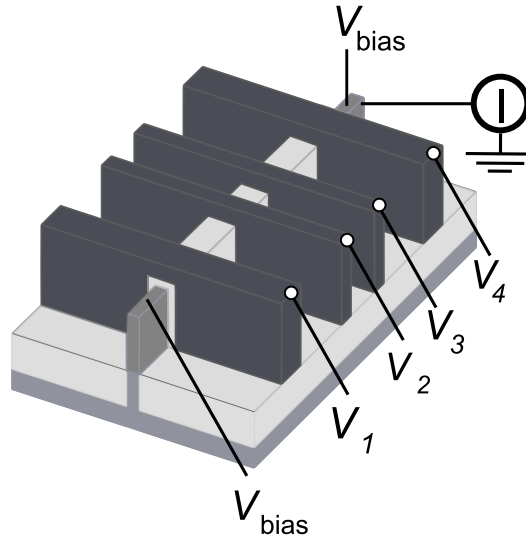


Figure 5.1: Schematic of the FinFET. Schematic of the device. A bias voltage V_{bias} is applied between the ohmic contacts to the channel. Voltages labelled V_1 to V_4 are applied to the gate electrodes.

channel, this device is quasi-1D. On top of the Si channel, 4 gate electrodes made of titanium nitride are patterned such that they wrap over the channel. An oxide is grown between the channel and the gate electrodes to prevent leakage. Gate voltages V_1 , V_2 , V_3 , and V_4 are applied. The channel is non-conducting at zero gate voltage and negative gate voltages must be applied in order to accumulate holes and allow for an observable transport current I to flow. The roles of gates V_1 and V_4 are to accumulate charge carriers in different regions of the channel. Gates V_2 and V_3 aim to control the electrochemical potential of the quantum dots formed beneath them.

5.1.2 The Nanowire

The Nanowire (Fig. 5.2) is a germanium silicon core shell nanowire device in which holes are the main charge carriers [10, 90]. The nanowire is placed over 5 gate electrodes and both ends of the nanowire make contact with a source or drain electrode, allowing a bias voltage V_{bias} to be applied. Similarly to the FinFET, holes are confined to a channel, so this device is quasi-1D. The 5 gate electrodes are individually controllable using gate voltages V_1 , V_2 , V_3 , V_4 , and V_5 . With no gate voltages applied, the nanowire is conducting and so charges must be depleted

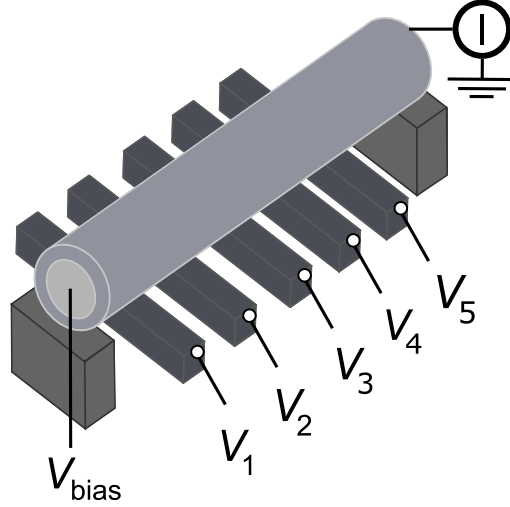


Figure 5.2: Schematic of the Nanowire. Schematic of the device. A bias voltage V_{bias} is applied between the source and drain contacts of the nanowire. Gate voltages labelled V_1 to V_5 are applied to the gate electrodes.

by applying positive gate voltages. The roles of gate electrodes V_1 , V_3 , and V_5 is to form tunnel barriers in the channel defining the double dot confinement potential. Gates V_2 and V_4 aim to control the electrochemical potential of the left and right quantum dots.

5.1.3 The Heterostructure

The Heterostructure (Fig. 5.3) is a device laterally-defined in a silicon germanium heterostructure [91] for which holes are the main charge carriers. The device consists of a strained Ge quantum well that forms a two-dimensional hole gas (2DHG). On top of the heterostructure, gate electrodes are patterned. Ohmic contacts to the 2DHG allow a bias voltage V_{bias} to be applied. The device is quasi-2D and has 7 gate electrodes that are relevant to the creation of a double quantum dot confinement potential. Gate voltages are labelled V_1 , V_2 , V_3 , V_4 , V_5 , V_6 , and V_7 . With no gate voltages applied, the heterostructure is conducting and charge carriers must be depleted by applying positive gate voltages. The role of gate electrodes V_1 and V_7 is to define a path for current to flow. Gate electrodes V_2 , V_4 and V_6 aim to form and control tunnel barriers, while gates V_3 and V_5 aim to control the

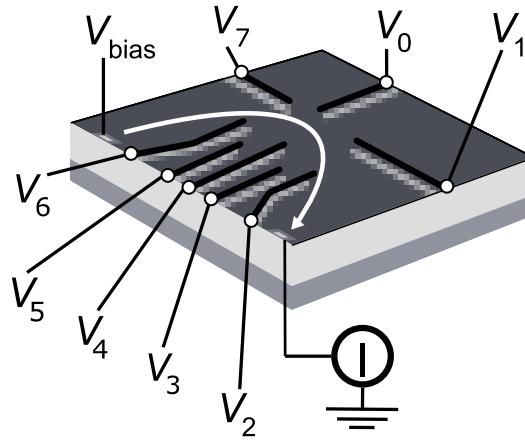


Figure 5.3: Schematic of the heterostructure. Schematic of the device. A bias voltage V_{bias} is applied between ohmic contacts to the 2DHG. Gate voltages labelled V_1 to V_7 are applied to the gate electrodes.

electrochemical potential of the left and right quantum dots.

5.2 Generalising the approach to different devices

To generalise the automatic tuning approach small modifications were made to the algorithm presented in Chapter 4.

5.2.1 Sampling modifications

The devices presented in section 5.1 differ significantly. The gate voltages that must be applied to achieve intended operation change significantly from device to device. However, the modeling assumptions discussed in section 4.4 still hold; we still expect the devices to behave like high-dimensional transistors (i.e. there is an hypersurface), and we still expect that large regions of the hypersurface do not display transport features.

Applying the definition of the hypersurface described in section 4.3 is problematic for the different devices considered. The FinFET is none conducting when no gate voltages are applied, so the hypersurface is undefined. The Nanowire and the Heterostructure are both conducting when no gate voltages are applied, but

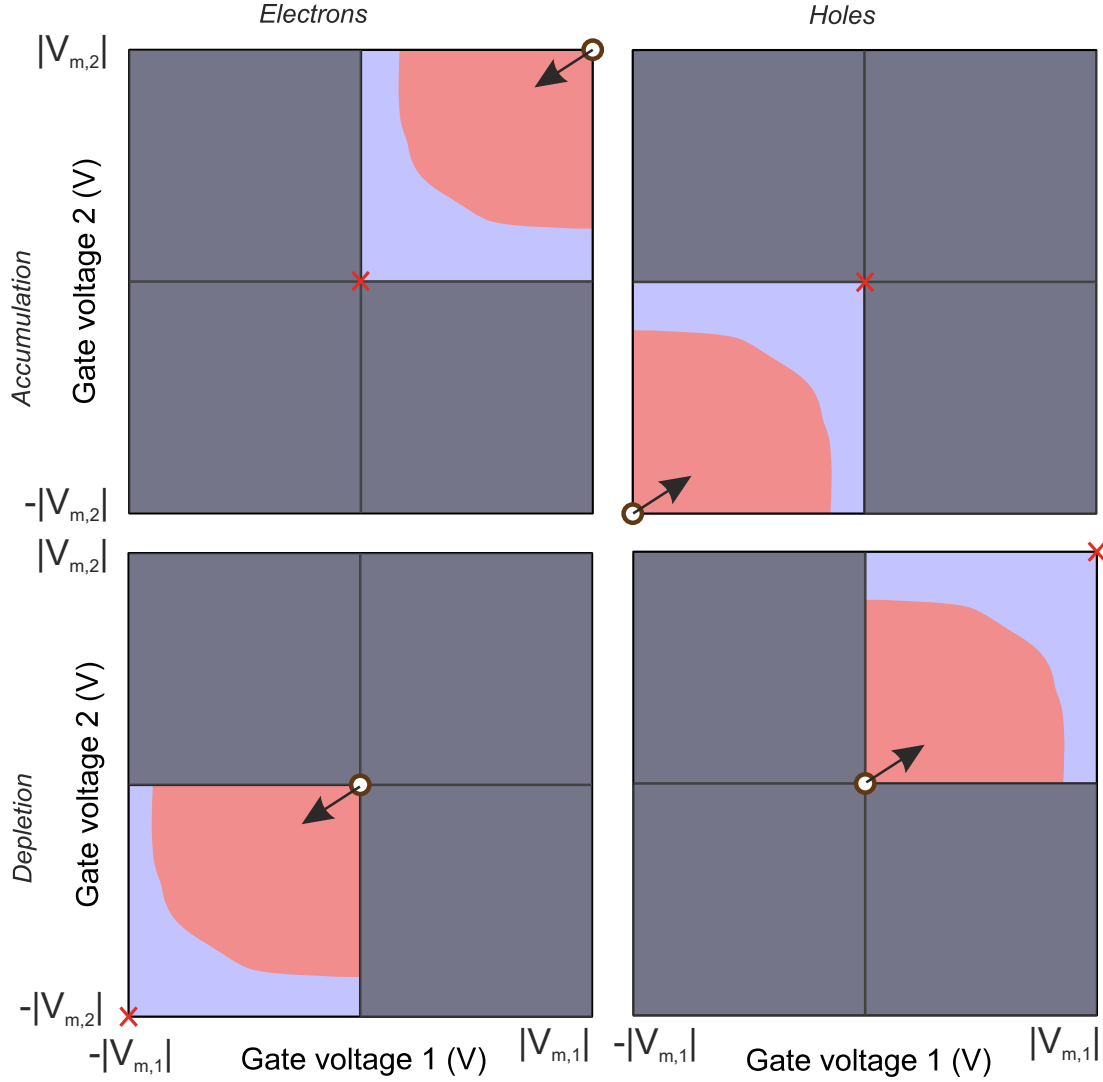


Figure 5.4: Generalising the hypersurface search Current maps corresponding to devices with different charge carriers are displayed in different columns. Current maps corresponding to devices with different gate operation modes are represented in different rows. Each panel illustrates the initial placement of the origin $\circ$ (white circle), search boundary V_m (red cross), and search direction $\mathbf{u}$ (black arrow). The gate voltage space is divided into regions of near-zero (blue) and non-zero (pink) current. Regions of voltage space which cannot be explored due to the gate voltage bounds set to avoid device damage are greyed out.

the charge carriers are holes and thus the hypersurface is not defined in the negative quadrant of $\mathbf{u}$.

To generalise the hypersurface search, I altered the direction of the search so that $\mathbf{u} = |\mathbf{u}_{\text{GaAs}}|$ where $\mathbf{u}_{\text{GaAs}}$ is the negative quadrant of gate voltage space considered throughout chapter 4. The safe gate voltage range must also be adjusted.

Device	$\mathbf{o}$ (V)	V_m (V)
The FinFET	[-6.5, -1.5, -1.5, -6.5]	[0, 0, 0, 0]
The Nanowire	[0, 0, 0, 0]	[4.0, 2.5, 2.5, 4.0, 4.0]
The Heterostructure	[0, 0, 0, 0, 0, 0, 0]	[2, 2, 2, 2, 2, 2, 2]

Table 5.1: Experimental parameters. Values of $\mathbf{o}$ and V_m used for the tuning runs described in this chapter.

The variable previously referred to as $V_{\min}$ will now be referred to as V_m to avoid confusion. By allowing $V_m > 0$ I can adapt the approach to the gate voltage space corresponding to the Nanowire and the Heterostructure. However, the FinFET's hypersurface remains undefined. To resolve this, the location of the origin $\mathbf{o}$ and extremum bound V_m are swapped. Figure 5.4 displays the changes to the origin, search direction, and extremum bound for different charge carriers and gate operation modes (accumulation or depletion). Chapter 4 demonstrates the success of the tuning algorithm for electrons in depletion mode, Chapter 6 section 6.2.2 demonstrates the success of the tuning algorithm for electrons in accumulation mode, and the current chapter deals with holes in both modes. Values of $\mathbf{o}$ and V_m used for the experiments described in this chapter are contained in table 5.1. These values were selected by asking experts who regularly operate these samples.

5.2.2 Other modifications

To generalise the tuning algorithm other changes were made. The threshold defining the search for the hypersurface was changed to be 0.5% of the maximum current range. Also, during the Coulomb peak check (section 4.5.2), a different method is used for peak detection. The *find_peaks* method miss-classifies charge switches as Coulomb peaks. The lack of charge switches in the experiments described in chapter 4 did not allow me to identify this problem. To circumvent it, I trained a Random Forrest classifier on saved data (see appendix .13). This classifier is applied to the 128 measurements performed for the Coulomb peak check and its output is used to decide whether to perform a low resolution scan and to

Device	CPC (pixels), (mV)	Orientation
The FinFET	128, 100	$\frac{1}{\sqrt{2}}(\hat{V}_2^2 + \hat{V}_3^2)$
The Nanowire	128, 200	$\frac{1}{\sqrt{2}}(\hat{V}_2^2 + \hat{V}_4^2)$
The Heterostructure	128, 100	$\frac{1}{\sqrt{2}}(\hat{V}_3^2 + \hat{V}_5^2)$

Table 5.2: Coulomb peak grid scan parameters. Gate voltage ranges used for coulomb peak grid scan in the coulomb peak check (CPC), and selected gate orientation for the different devices considered in this chapter.

Device	LRS (pixels), (mV)	HRS (pixels), (mV)	Orientation
The FinFET	16x16, 80x80	48x48, 120x120	$\hat{V}_2, \hat{V}_3$
The Nanowire	16x16, 150x150	48x48, 200x200	$\hat{V}_2, \hat{V}_4$
The Heterostructure	16x16, 80x80	48x48, 120x120	$\hat{V}_3, \hat{V}_5$

Table 5.3: Low and high resolution scan parameters. Gate voltage ranges used for low resolution scans (LRS) and high resolution scans (HRS), and selected gate orientation for the different devices considered in this chapter.

train the sampler. The parameters used to measure during the coulomb peak check are displayed in table 5.2.

I did not use the adaptive window resizing approach detailed in section 4.5.3 and I did not use the axes $\hat{V}_e$ and $\hat{V}_a$ to orient the grid scans as these were found to add little improvement to the algorithm. The size and resolution of low and high resolution scans are in this case selected by the user before running the algorithm. The high resolution scan is performed in a larger gate voltage range than the low resolution scan, in order to provide additional information about promising regimes. All parameters defining the grid scans are detailed in Table 5.3.

During sampling, the initialisation was applied for 12 iterations instead of 30 (see section 4.6.2). The Pruning heuristic (section 4.7), is not applied because it is only applicable in devices for which a current path must first be defined. This excludes the quasi-1D devices. The method would be applicable to the Heterostructure but is excluded for consistency.

5.3 Tuning Si and SiGe-based devices and analysis of expected tuning times

The algorithm (*Full decision*) was run multiple times for all the devices considered in this chapter in order to assess the expected tuning times μ_t . The labelling procedure (Section 4.9.1) and multi-labeller statistics (section 4.9.2) were carried out with four independent labellers. The algorithm was bench-marked against *Pure random* (see section 4.9.3). The *Pure random* algorithm was applied over safe gate voltage ranges. The modifications made to the algorithm’s investigation stage described in this chapter were also applied to the *Pure random* algorithm.

For each device, the algorithm was run for 250 iterations. It was run at least 12 times for each device. All high resolution scans were compiled into one shuffled data set for labelling. High resolution scans along with their human labels, high resolution score and time of acquisition are all displayed in appendix .11.

The number of positively labelled stability diagrams averaged across labellers, $\bar{C}$, is displayed as a function of time for each device and algorithm run in Fig. 5.5. For all *full decision* runs I observe that the *Pure random* counterpart performs worse. The estimated tuning times μ_t can be found in Table 5.4.

Device	Tuning Times (Hr)	
	Full Decision	Pure Random
Si FinFET	0.50 (0.43, 0.62)	N/A
GeSi Nanowire	0.16 (0.11, 0.20)	0.28 (0.17, 0.43)
SiGe Het.	1.5 (1.2, 2.0)	5.8 (3.0, 13.2)

Table 5.4: Median device tuning time μ_t , with 80% credible intervals (equal tailed) for the *full decision* and *pure random* algorithm runs described in this chapter. The *Pure random* tuning time could not be estimated for the FinFET due to a lack of scans labelled as displaying double quantum dot features in the time allocated for the experiment.

For the FinFET, the improvement in performance over the *Pure random* baseline is significant. *Pure random* was unable to sample a positively labelled high resolution scan across 12 runs (totalling 3000 iterations). In this instance, the speedup is likely attributed to the hypersurface search part of the algorithm. Table 5.1 shows that gates V_1 and V_4 have a comparatively large safe range in the case of the

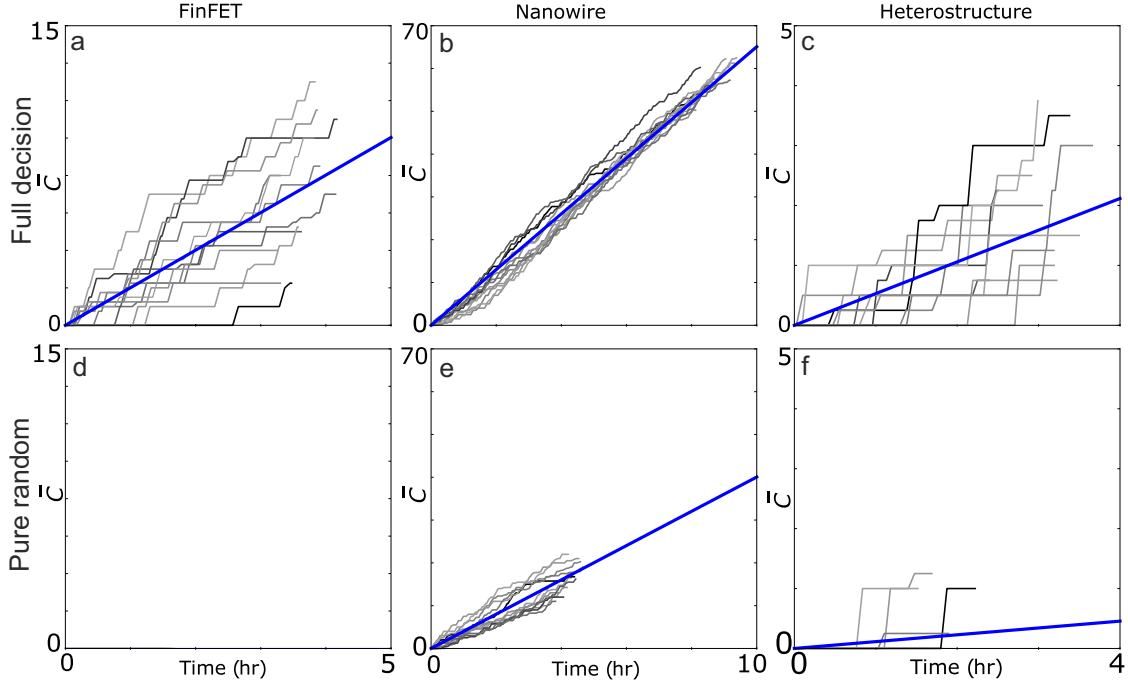


Figure 5.5: Tuning Si and SiGe-based devices. **a-f** Average number of current maps displaying double quantum dot features, $\bar{C}$, as a function of laboratory time. Current maps are labelled by humans a posteriori, i.e., after the algorithm is stopped. Solid blue lines show the average tuning rate. **a**, **b**, and **c**, correspond to at least 12 runs of the algorithm. **d**, **e**, and **f** correspond to at least 12 runs of the *Pure random* implementation. All measured high resolution stability diagrams for all full decision tuning runs are displayed in Appendix .11.

FinFET. In most of the gate voltage space, current transport is not possible. In the gate voltage space for which holes are accumulated, the device displays a small number of double dot charge transitions (in the order of order 3 per dot). By reparameterising the search across the hypersurface, *Full decision* overcomes the difficulty presented by such a flat optimisation landscape.

For the Nanowire, the improvement in algorithm’s performance over Pure random is less significant. The *Full decision* method collected as many as 60 positively labelled current maps per run. The algorithm spends the majority of the time acquiring high resolution scans. This is evidenced by the algorithm runs approaching the 10 hour mark. When the device is in a double dot regime, in the order of 30 charge transitions can be observed. This might be the case because the gate electrodes in the Nanowire define almost diagonal capacitance matrices, and the device operates in the many hole regime, never emptying the quantum dots.

The Hetrostructures sees a modest improvement in algorithm's performance over Pure random. The highest expected tuning times are obtained in this devices. The plot in Fig. 5.5c closely resembles that of Fig. 4.10b, (considering that in Chapter 4 500 iterations were performed per run), indicating quasi-2D systems are intrinsically harder to tune - likely due to their higher voltage space dimensionality. The inclusion of the pruning heuristic would have improved the algorithm's performance.

5.4 Analysis of score function for classification

The score function presented in section 4.5.3 can be applied directly to high resolution stability diagrams to determine whether the algorithm should halt. To demonstrate this capability I apply the score function to all high resolution stability diagrams measured in chapter 4 and this chapter. The goal of the score function is to classify if a given stability diagram is indicative of a double quantum regime. Classification performance is evaluated by comparing score function classification to the human expert labels used to calculate μ_t . There is significant class distribution imbalance that varies from device to device, making robust evaluation of performance challenging. I quantify the score functions performance as a classifier by analysing its Receiver operating characteristic (ROC) and area under curve (AUC) for different devices.

In Figure 5.6 it can be seen that the score function is capable of classifying whether a high resolution stability diagram is indicative of a double quantum dot regime. The AUC values for each device are: 0.83 for the GaAs devices, 0.77 for the finFET devices, 0.77 for the SiGe heterostructure devices, and 0.70 for the nanowire devices. The performance of the score across devices is relatively stable considering their differences, however it is clear the different components of the score function behave quite differently. The sharpness score has the largest variability and even harms performance in the finFET device. The orientation of gradient score provides the most consistent performance.

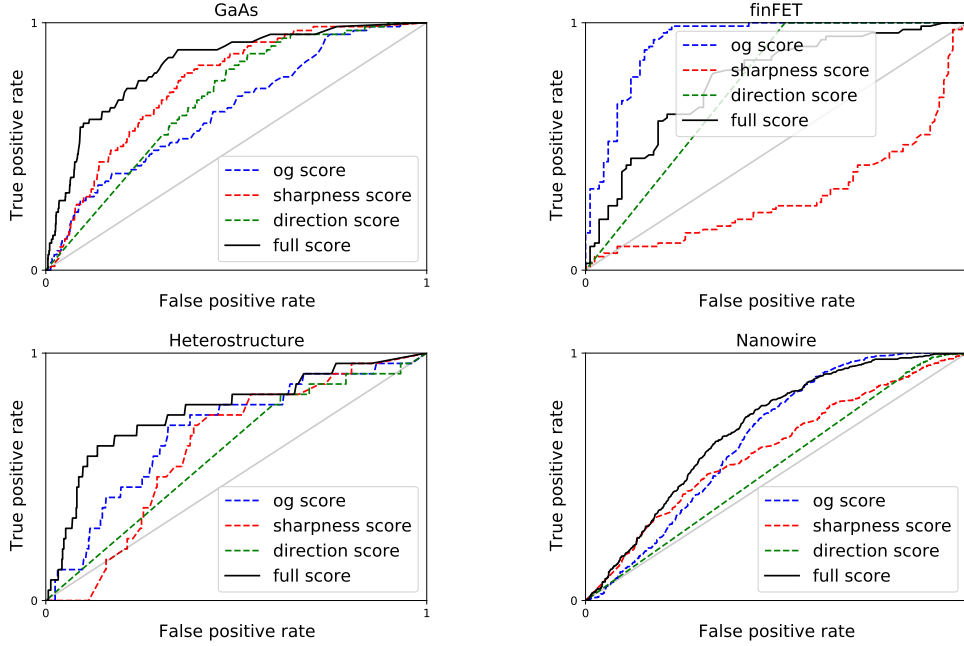


Figure 5.6: ROC curves for different devices. ROC curves of score function (solid black) as a classifier on all high resolution scans measured in chapters 4 and 5 separated according to device type. All components of the score function are also plotted.

From this analysis it is clear that using the score function as a classifier is an effective way of detecting promising regimes but is also likely to produce false positives. This can be mitigated by allowing the algorithm to collect multiple positives and allowing a human operator to select from a small set of high resolution stability diagrams. Alternatively, other data driven approaches have been demonstrated for the classification of quantum dot regimes [36] however, training data-sets are small and device specific so there is a significant risk of over-fitting.

5.5 Discussion

I demonstrate that, with a few modifications, the algorithm introduced in Chapter 4 can be used for different material systems and device architectures. The modifications required are minimal; only the search directions, search bounds, the origin of the algorithm search and the Coulomb peak detector had to be revised.

The algorithm tuned all devices it was presented with, achieving an expected tuning time of under 10 minutes for the Nanowire device. A difference in the tuning

times of quasi-1D and quasi-2D systems is evident, indicating that quasi-2D systems are more difficult to tune. I hypothesised that is due the quasi-2D systems in this work have more gate electrodes than the quasi-1D systems, however more research will be required to verify this hypothesis. This is the first demonstration of such a general tuning algorithm. Determining when to stop is a significant challenge and is a weakness of the current method. Future work should focus on designing more robust features or training reliable models for classification of quantum dot regimes.

In Chapter 6 I build on the algorithm's framework to demonstrate its ability to explore a different device characteristic, that of variability.

6

Characterising device variability

In this chapter, I describe how the algorithm presented in chapter 4 can be used for a different purpose. The focus is to study device variability and the core idea is to use a uniform search approach to probe regions of interest in gate voltage space. Post-processing techniques are applied to the outputs of these uniform search algorithms providing new metrics of device variability.

6.1 Variability via point-set registration

The methods introduced in Chapter 4 can be repurposed to produce an approximately uniformly distributed set of observations of the hypersurface. These observations are a set of locations in gate voltage space. I will refer to these locations as a point set. By running the same algorithm again, or by running it on a different device, a second point set can be generated. The transformation that maps one point set to the other can be learnt through point set registration. An insight of the mechanisms that lead to device variability can be explored from such transformations.

6.1.1 Devices

The devices used for the experiments presented in this chapter are Device 1 and Device 2, described in section 4.1 and displayed in Fig. 4.1. Gate electrodes V_1 to V_8 are used to control the transport regime, and V_{bias} is applied to drive a current I through the device.

6.1.2 Sampling the hypersurface

Generating a point set involves sampling the hypersurface of the experimental device uniformly. The hypersurface being measured is defined in section 4.3. The procedure used to generate a point set $\mathbf{v}_a$ is as follows. The sampler runs the initialisation stage for 30 iterations. During the initialisation state, the hypersurface search begins at the origin $\mathbf{o}$ and unit vectors $\mathbf{u}$ are chosen uniformly on a unit hypersphere. The search method detailed in section 4.5.1 is used to find the value of r for which the current threshold condition is satisfied. Throughout this section, the value used for I_{thresh} is the same as in chapter 4; 20% of the maximum current (current at the origin). In every iteration, $s(\mathbf{u}) = r\mathbf{u} + \mathbf{o}$ is found and appended to $\mathbf{v}_a$.

After the sampler exits the initialisation state, a Gaussian process is trained in every following iteration to model $\bar{p}(r|\mathbf{u})$. The sampler uses this model to select a unit vector $\mathbf{u}$ using the Brownian motion approach detailed in section 4.6.2. The posterior of the Gaussian process is $\bar{p}(r|\mathbf{u}) = N(m(\mathbf{u}), \sigma(\mathbf{u})^2)$. The linear search is started at $\mathbf{o}_{\text{adj}} = (m(\mathbf{u}) - 2\sigma(\mathbf{u}))\mathbf{u} + \mathbf{o}$. In every iteration, $s(\mathbf{u})$ is found and appended to $\mathbf{v}_a$.

Generation of the second point set $\mathbf{v}_b$ can either be achieved by running the algorithm again or repeating the set of unit vectors $X_{1:N} = \{\mathbf{u}_j\}_{j=1,\dots,N}$ that have been observed in a previous point set run. In this chapter, I present results produced with the second method.

6.1.3 Point set registration

Affine transformations have proven useful to find adequate combinations of gate voltages which simplify device tuning [41, 43]. To find a measure of device variability, understood as changes occurring to a device's hypersurface, I use an affine

transformation $\mathbf{v}_t = B\mathbf{v}_b + t$, with B a matrix which contains the transformation parameters. I look for a transformation of coordinates that converts $\mathbf{v}_b$ into a set of gate voltage locations $\mathbf{v}_t$ that is as similar to $\mathbf{v}_a$ as possible.

Coherent point drift [92] was chosen for point set registration, since it can be used for point sets of high dimensionality. It is simple to implement, robust to noise in the point sets, and supports affine transformations as well as rigid and non-rigid transformations. As mentioned above, I focused on affine transformations of the type $T(x) = Bx + t$, where B is a square matrix, and t is a column vector. Reference [92] presents the mathematical derivation for this case. In this application t does not have a physical interpretation, and so it was set to 0. A derivation for the case $t = 0$ was required. This was achieved by setting the centered point matrices to $\hat{X} = X$ and $\hat{Y} = Y$ in the derivation detailed in Ref. [92].

This approach was used to quantify the variability between Devices 1 and 2, and the effects of a thermal cycle in the hypersurface of Device 1 and 2. For these two cases, Fig. 6.1 and Fig. 6.2 display the matrix $B_c = B - I$ that quantifies how much B , the transformation that transforms a set of locations from one hypersurface onto the other, differs from the identity matrix (I). Non-zero elements of B_c indicate device variability. Diagonal elements of B_c are responsible for scale transformations and can be interpreted as a capacitance change for a given gate electrode. Off-diagonal elements are responsible for shear transformations and can be interpreted as a change in cross-capacitance between pairs of gate electrodes.

Figure 6.1a shows the matrix B_c obtained after a thermal cycle of Device 1. The transformation shows that the effect of a thermal cycle on the device hypersurface is dominated by a uniform change in capacitance for all gate electrodes. The matrix B_c illustrating the effect of a thermal cycle of Device 2 is displayed in Fig. 6.1b and it is similar to that obtained for Device 1. Figure 6.2 displays the matrix B_c corresponding to the mapping of the hypersurface of Device 2 onto the hypersurface of Device 1. I observe that the variability between these devices, which share a similar gate architecture, is given by non-uniform changes in gate electrode capacitance, as well as by changes in cross-capacitance. This big variability

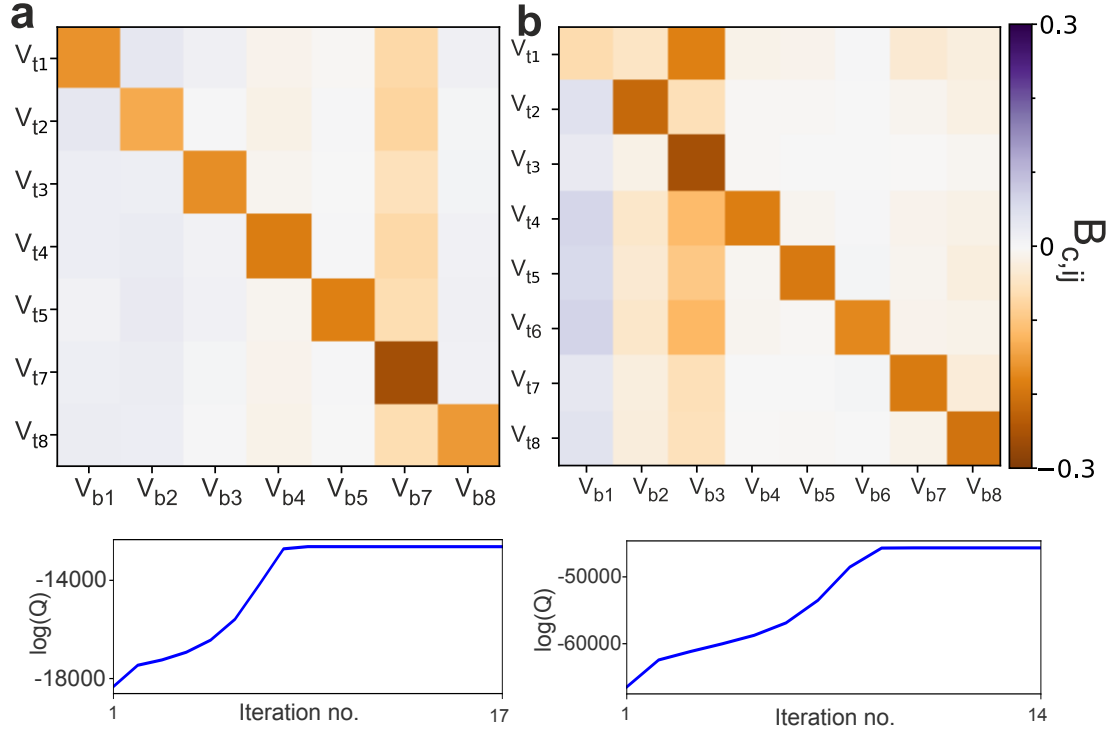


Figure 6.1: Learning about device variability after a thermal cycle. B_c matrices obtained using point set registration. Indices are the gate voltage represent gate voltages for different point sets $\mathbf{v}_b$ and $\mathbf{v}_t$. $V_6 = 0$ mV was fixed in Device 1 to prevent leakage currents. Q function values are plotted as a number of iterations below their respective transformation (see Myronenko and Song [2010] equation 10 for definition). **a** Transformation of the hypersurface of Device 1 after a thermal cycle. **b** Transformation of the hypersurface of Device 2 after a thermal cycle.

is attributed to charge traps and other device defects, such as inaccuracies in the patterning of gate electrodes.

The hypersurface provides information about the tunnel barriers that can be defined in the device. This method tries to explain their variability by considering simple linear transformations, and in this way, shares similarity with the virtual barrier gate approach detailed in section 1.8.3. By registering a point set generated by an hypercube onto a real point set the resulting transformation can be used to estimate virtual barrier gates.

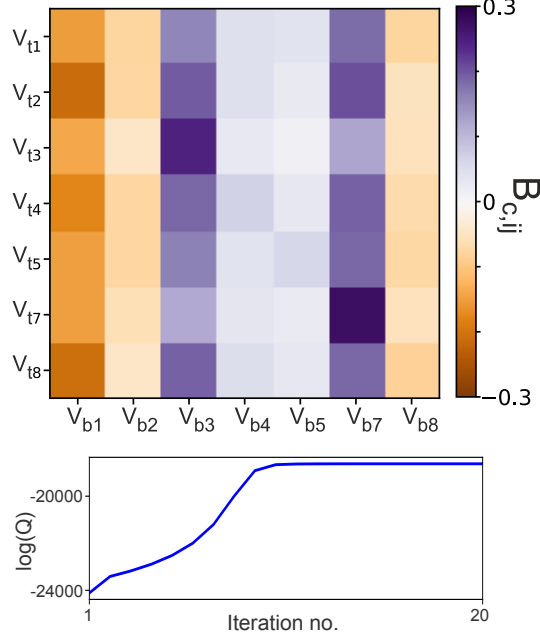


Figure 6.2: Learning about device variability. B_c matrix evidencing the transformation between hypersurfaces of Device 1 and Device 2. Q function values are plotted as a number of iterations below the transformation (see Myronenko and Song [2010] equation 10 for definition).

6.2 Variability via operable region sampling and characterisation

In this section, I build upon the concept of device characterisation via uniformly sampling regions of the parameter space. I aim to draw samples from all locations in device parameter space in which the device is operable.

6.2.1 Device

The device used for this section is designed to define a single quantum dot and is based on CMOS-compatible fabrication techniques [93]. It consists of a channel of planar silicon that is connected to source and drain ohmic contacts. By applying a DC voltage bias to the source ohmic contact V_{bias} , a current I flows through the device. Over the Si channel, three gate electrodes are lithographically defined. They can be controlled with DC voltages V_{LB} , V_{RB} , and V_{ST} . These gate voltages can create an electrostatic potential capable of confining electrons by accumulation.

Gate electrodes V_{LB} and V_{RB} are designed to control the strength of the tunnel barriers. V_{ST} controls the electrochemical potential of the dot.

6.2.2 SET operation and sensitivity

These devices, which act as single-electron transistors (SETs), are often used as charge sensors since they are able to detect small rearrangements in the local charge distribution. DC voltages are applied to the gates of the SET and a regime of high transconductance is found. This regime is commonly found in the fringes of a Coulomb peak. A change in the local distribution of charges alters the electrostatic environment of the SET producing a large change in the transport current. The key metric for charge sensor performance is its sensitivity [94].

$$\delta q = \frac{\sqrt{S_I}}{\partial I_Q}, \quad (6.1)$$

where $\partial I_Q \equiv \frac{\partial I}{\partial Q}$ and

$$\partial I_Q = \frac{\Delta V_g}{e} dI_{V_{ST}}. \quad (6.2)$$

The device transconductance is $dI_{V_{ST}} \equiv \frac{dI}{dV_{ST}}$ and ΔV_g is the spacing between Coulomb peaks. S_I is the power spectral density that accounts for noise. I make the assumption that S_I is constant which is justified if charge noise is a property of the environment of the device and not the specific gate voltage configuration, hence

$$\delta q \propto g_m = \frac{1}{\Delta V_g dI_{V_{ST}}}. \quad (6.3)$$

I refer to g_m as the sensitivity score and report it in units of nA^{-1} . A lower sensitivity score is better.

6.2.3 The operable region of gate voltage space and the algorithm's goal

I consider the ‘operable region’ as the region in gate voltage space in which a device can be used for its intended purpose. For a charge sensor to be rendered operable, it must be sensitive enough to charge changes. The operability condition I consider

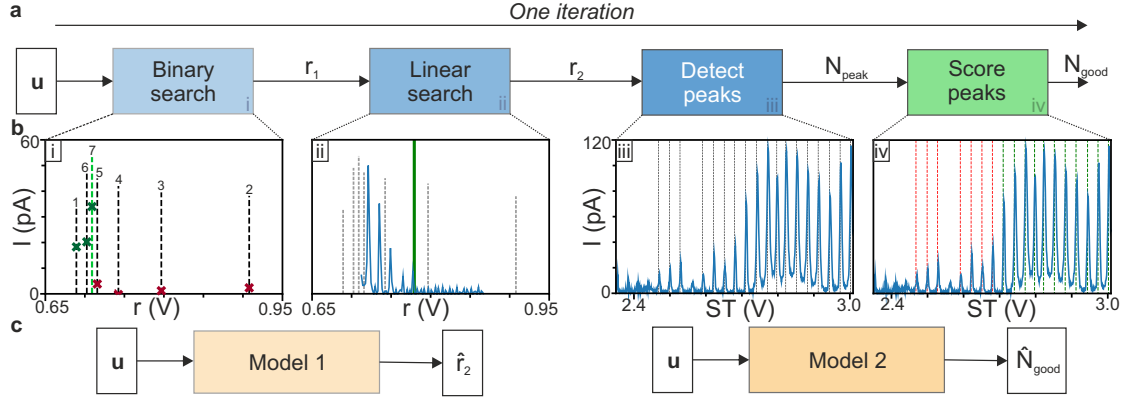


Figure 6.3: Overview of the algorithm's sequence. **a** Sequence of experiments for a given iteration. An iteration is parameterised by an input unit-vector $\mathbf{u}$. The sequence of experiments proceeds as follows: A binary search is used to find r_1 ; a linear search is used to find r_2 ; V_{ST} is swept in order to detect Coulomb peaks; and a sensitivity score is applied to each detected peak. **b** Example of the experiments performed for a given iteration of the algorithm. Plot **i** demonstrates a binary search. Red (green) crosses indicate a false (true) binary evaluation. The sequence of observations is indicated on the plot above each evaluation. The final evaluation highlighted with a dashed green line is returned as a value for r_1 . Plot **ii** demonstrates a linear search. The blue line shows current measurements. The solid green line indicates the first point in the sliding window where the search condition is satisfied, r_2 . Plot **iii** demonstrates peak detection. Vertical dashed lines indicate the locations of detected Coulomb peaks. Plot **iv** demonstrates Coulomb peak scoring. Vertical dashed lines indicate peaks detected in **iii** and red (green) lines denote peaks that have been identified to have a sensitivity score above (below) the sensitivity threshold. **c** Modelling of the sequence. $\tilde{p}(\hat{r}_2|\mathbf{u})$ is trained to predicted r_2 given $\mathbf{u}$, $\tilde{p}(\hat{N}_{\text{good}}|\mathbf{u})$ is trained to predicted N_{good} given $\mathbf{u}$.

is $\bar{g}_m < g_{m,\text{thresh}}$ where $g_{m,\text{thresh}}$ is a sensitivity score threshold to be satisfied and $\bar{g}_m$ is the sensitivity score extracted from experiments. The algorithm's goal is thus to efficiently explore the gate voltage space of the device in order to find regions that satisfy such condition. The algorithm operates similarly to that presented in Chapter 4; it has an investigation stage and a sampling stage.

6.2.4 Investigation stage

This algorithm performs a sequence of experiments that are defined by a single input parameter; a unit vector $\mathbf{u}$ (in a similar way to the algorithm presented in chapter 4). The experimental sequence performed is displayed in Fig. 6.3a. Examples of the measurements performed can be seen in Fig. 6.3b.

Improved search for the hypersurface

The first experiment in the sequence performs a search for the hypersurface to find a radius r_2 , in a similar way to how $r_{\max}$ is found in section 4.5.1. In this section, however, the hypersurface has to be located more accurately. The gate voltage step size defined in section 4.5.1 was $\delta = 10 \text{ mV}$, but Coulomb peaks in these devices are observed to have a width of order $\sim 1 \text{ mV}$. Coulomb peaks can also have heights far smaller than 20% of the maximum current. The search for is thus performed with a smaller $\delta = 2 \text{ mV}$, a larger $K_s = 50$ and a lower current threshold. Decreasing δ by a factor of 5 increases the time cost of searching for the surface by 5 times.

I thus introduce a search for an intermediate radius r_1 , speeding up the search for r_2 . To search for r_1 , I use a binary search instead of a linear search. This efficiently removes monotonic regions of gate voltage space, allowing the linear search to focus on the search of Coulomb peaks. The search for r_1 introduces a second current threshold $I_{r_1, \text{thresh}} > I_{\text{thresh}}$ such that $r_1 < r_2$, meaning r_1 lies inside the volume of high current defined by the hypersurface. To set these current thresholds, the mean current μ_{noise} and standard deviation σ_{noise} is characterised in the pinched off regime. The current thresholds used are $I_{\text{thresh}} = \mu_{\text{noise}} + 5\sigma_{\text{noise}}$ and $I_{r_1, \text{thresh}} = \mu_{\text{noise}} + 10\sigma_{\text{noise}}$.

An example of the search for r_1 and r_2 is displayed in Fig. 6.3b i and ii. In this example, the binary search is completed in 7 evaluations, placing r_1 70.3 mV away from r_2 . r_2 is subsequently found in 88 evaluations. Using only the linear search would require 434 evaluations.

If the hypersurface is identified using this approach, the next experiment in the investigation stage starts at the gate voltage location $\mathbf{s} = \mathbf{u}r_2 + \mathbf{o}$, where $\mathbf{o}$ is the origin of the search. If no surface is found, the investigation stage is stopped and the sampler proposes a new unit vector $\mathbf{u}$.

Coulomb peak count

From point $\mathbf{s}$, the algorithm looks for Coulomb peaks. This is achieved by sweeping gate voltage V_{ST} while measuring the current I . V_{ST} is varied in steps of $\delta_{\text{ST}} = 1 \text{ mV}$

and other gate voltages remain fixed. V_{ST} is swept until a halting condition is met. The same Coulomb peak detection method used in section 4.5.2 is applied to the current measurement in order to extract: the number of peaks N_{peaks} , their edges $V_{ST:\text{left}}$ and $V_{ST:\text{right}}$, and the location of their maxima $V_{ST:\text{mid}}$.

The full halting condition is:

$$\text{HALT} = (\text{TURNON} \wedge \text{NOPEAK}) | \text{OUTOFBOUNDS} \quad (6.4)$$

TURNON is true if current rises above I_{thresh} in a sliding window of $K_{s,\text{TURNON}} = 50$ (see section 4.5.1). OUTOFBOUNDS is true if the voltage difference between gate voltages equals $\Delta V_{\text{safe}} = 2.5 \text{ V}$. NOPEAK will return true if no Coulomb peaks are identified in the last 50 mV of the V_{ST} sweep.

Coulomb peak sensitivity scoring

Each detected peak is assigned a sensitivity score $g_{m,n}^-$, using the definition in equation 6.3. The index n denotes a given peak.

To compute the score corresponding to the n th peak, the numerical derivative of the current $dI_{V_{ST}}$ is calculated. A value of $dI_{V_{ST},n}$ is then assigned to each detected Coulomb peak by taking the maximum value of $dI_{V_{ST}}$ within $V_{ST:\text{left},n}$ and $V_{ST:\text{right},n}$.

A weighted average over distances between identified Coulomb peaks is calculated as

$$\Delta V_g \approx \sum_{n=1}^{\hat{N}_{\text{peaks}}-1} w_n (|V_{ST:\text{mid},n+1} - V_{ST:\text{mid},n}|), \quad (6.5)$$

where the weights w_n are produced using kernel density estimation (with a Gaussian kernel and bandwidth of 3 mV). This weights assign a low likelihood to anomalously large values of ΔV_g arising due to peak detection errors.

The peaks sensitivity scores $g_{m,n}^-$ are then calculated using $dI_{V_{ST},n}$ and ΔV_g . The number of detected Coulomb peaks which satisfy the sensitivity condition are returned as N_{good} . After scoring and counting investigation is stopped and the sampler proposes a new unit vector $\mathbf{u}$.

6.2.5 Sampling stage

The algorithm iterates between investigating the device and generating a new candidate $\mathbf{u}$ (in a similar way to the algorithm presented in chapter 4). The sampler in this chapter biases sampling to improve the rate at which Coulomb peaks that satisfy the condition $\bar{g}_m < g_{m,\text{thresh}}$ are found. In order to achieve this the strategy from Chapter 4 is employed; device behavior is modelled and the resulting models are used for sampling.

Modelling the device

At a given iteration i the investigation stage produces a set of results $Y_i = (r_{2,i}, N_{\text{good},i})$, where $r_{2,i}$ is the distance from the origin found by the hypersurface search and $N_{\text{good},i}$ is an integer denoting the number of Coulomb peaks found that satisfy the sensitivity criterion. All information gathered until iteration i is stored as $Y_{1:i} = (Y_j)_{j=1,\dots,i}$ along with all proposed unit vectors $X_{1:i} = (\mathbf{u}_j)_{j=1,\dots,i}$.

At the end of a given iteration i , the sampler models the device using a set of models $\tilde{P} = \{\tilde{p}(\hat{r}_2|\mathbf{u}), \tilde{p}(\hat{N}_{\text{good}}|\mathbf{u})\}$. Given an input X , these models predict outputs of the investigation stage $\hat{Y} = \{\hat{r}_2, \hat{N}_{\text{good}}\}$ by using $Y_{1:i}$ and $X_{1:i}$ as training data. The variables in $\hat{Y}$ are the same than those in Y_j , with hats denoting that such variables represent predictions.

Both models in $\tilde{P}$ are Gaussian process regression models. The first model predicts the hypersurface location and the second predicts the number of Coulomb peaks that will satisfy the sensitivity criterion. Both models use a Matern52 kernel and are initialised with a prior mean of 2.0 and 1.0 respectively. All other kernel hyperparameters like length scales and variances are periodically updated using MAP. Note that during the reproducibility experiment (section 6.2.8) hyperparameters were shared between runs. This has the effect of including an informative prior for MAP. Such an informative prior can easily be recreated by setting reasonable maximum and minimum values to bound the MAP optimisation. I demonstrated how this can be achieved in appendix .12.

Sampling using the device models

The sampler starts out in an initialisation stage lasting for 30 iterations. In this initialisation stage models are not used for generating candidate unit vectors, unlike in section 4.6.2. In this case, unit vectors are sampled uniformly from a unit hyper-sphere. After the initialisation stage, the sampler aims to generate those $\mathbf{u}$ associated with large values of $\hat{N}_{\text{good}}$, while occasionally exploring regions with comparatively low values of $\hat{N}_{\text{good}}$. The sampler excludes regions of gate voltage space for which $\hat{N}_{\text{good}} = 0$.

The Brownian motion approach detailed in section 4.6.2 is used to generate uniform candidate locations on the hypersurface. Each candidate location on the hypersurface is assigned a weight proportional to the value $\mu_{\text{good},\mathbf{u}}$ where $\tilde{p}(\hat{N}_{\text{good}}|\mathbf{u}) = N(\mu_{\text{good},\mathbf{u}}, \sigma_{\text{good},\mathbf{u}}^2)$. This weighting condition is effectively equivalent to aiming to draw uniform samples from $p(S, N | \bar{g}_m < g_{m,\text{thresh}} = \text{True})$. This has a similar effect to that of the Thompson sampling approach used in chapter 4. The variable S represents the hypersurface and it is the same as that defined in section 4.3. The variable N is categorical, selecting from Coulomb peaks found for those $\mathbf{s} \in S$. That is to say that $N = 2$ for a given $\mathbf{s} \in S$ selects the second peak found when searching for Coulomb peaks from $\mathbf{s}$.

6.2.6 Unstructured point-sets

The output of the algorithm is an unstructured set of M samples $X_{1:M}$. Each sample $x_i \in X_{1:M}$ has associated properties

$$d_i = \{r_{1,i}, r_{2,i}, \bar{g}_{m,i,1:N_i}, v_{\text{points},i,1:N_i}\} \quad (6.6)$$

and the set of all properties is $D_{1:M} = \{d_i\}_{i=1,\dots,M}$. For iteration i , the data that d_i contains is the following: $r_{1,i}$, the radius found by the binary search, $r_{2,i}$, the radius found by the combined binary search and linear search, $\bar{g}_{m,i,1:N_i} = \{\bar{g}_{m,i,n}\}_{n=1,\dots,N_i}$ all measured sensitivity scores for all N_i Coulomb peaks in iteration i , and $v_{\text{points},i,1:N_i} = \{\mathbf{v}_{\text{points},i,n}\}_{n=1,\dots,N_i}$, all locations in gate voltage space for all N Coulomb peaks in iteration i .

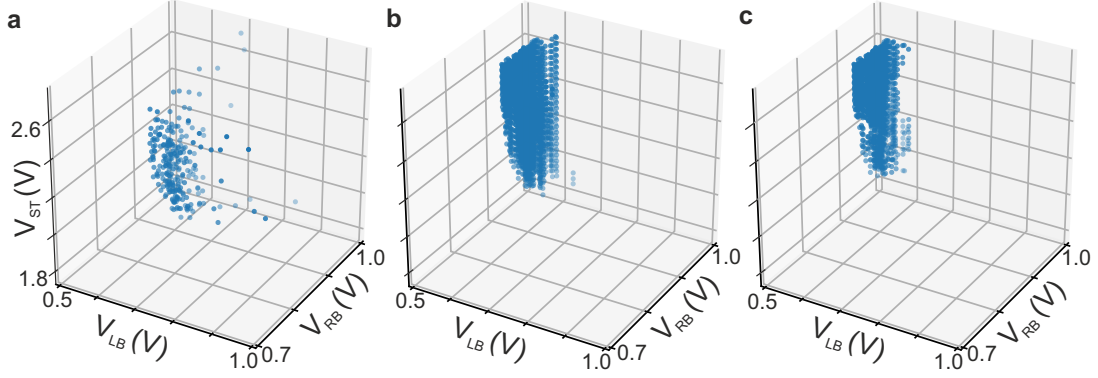


Figure 6.4: Point sets obtained. **a** Hypersurface point set (v_{surf}) represented as a scatter plot. **b** Coulomb peak point set (v_{peak}) represented as a scatter plot. **c** Sensitive Coulomb peak point set ($v_{\text{sensitive}}$) represented as a scatter plot.

Using $X_{1:M}$ and $D_{1:M}$ I can produce three point sets; all points on the surface,

$$v_{\text{surf}} = \{\mathbf{v}_{\text{surf},i}\}_{i=1,\dots,M}, \text{ where } \mathbf{v}_{\text{surf},i} = r_{2,i}\mathbf{u}_i + \mathbf{o}; \quad (6.7)$$

all locations in gate voltage space identified as Coulomb peaks

$$v_{\text{peak}} = \bigcup_{i=1}^M v_{\text{points},i,1:N_i}; \quad (6.8)$$

and all locations in gate voltage space identified as Coulomb peaks that are below the sensitivity threshold

$$v_{\text{sensitive}} = \bigcup_{i=1}^M \{v_{\text{points},i,n}; (\bar{g}_{m,i,n} < \delta g_{m,\text{thresh}})\}_{n=1,\dots,N_i}. \quad (6.9)$$

The algorithm was run for $M = 250$ iterations and a scoring threshold of 0.33 nA^{-1} in order to gather $X_{1:M}$ and $D_{1:M}$. I then used these to compute point sets v_{surf} , v_{peak} , and $v_{\text{sensitive}}$. Figure 6.4 shows the resulting point sets. A cube like hypersurface, similar to that shown in section 4.2, is observed. Samples are clearly biased along one edge of the hypersurface. This edge belongs to the intersecting pinch-off planes produced by V_{LB} and V_{RB} . Coulomb peaks found, the majority of which satisfying the scoring threshold, form a large connected volume.

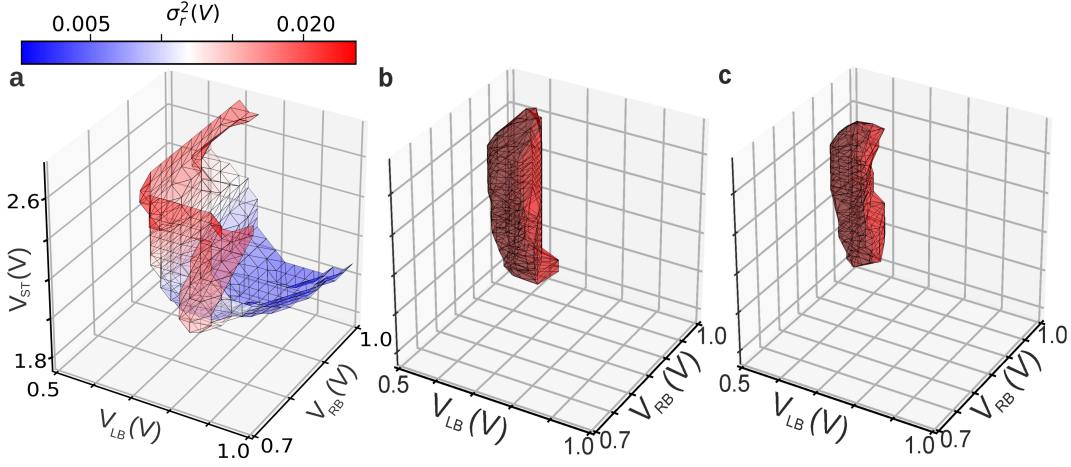


Figure 6.5: Extracted volumes. **a** Hypersurface generated for the point set v_{surf} from Fig. 6.4a. This hypersurface was obtained by training a Gaussian process model to predict $\tilde{p}(\hat{r}_2|\mathbf{u})$ and applying marching cubes to the model. The colour map displays the GP’s uncertainty in its prediction of r across the hypersurface. **b** Alpha-shape generated for v_{peak} from Fig. 6.4b. **c** Alpha-shape generated for $v_{\text{sensitive}}$ from Fig. 6.4c.

6.2.7 Alpha-shapes for volume reconstruction

The point sets acquired in the previous section are difficult to interpret. I propose a methodology to convert them into an approximate volume based on alpha-shapes.

An alpha-shape [95] is a generalisation of a convex hull. It is a set of piece-wise simple curves that is defined by a parameter α and a point set. If $\alpha = 0$ then the alpha shape that is produced by a given point set is a convex hull. An alpha-shape can be used as a method which, by considering the distance $\frac{1}{\alpha}$, allows us to determine whether two points belong to the same volume.

Figure 6.5 shows the volumes predictions extracted with this approach for the point sets displayed in Fig. 6.4. In this case, $\alpha = 10$. Volumes appear to be subsets of each other with the sensitive peak volume being the smallest volume. Both the the peak and sensitive peak volumes are elongated indicating they are less influenced by V_{ST} then V_{RB} or V_{LB} . The hypersurface Figure 6.5a was not obtained using an alpha-shape, but by training a Gaussian process to predict $\tilde{p}(\hat{r}_2|\mathbf{u})$ and applying the marching cubes algorithm like detailed in (similar to section 4.2 but with the Gaussian process in place of the three dimensional grid scan).

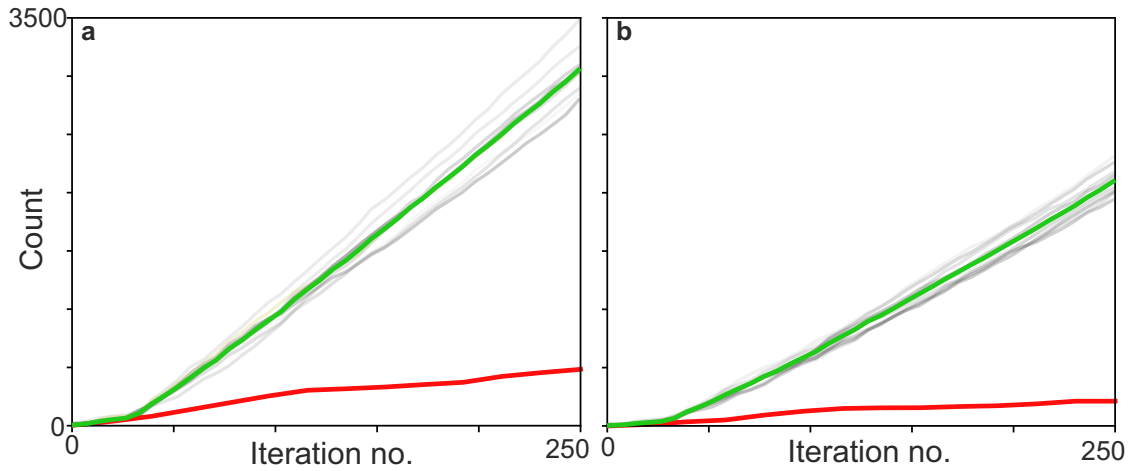


Figure 6.6: Rate of Coulomb peak acquisition. **a** Number of Coulomb peaks identified as a function of the number of iterations for 10 algorithm runs (grey lines). The average performance is indicated with a green line. The performance of the baseline method, *Initialisation only*, is displayed in red. **b** Number of high-sensitivity Coulomb peaks identified as a function of the number of iterations for 10 algorithm runs (grey lines). The average performance is indicated with a green line. The performance of the baseline method, *Initialisation only*, is displayed in red.

6.2.8 Reproducibility

To examine the reproducibility of the algorithm, it was run nine times. For each run, samples $X_{1:M}$ are investigated and data $D_{1:M}$ is stored. I examine the resulting $X_{1:M}$ and $D_{1:M}$ for each run. Figure 6.6 shows the rate at which Coulomb peaks are identified, as well as the rate at which high-sensitivity Coulomb peaks are found, for all ten runs. The average performance is indicated. Figure 6.6 also shows the rate of Coulomb peaks identified with a baseline algorithm *Initialisation only*, for which the sampler remains in the initialisation state, i.e. unit vectors are always sampled from a unit hyper-sphere. The Bayesian statistics approach used in section 4.9.2 (without considering multiple labellers), can be employed to estimate the number of iterations expected to take place between the identification of two Coulomb peaks. I found that 0.082 (0.081, 0.083) of an iteration is expected between the identification of two Coulomb peaks and 0.119, (0.118, 0.120) of an iteration is expected between high-sensitivity Coulomb peaks. The *Initialisation only* implementation performs worse with corresponding values of 0.455, (0.441, 0.470) and 0.872, (0.834, 0.910), respectively.

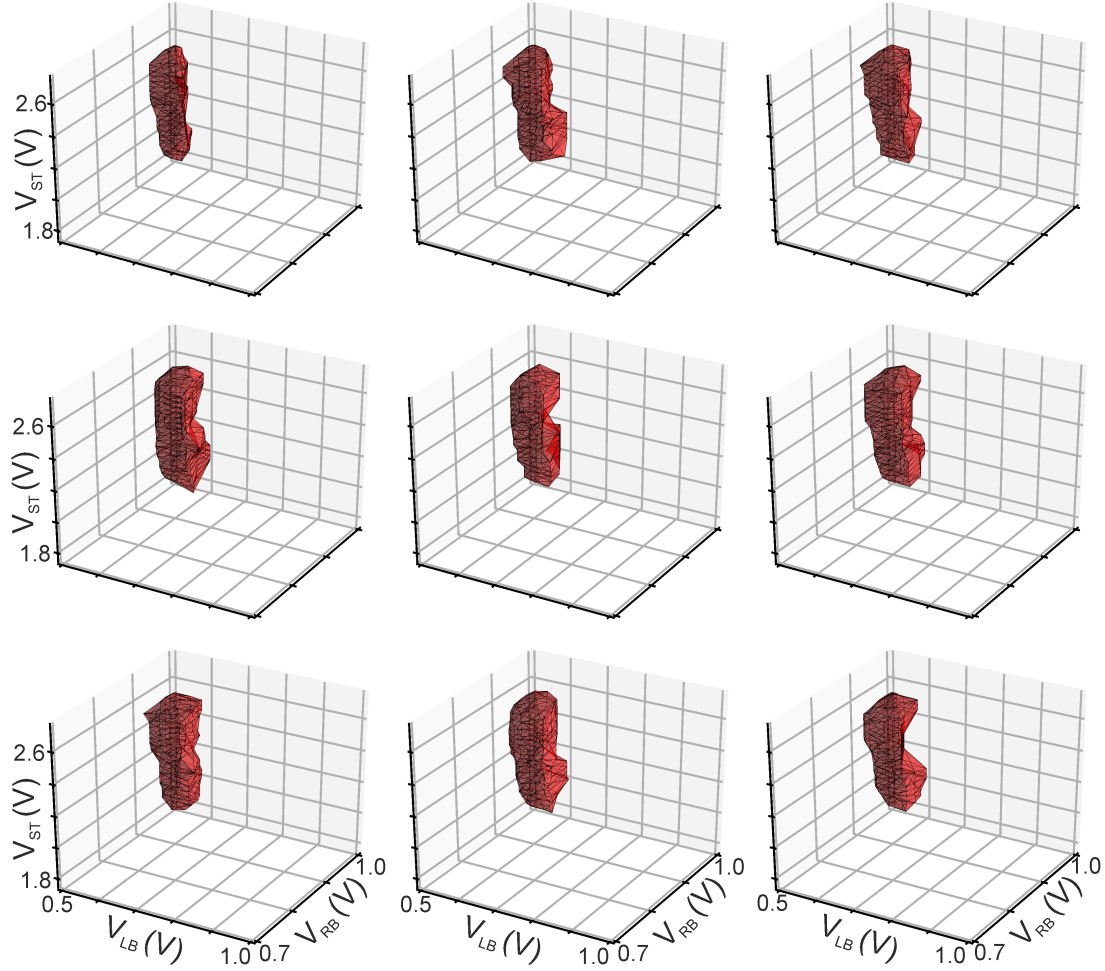


Figure 6.7: Reproducibility of sensitive volumes. Alpha-shapes obtained for nine algorithm runs. Axes are the same in all the plots displayed.

The nine volumes obtained using the approach detailed in section 6.2.7 for $\alpha = 10$ are plotted in Fig. 6.7. Even though only 250 iterations of the algorithm are considered, the volumes obtained have a similar size and are approximately co-located in gate voltage space. To verify that learned volumes are similar, I investigate the volume overlap to estimate a classification accuracy thereby measuring the similarity between volumes. Samples of gate voltage values inside a given volume are generated uniformly from inside one alpha-shape model and classified using a second alpha-shape model to check if the sample is inside both volumes. Doing this for all nine volumes and pairings produces a 9×9 matrix of accuracy values displayed in Fig. 6.8. The average value of off-diagonal elements is 0.865, with a standard deviation of 0.093. I thus conclude that volumes overlap significantly.

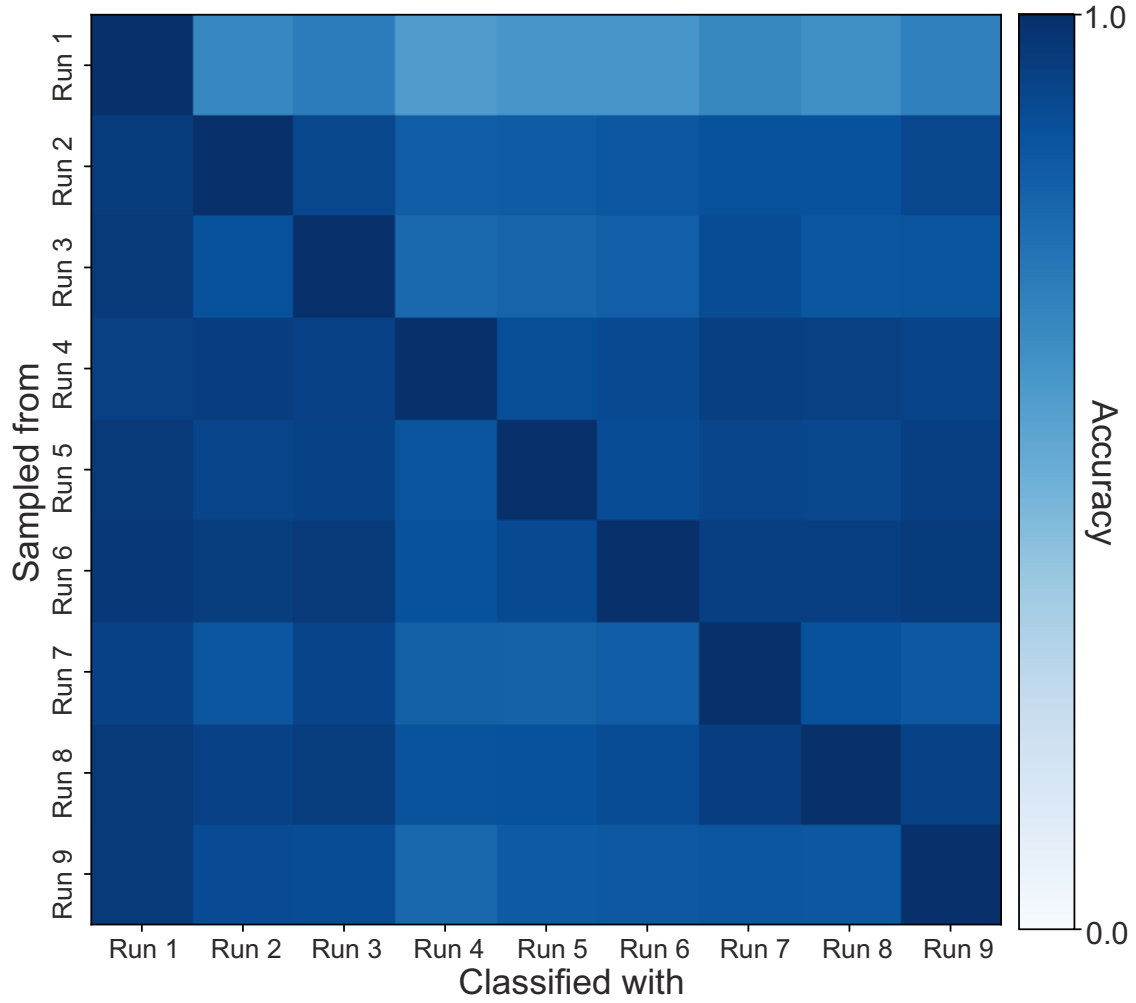


Figure 6.8: Sensitivity volume overlap. Classification accuracy matrix obtained for samples gate voltage values inside a given volume drawn uniformly from inside the alpha-shape model trained with data from run i , and classified by the alpha-shape model trained with data from run j .

High-accuracy and low-accuracy values are randomly distributed in the accuracy matrix, indicating that errors are not systematic. The exception is run 1, which predicts a comparatively small volume. The average volume predicted by the models is 0.00271 V^3 with a standard deviation of 0.00034 .

6.3 Discussion

I demonstrate that tools used for a different purpose in the experiments described in Chapter 4 can be used to quantify device variability and characterise the sensitivity of a charge sensor. I show that uniform search can be designed to target efficient

acquisition of specific point sets. Post-processing methods can be applied to point sets to quantify device variability and charge sensor performance. I develop two post-processing methods: point set registration for learning device variability and alpha shapes for learning the volume in which a sensor device is considered operable. These methods will be most useful for improving fabrication and device design processes through the quantification of performance/variability in large batch's of similar or nominally identical devices.

Future work can extend on this by incorporating additional experiments in the investigation stage to improve characterisation. A step that characterises charge noise could be included to improve the sensitivity estimate presented in section 6.2.2.

7

Automatic fine tuning of a quantum device

In this chapter I demonstrate how modern machine learning methods can be utilised to produce score metrics for tasks such as device fine tuning. The focus is on generating a scoring metric for transport features, with very little labelled training data.

7.1 The device

The device used for the experiments described in this chapter is the same as Device 1 from section 4.1, displayed in Fig. 4.1. Gate electrodes V_1 to V_8 are used to define and control the confinement potential. V_{bias} drives a current I through the device.

7.2 Algorithm's overview

The algorithm presented in this chapter aims to improve transport features already found through the method discussed in chapter 4. The specific transport features I aim to improve are bias triangles; small isolated regions of current that appear at the triple points of the honeycomb patterns detailed in section 1.5.1. These features are important as they indicate operable points of the device and allow us to characterise and fine tune the confinement potential. When bias triangles do not appear as sharp features, as in the lower inset of Fig. 7.1a for example, the device might need to be fine tuned by modifying the set of gate voltages that had been fixed.

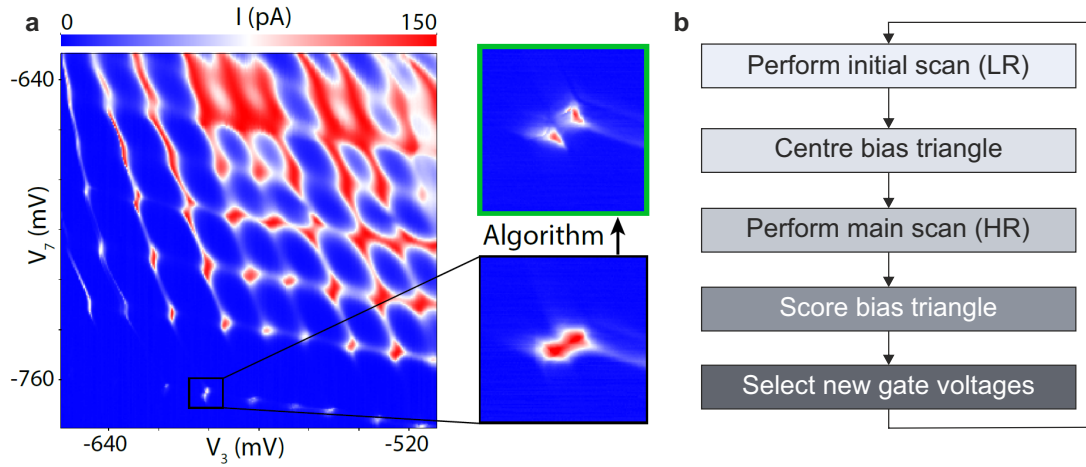


Figure 7.1: Algorithm overview **a** A current map showing a honeycomb pattern. A pair of bias triangles is highlighted. Inset shows a zoom in of the pair of bias triangles before and after application of the algorithm. **b** Flow diagram of the algorithm.

Figure 7.1b illustrates the algorithm’s overview. The algorithm assigns a score to a current map containing bias triangles, then a simple greedy optimiser proposes a new set of gate voltages to explore. To keep bias triangles from drifting outside of the gate voltage window, they are tracked and centered. Bias triangles, and in particular the characteristics that need to be fine tuned, are not easy to parametrise and thus cannot be easily optimised. All the current maps in this chapter are acquired via grid scanning (section 1.8.1).

7.3 Scoring bias triangles via embedding and distance metric

The score function used learns a low dimensional embedding in a latent space. Every current map measured can be mapped into this low dimensional space. The goal is to learn a mapping for which the local neighborhood of an embedded current map is occupied by current maps that are contextually similar. For this purpose, a Variation Autoencoder (VAE) [75] is used. The final score uses current maps labelled by human as ‘ideal’ to compute distances in this low dimensional space.

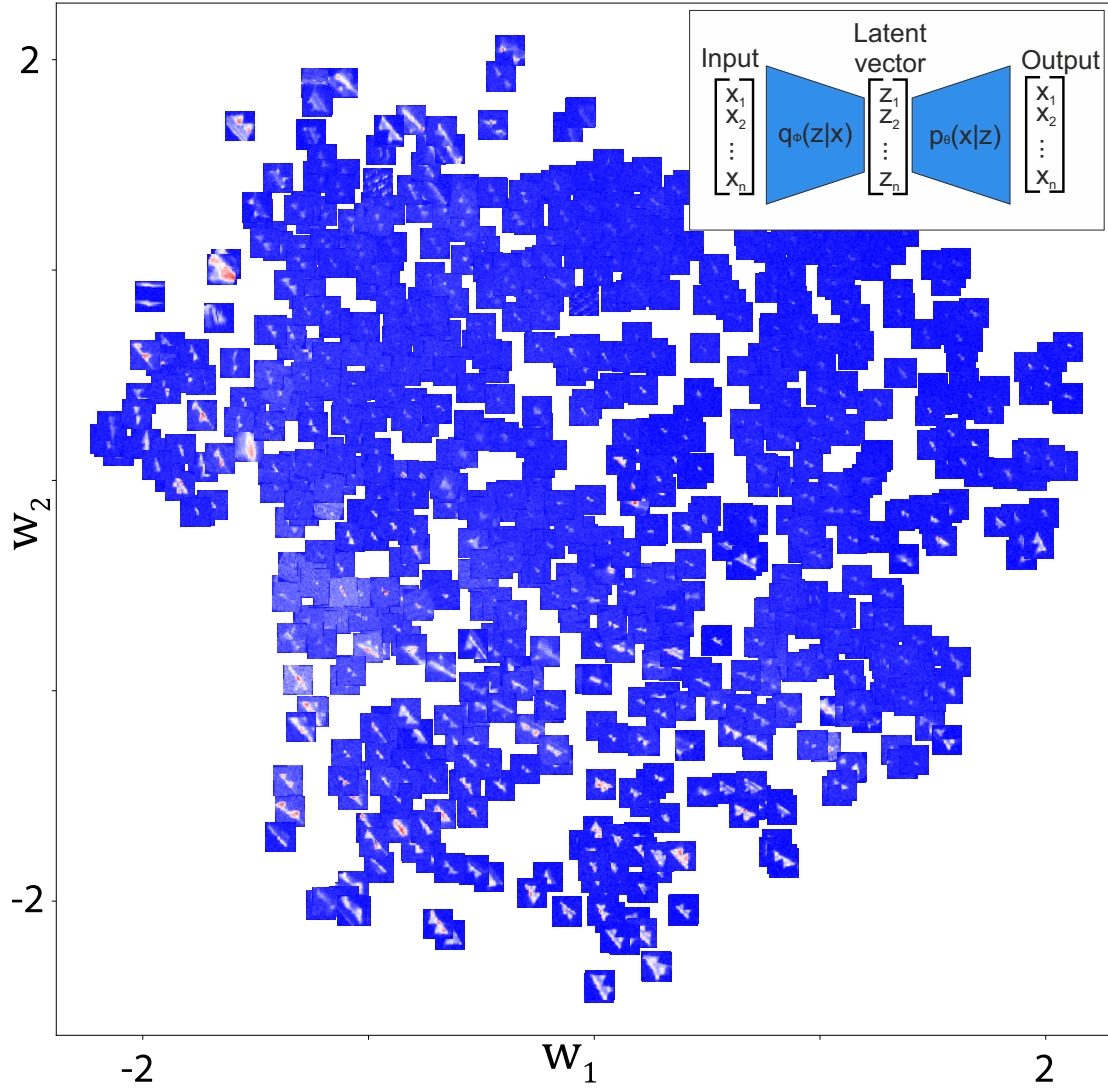


Figure 7.2: The latent space. A two dimensional representation of the trained latent space. Training examples are embedded into the 10 dimensional latent space $\mathbf{z}$ using the encoder $q_\phi(\mathbf{z}|\mathbf{x})$. The set of all 10 dimensional latent vectors are further embedded into two dimensions, producing vectors w . This is achieved by using the t-SNE algorithm. Each current map is plotted in the location it is embedded. Inset: VAE diagram.

7.3.1 Low dimensional embedding via VAE

The VAE consists of an encoder and decoder, which are probability distributions whose distribution parameters are computed by convolutional neural networks [75] (CNN). The encoder $q_\phi(\mathbf{z}|\mathbf{x})$ maps input data $\mathbf{x}$ to a low-dimensional latent vector $\mathbf{z}$ which is real-valued. The decoder $p_\theta(\mathbf{x}|\mathbf{z})$ maps a latent vector to a reconstruction $\hat{\mathbf{x}}$. The parameters of the encoder and the decoder CNN's are ϕ and θ , respectively. The VAE is a generative model, it seeks to preserve the information during the

encoding process so that input data can be reconstructed with minimal error during the decoding process. During a training phase, ϕ and θ are iteratively updated to minimise a loss function. The loss function is given by a reconstruction error $\mathcal{L}_{rec}$, which penalises the networks for producing reconstructions that are dissimilar from the input data, and a regularisation term $\mathcal{L}_{reg}$, which enforces input data with similar characteristics to be encoded in close proximity in latent space. The reconstruction error and the regularisation term have weights α and β , respectively. We thus express $\mathcal{L}_{rec}$ and $\mathcal{L}_{reg}$ as:

$$\mathcal{L}_{rec} = -\alpha \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} [\log(p_\theta(\mathbf{x}|\mathbf{z}))] \quad (7.1)$$

$$\mathcal{L}_{reg} = \beta D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) || p(\mathbf{z})) \quad (7.2)$$

.

I use Factor-VAE [96], an adaption of VAE that seeks to generate a latent space in which each dimension corresponds to a unique characteristic of the input data. The Factor-VAE framework assumes that there are underlying independent characteristics or factors associated with the data. This assumption is referred to as disentanglement. If factors are fully disentangled, each can be identified with a dimension in the latent space. By using a Factor-VAE, the aim is to generate a latent space in which each dimension is associated with a single characteristic of the bias triangles, such as size or brightness. In this way, the distance in latent space to a target location results a good metric to score new current maps.

Factor-VAE includes an additional loss term, the correlation loss. The correlation loss encourages the distribution of embeddings $q_\phi(\mathbf{z})$ to be disentangled. It is given by:

$$\mathcal{L}_{Factor-VAE} = \mathcal{L}_{rec} + \mathcal{L}_{reg} + \gamma D_{KL}(q_\phi(\mathbf{z}) || \prod_j q_\phi(z_j)), \quad (7.3)$$

where the total correlation term is given by $D_{KL}(q_\phi(\mathbf{z}) || \prod_j q_\phi(z_j))$, the Kullback-Leibler divergence between the distribution of embeddings $q_\phi(\mathbf{z})$ and the product of the distribution of embedding components $\prod_j q_\phi(z_j)$, with the index j corresponding

to the j th latent space dimension. Note that the distribution of embeddings is given by $q_\phi(\mathbf{z})$, whereas $q_\phi(z_j)$ describes the distribution of a single embedding dimension z_j . The total correlation loss term has a weight γ . Since this term is intractable, it is estimated using a discriminator $D(\mathbf{z})$. The discriminator is trained to classify between non-factorial and factorial samples, i.e. if the input is a sample from $q_\phi(\mathbf{z})$ or a sample from $\prod_j q_\phi(z_j)$.

The training set for the Factor-VAE was collected from a device which differs considerably from the device used to demonstrate the algorithm, evidencing the algorithm's general applicability. The VAE was trained using 2253 sets of bias triangles measured in a double quantum dot defined in a Ge/Si core-shell nanowire [10, 90]. This device is similar to the devices described in section 5.1.2). Simple data augmentation techniques were applied to further improve the training set. Augmentation included rotation, mirroring, Gaussian noise and random contrast, resulting in a total training set of 8732 stability diagrams of pixel resolution 32×32 . The dimension of the latent space was set to 10, as in Ref. [96], given the similar structure of input data. Optimal VAE performance, was found empirically for $\alpha = 34$, $\beta = 1$ and $\gamma = 1$.

Figure 7.2 shows training data plotted in the latent space of the trained VAE. The visualisation is produced using t-SNE [97] in order to achieve the further dimensionality reduction required to plot a 10 dimensional latent space in a 2 dimensional representation. Clusters of similar current maps can be observed in this representation. For example, faint bias triangles or even no bias triangles can be observed in the upper right corner.

7.3.2 Scoring via distance metric

The score metric used by the algorithm is given by the distance between the latent space representation $\mathbf{z}$ of an input current map and the latent space representation of a set of target inputs $Z = \{\tilde{\mathbf{z}}_j\}_{j=1,\dots,B}$. A current map acquired by the algorithm is assigned a low VAE score if its representation in latent space is near to the targets in latent space. As a result, bias triangles that are assigned a low score possess

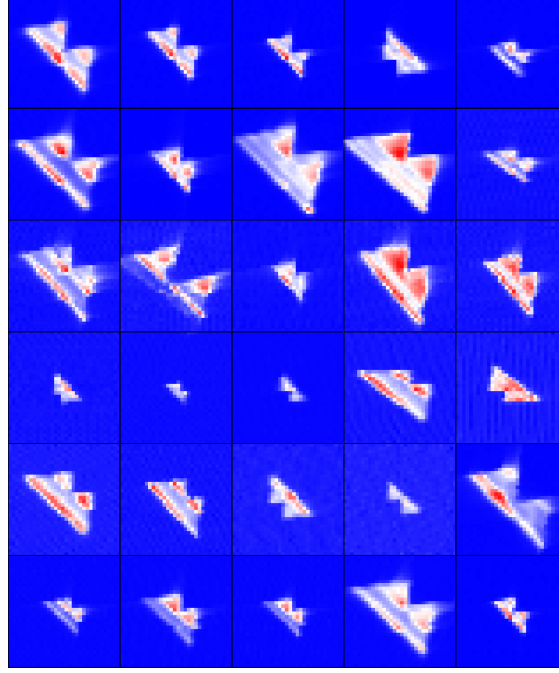


Figure 7.3: Selected targets All 30 target bias triangles selected from the training dataset before augmentation. These current maps are plotted as a function of V_3 and V_7 with independent colour scales from blue (minimum current) to red (maximum current).

similar characteristics to the target bias triangles. Target bias triangles are chosen from the unaugmented training set by a human expert who recognises in these triangles characteristics indicative of favourable quantum dot parameters (Fig. 7.3). Targets are augmented using the same augmentation techniques as described in 7.3.1. 30 targets (before augmentation) were selected, and after augmentation we have 360 targets which were used for scoring. The embedding locations of example current maps and a target is indicated in Fig. 7.4.

The score for a given iteration i of the algorithm is S_i . In iteration i , a current map is acquired and embedded, producing a latent vector $\mathbf{z}_i$. The output of the encoder is trained to follow a multivariate Gaussian with diagonal covariance structure: $q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \mu, \text{diag}(\sigma^2))$, where the mean μ and variance σ^2 are outputs of the encoder. The embedding of $\mathbf{z}$ is normally distributed. The expectation value of the squared distance between two independent embedding distributions is given by:

$$d(\mathbf{z}_i, \tilde{\mathbf{z}}_j) = \mathbb{E} \|\mathbf{z}_i - \tilde{\mathbf{z}}_j\|^2 = \|\mu_{\mathbf{z}_i} - \mu_{\tilde{\mathbf{z}}_j}\|^2 + \sigma_{\mu_{\mathbf{z}_i}}^2 + \sigma_{\tilde{\mathbf{z}}_j}^2 \quad (7.4)$$

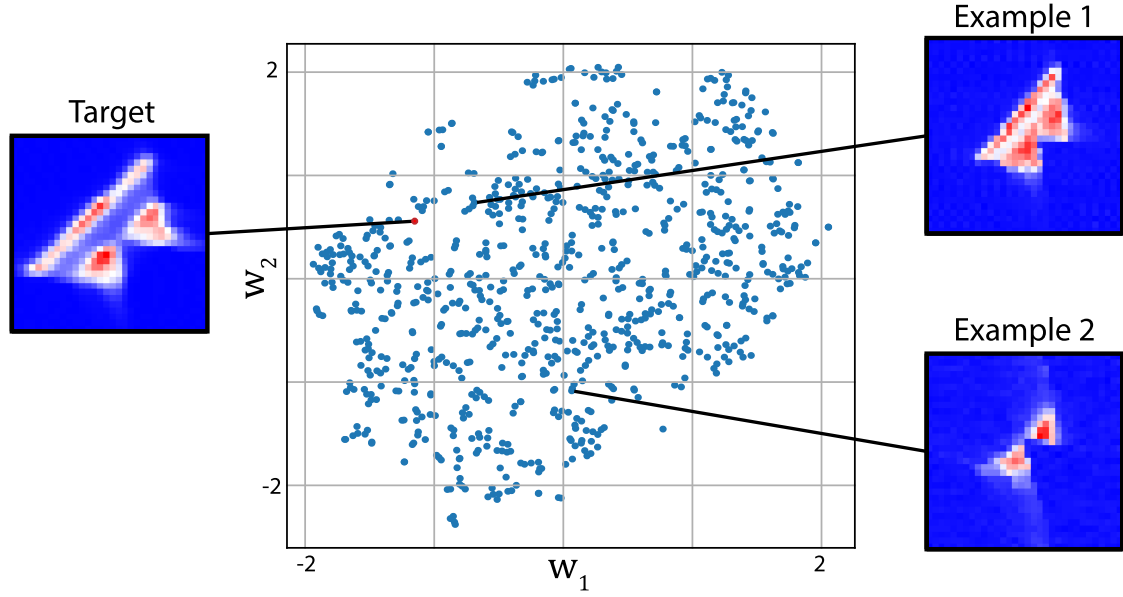


Figure 7.4: Scoring using the embedding. It is expected that embeddings which are close to each other in latent space are generated by input data with similar characteristics. Test example 1, with similar characteristics to the target, can be found in close proximity to the target, whereas test example 2 is further away in latent space. The target and example current plots are a function of V_3 and V_7 and use a colour scale running from red, the highest current measured, to blue, the lowest current.

For each input current map embedded in latent space, $\mathbf{z}_i$, 7.4 is used to determine the distance in latent space to target input $\tilde{\mathbf{z}}_j \in Z$. The final score S_i consists of the average of the distance to its k nearest targets:

$$S_i = \frac{1}{k} \sum_{\tilde{\mathbf{z}}_j \in A_k} d(\mathbf{z}_i, \tilde{\mathbf{z}}_j), \quad (7.5)$$

where $A_k \subset Z$ is the set of k targets closest to $\mathbf{z}_i$ according to the metric defined in Eq. 7.4. Optimal tuning corresponds to a low score. It was found that for $k = 3$, the score metric produced a robust ranking of training inputs.

To demonstrate the score measures ability to rank bias triangles in order of quality it was applied to the withheld data set. Figure 7.5 displays the ranking of a randomly sampled subset. The observed ranking is in very good agreement with human preference. Bias triangles that are well defined and display desirable features such as excited states are assigned the lowest S_i while poorly defined, or faint bias triangles are assigned the highest values.

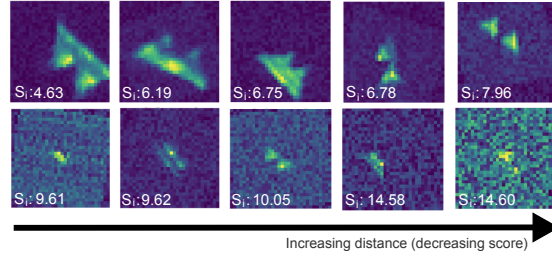


Figure 7.5: Ranking of withheld data using distance measure Ten randomly sampled bias triangles are plotted in order of increasing distance S_i with values displayed.

7.4 Greedy optimisation algorithm

The optimisation algorithm adjusts the gate voltage settings so that new gate voltage configurations can be measured and scored. The decision model is illustrated in Fig. 7.6. In each iteration, one gate electrode is selected at random, and the voltage applied to this electrode is modified by a fixed amount $\pm\Delta V$. Therefore, the algorithm chooses between a number of gate voltage branches equal to twice the number of gate electrodes to be tuned minus one, as branches that lead to the reversal of the latest accepted gate voltage change are excluded. $\Delta V = 2$ mV is chosen based on human experience in tuning similar devices. After centering and acquiring a high resolution grid scan of the resulting bias triangles, the value of S_i determines the algorithm's decision. If S_i is lower than the previous best (lowest) scored bias triangles, the gate voltage change is accepted, leading to a new gate voltage configuration N_{j+1} . Conversely, if S_i is higher, the gate voltage change is rejected and the gate voltage setting returns to its previous configuration. In this case, the rejected gate voltage change will be excluded in the random selection of the next iteration. It is possible that all gate voltage branches become depleted, in which case the decision model returns to a previously accepted gate voltage configuration with unexplored branches.

7.5 Experimental optimisation of bias triangles

The algorithm is tested on different bias triangles measured on the device. Grid scans are performed as a function of barrier gate voltages V_3 and V_7 , which are adjusted

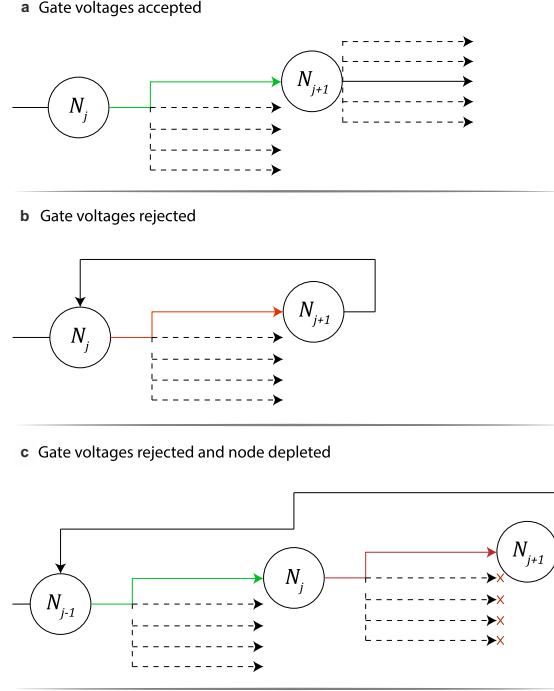


Figure 7.6: Overview of the decision model. The node N_j represents the best scored gate voltage configuration obtained after a number of algorithm iterations. The branched arrows represent different gate voltage adjustment options; $\pm\Delta V$ for every gate electrode. In this case, 5 gate voltage branches are considered as 3 gate electrodes are used for fine-tuning. **a** Score S_i corresponding to a new configuration N_{j+1} is lower than at N_j , so the gate voltage change is accepted. For N_{j+1} , a new random gate voltage branch is selected and explored. **b** Score S_i corresponding to a new configuration N_{j+1} is higher than at N_j , so the gate voltage change is rejected. For N_j , one of the remaining gate voltage branches is randomly selected and explored. **c** If all possible gate voltage configurations are rejected the algorithm returns to the closest previously accepted gate voltage node that has unexplored branches.

during the centering of bias triangles in the current map. The gate voltages tuned by the algorithm are V_1 , V_2 and V_8 . For simplicity, gate voltages V_4 , V_5 and V_6 are fixed. Their effect on bias triangles were observed to be weak. All measured current maps are normalised with respect to the minimum and maximum current values measured during coarse tuning to the double quantum dot regime, achieved with the algorithm from chapter 4. Fig. 7.7, shows fine-tuning of four different bias triangles.

In cases 1 to 3, the initial bias triangles lack a well-defined shape, indicative of small inter-dot tunnel coupling [98]. Pronounced co-tunnelling lines, which are denotative of second-order transport processes, are observed. As the fine-tuning

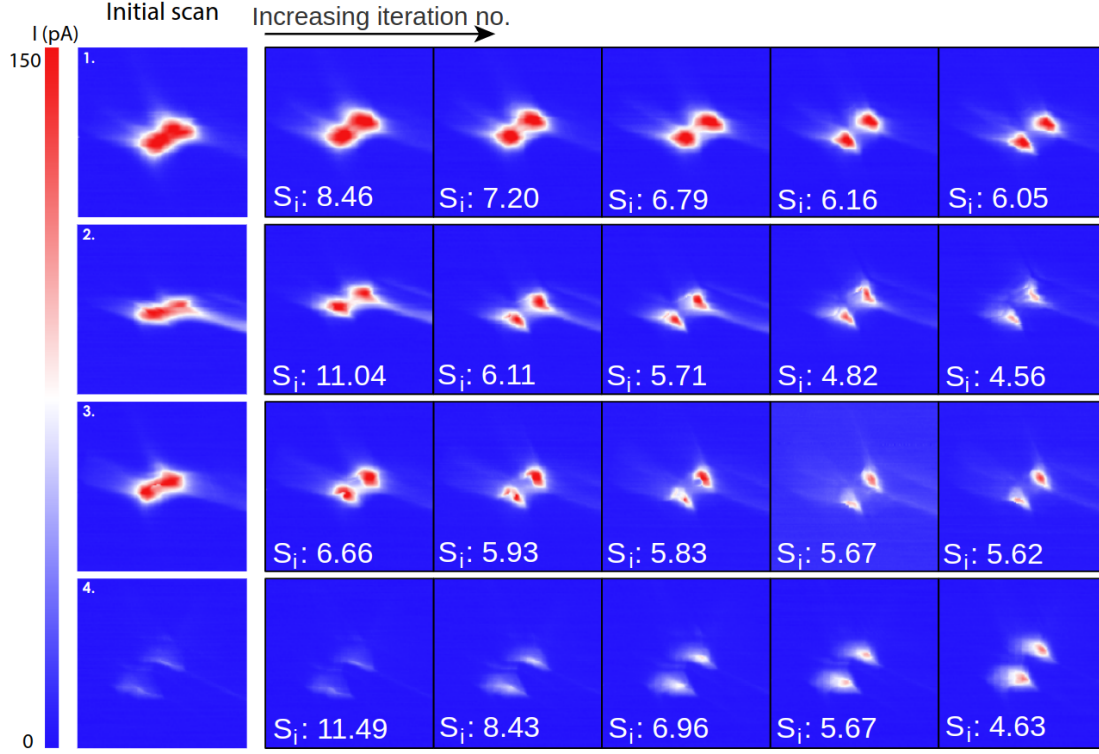


Figure 7.7: Experimental demonstration of the algorithm. The first column shows four different pairs of bias triangles before the algorithm was run. Each row displays each of these bias triangles at selected iterations of the algorithm. For all these iterations, the applied gate voltage change led to a decrease in score S_i . All measurements were performed with $V_{bias} = 0.2$ mV. The current maps were measured as a function of barrier gates V_3 and V_7 , while gate voltages V_1 , V_2 and V_8 were tuned by the algorithm.

progresses, the bias triangles separate from each other and acquire a sharper triangular shape. Co-tunnelling currents are also reduced. In the fourth case, the initial current map shows very faint bias triangles. The algorithm proves capable of increasing the current flowing through the double quantum dot while preserving most other bias triangle characteristics.

Figure 7.8a shows S_i as a function of the number of iterations of the algorithm for cases 1 to 4. Most of the fine-tuning takes place during the first ten iterations. After this the score does not change significantly. In all cases, the algorithm completes the fine-tuning within 26 iterations, corresponding to a total tuning time of 36 min. This time is limited by the measurement time.

Fig. 7.8b plots the trajectories in gate voltage space corresponding to each fine-tuning run of the algorithm. The average distance in gate voltage space

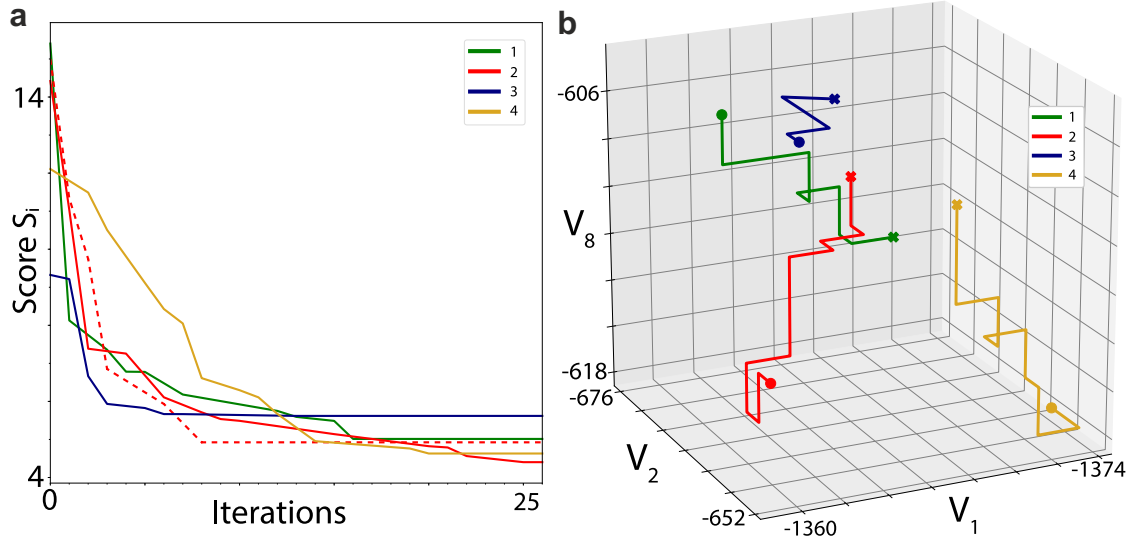


Figure 7.8: Values of the score S_i and gate voltage space trajectories during fine-tuning. **a** Score S_i as a function of the number of algorithm iterations. Solid lines 1 to 4 correspond to the fine-tuning runs presented in Fig. 7.7. The dashed line represents a second run of the algorithm for case 2. **b** Gate voltage space trajectories for the fine-tuning runs in Fig. 7.7. The starting (final) gate voltage configuration is denoted by a circle (cross).

between the initial gate voltage configurations is greater than for the final gate voltage configurations. This suggests that there exists a region in gate voltage space for which the bias triangles exhibit the most favourable transport characteristics, regardless of their values of V_3 and V_7 .

7.6 Discussion

I demonstrate a method for producing a score metric for bias triangles using distances in a low dimensional latent space. The score metric can be applied to other types of measurements if an appropriate training dataset is available. Only a small number of target current maps had to be selected by a human operator to construct the score metric. I shown that this score can be used to locally optimise double quantum dot parameters in a completely automated manner using a simple greedy optimiser. The embedding was trained on data obtained from a Ge/Si core-shell nanowire device and tested on a laterally-defined device in an AlGaAs/GaAs hetrostructure, showing the method is device agnostic.

Future work could make use of the reconstructions produced by the encoder to replace the grid scanning approach with the efficient measurement approach used in chapter 3.

8

Automatic tuning and optimisation of spin qubit readout

I now focus on qubit readout, one of the requirements of the DiVincenzo criteria (section 1.1). I present a general methodology based on parametric density estimation to compute the distribution of spin states being probed. I extract the readout fidelity and show that this approach can be applied to optimise experimental parameters.

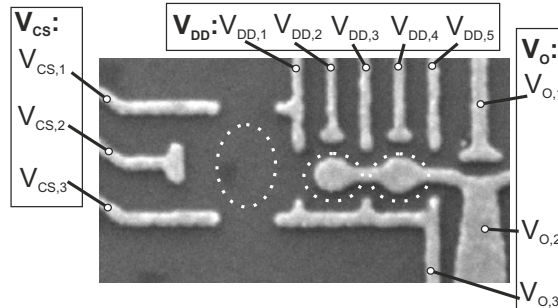


Figure 8.1: Scanning electron micrograph of the device. Dashed white circles show where quantum dots are expected to form. Gate voltages applied to the device are labelled, the set of gate voltages V_{CS} predominantly confines and controls the charge sensor, V_{DD} predominantly confines and controls the double quantum dot, and V_O contributes to the control of the confinement potential.

8.1 The device

In this chapter, experiments are performed in a double quantum dot defined in a GaAs/AlGaAs heterostructure (Fig. 8.1). The device, described in more detail in Ref. [99], allows for on-chip charge sensing; a comparatively large single dot can be defined in close proximity to the double dot (section 6.2.2).

The charge sensor is defined and controlled by the set of gate voltages $V_{\text{CS}} = \{V_{\text{CS},1}, V_{\text{CS},2}, V_{\text{CS},3}\}$. The double quantum dot is defined and controlled by the set of gate voltages $V_{\text{DD}} = \{V_{\text{DD},1}, V_{\text{DD},2}, V_{\text{DD},3}, V_{\text{DD},4}, V_{\text{DD},5}\}$. Additional gates $V_{\text{O}} = \{V_{\text{O},1}, V_{\text{O},2}, V_{\text{O},3}\}$ contribute to the control of the confinement potential. $V_{\text{DD},2}$ and $V_{\text{DD},4}$ can be both DC and high-frequency signals, as the respective gate electrodes are connected to bias tees.

The charge sensor is measured using standard radio frequency reflectometry (rf) techniques [100]. The rf circuit outputs a voltage value q which can be measured using a Digital Multi Meter (DMM) or digitiser. The DMM or digitiser acquires N_τ samples of q over time τ and returns their average $\bar{q}$.

I set gate voltages V_{CS} to tune the charge sensor to an operable condition in which changes in the local electrostatic environment are sensed with high sensitivity (similar considerations to those described in section 6.2.2). The double dot is tuned to specific charge configurations (n,m), where n is the number of electrons in the left dot (LD) and m the number of electrons in the right dot (RD). In order to readout spin states, we tune the double dot to access to the (2,0) - (1,1) inter-dot transition (Fig. 8.2a).

8.2 Estimating the sensitivity of spin state readout

Because the direct measurement of an electron spin is challenging, spin qubit readout relies on spin-to-charge conversion; spin states are mapped into different charge states that can be distinguished, for example, with a charge sensor. I focus on spin qubits encoded in the relative alignment of two electron spins, so the spin

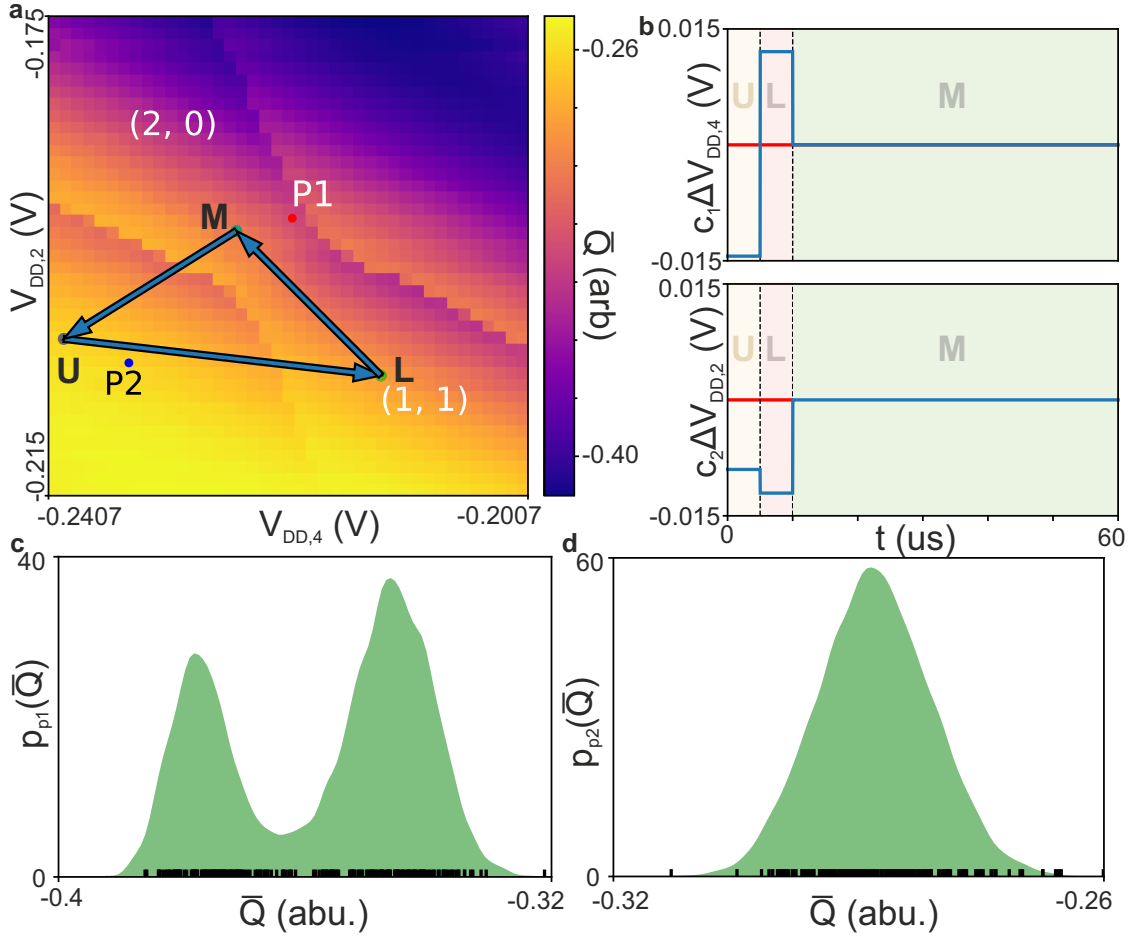


Figure 8.2: T1 sequence overview **a** Current map measured by rf charge sensing showing the (2,0)-(1,1) inter-dot transition. The T1 sequence is indicated with arrows. Gate voltage locations P1 and P2 are highlighted. **b** T1 sequence. Wave forms corresponding to a sequence similar to that indicated in **a**. C_1 and C_2 are constants determined after the calibration of the pulse amplitude. This calibration is required due to the attenuation that the high-frequency signals are subject to in the dilution refrigerator. **c** and **d** Sensor response densities $P_{p1}(\bar{Q})$ and $P_{p2}(\bar{Q})$ approximated via kernel density method for $N = 8000$ runs of the T1 cycle. A subset of the rf charge sensor measurements are plotted along the horizontal axis as black ticks. The averaging time was $\tau = 10 \mu\text{s}$.

states considered in this chapter are singlet and triplet states (section 1.7.2). Pauli spin blockade (PSB) allows for spin-to-charge conversion near the (2,0) - (1,1) inter-dot transition (Fig. 8.2a). High-frequency pulses of $V_{DD,2}$ and $V_{DD,4}$ allow us to probe this inter-dot transition. Starting from the (1,1) charge configuration and pulsing gate voltages towards (2,0), it becomes energetically favorable for the singlet to transition to the (2,0) charge state. The triplet, however, can not make this transition because of spin selection rules, remaining in the (1,1) configuration.

After a time $\sim T_1$, the triplet relaxes into the singlet and the transition from the (1,1) to the (2,0) configuration is then permitted.

The pulse sequence, often called T1 cycle, is illustrated in Fig. 8.2a. It is defined by the wave forms for $V_{DD,2}$ and $V_{DD,4}$ (Fig. 8.2b). These two wave forms allows us to explore the inter-dot transition in three stages. Starting from the charge configuration (2,0), in the first stage (U) an electron from LD into the left reservoir is unloaded. In the second stage (L) an electron is loaded into RD from the right reservoir. The third and final stage (M) returns the charge configuration to (2,0). This cycle allows for spin-to-charge conversion; monitoring $\bar{q}$ during M would allow us to determine if the spin state is a singlet or a triplet. The entire pulse sequence can be parametrised with 9 variables.

I performed a T1 cycle $N = 8000$ times starting from a gate voltage location $V_{DD} = P_1$ indicated in Fig. 8.2a. The result is a set of measurements $\{\bar{q}_i\}_{i=1,\dots,N}$. A subset is represented by black ticks in Fig. 8.2c. For a given cycle, these measurements can correspond either to a singlet or a triplet state. The set of measurements is thus constituted by independent and identically distributed (iid) samples from a probability distribution. I refer to this distribution with the random variable $\bar{Q}$. From such a set of measurements, $p_{P_1}(\bar{Q})$ (where $p_{P_1}(\bar{Q})$ is the density $p(\bar{Q})$ at P_1) can be approximated via kernel density estimation. The result is illustrated in Fig. 8.2c. The bi-modal density indicates that two different charge states are measured. The distribution is thus the result of the sum of two different conditional distributions $p(\bar{Q}) = p(\bar{Q}|S = s)p(S = s) + p(\bar{Q}|S = t)p(S = t)$, where S denotes the state of the qubit. The state s represents a singlet and the state t represents the triplet.

A T1 cycle starting from a gate voltage location $V_{DD} = P_2$ indicated in Fig. 8.2a results in a uni-modal density (Fig. 8.2d). As expected, starting the pulse cycle at P_2 does not fulfill the charge configuration sequence that allows for spin state readout.

8.2.1 A parametric density for spin readout

Parametric density estimation offers a method to approximate $p(\bar{Q})$ without labelled measurements. It aims to approximate its components $p(\bar{Q}|S = s)$ and $p(\bar{Q}|S = t)$ as parametric probability distributions which I derive in this section.

A sample $\bar{q}$ is the average of several measurements q_j made by the digitizer. q_j is also a sample from an iid distribution which I denote Q_j . While averaging N_τ individual measurements are acquired to produce a single measurement, hence

$$\bar{Q} = \frac{1}{N_\tau} \sum_{j=1}^{N_\tau} Q_j, \quad (8.1)$$

where each Q_j is distributed according to

$$Q_j \sim \mathcal{N}(\mu_j, \sigma_j^2). \quad (8.2)$$

The sum of Gaussian random variables is also Gaussian,

$$\bar{Q} \sim \mathcal{N}\left(\frac{1}{N_\tau} \sum_{j=1}^{N_\tau} \mu_j, \frac{1}{N_\tau} \sum_{j=1}^{N_\tau} \sigma_j^2\right). \quad (8.3)$$

When a singlet is loaded, it remains a singlet for the duration of the averaging time, and thus for every individual measurement μ_j and σ_j are the same, hence

$$p(\bar{Q}|S = s) = \mathcal{N}(\mu_s, \sigma_s^2). \quad (8.4)$$

When a triplet is loaded, there is a finite probability that the triplet will relax into the singlet state at any point during the averaging time window. Relaxation events are randomly distributed according to random variable T_{decay} , where

$$T_{\text{decay}} \sim E(\lambda_{T_1}). \quad (8.5)$$

E represents the exponential distribution with parameter $\lambda_{T_1} = \frac{1}{T_1}$ that governs the rate at which relaxation occurs. Relaxation complicates the form of $p(\bar{Q}|S = t)$. When the triplet does not decay during the averaging time window (i.e. $\tau < T_{\text{decay}}$), every individual measurement μ_j and σ_j are the same, hence

$$\bar{Q}_{\tau < T_{\text{decay}}} \sim \mathcal{N}(\mu_t, \sigma_t^2). \quad (8.6)$$

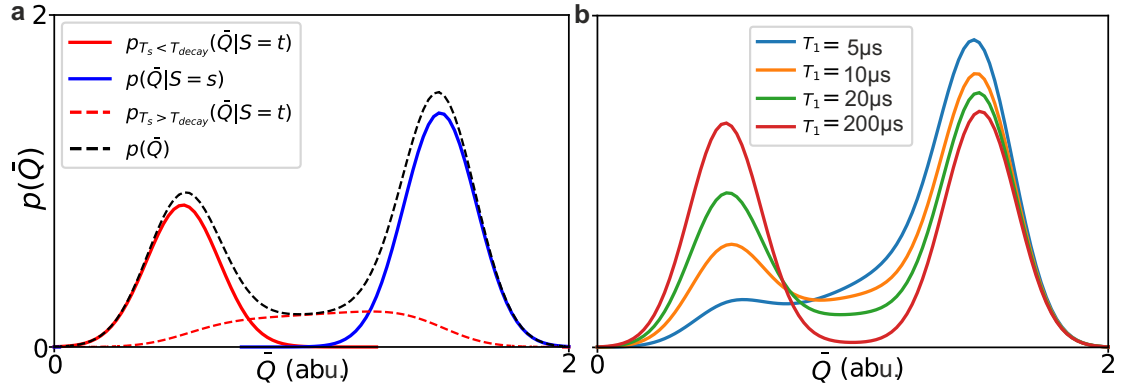


Figure 8.3: Theoretical likelihood of $\bar{Q}$ **a** Individual components of $p(\bar{Q}|\theta)$. The components of equation 8.10 are plotted individually for $\mu_s = 1.5$, $\mu_t = 0.5$, $\sigma_s^2 = 0.02$, $\sigma_t^2 = 0.02$, and $\lambda_{T_1} = 20 \mu s$. **b** $p(\bar{Q}|\theta)$ as a function of λ_{T_1} .

When the triplet relaxes after time t_j but before time t_{j+1} (i.e. $t_j < T_{\text{decay}} < t_{j+1}$), then μ_j and σ_j will change from singlet to triplet and the distribution will be given by a combination of equations 8.4 and 8.6,

$$\bar{Q}_{\tau > T_{\text{decay}}} \sim \mathcal{N}(\alpha_j \mu_t + (1 - \alpha_j) \mu_s, \alpha_j \sigma_t^2 + (1 - \alpha_j) \sigma_s^2), \quad (8.7)$$

where

$$\alpha_j = \frac{j}{N_\tau - 1}. \quad (8.8)$$

The probability that $\bar{Q}$ takes the form of equation 8.6 or equation 8.7 for a specific value of j is governed by the cumulative distribution function (CDF) of the exponential distribution. I represent the (CDF) with $P(\tau < T_{\text{decay}})$ and $P(t_j < T_{\text{decay}} < t_{j+1})$, respectively. The distribution for $p(\bar{Q}|S = t)$ is then

$$p(\bar{Q}|S = t) = \mathcal{N}(\mu_t, \sigma_t^2)P(\tau < T_{\text{decay}}) + \sum_{j=1}^{N_\tau-1} \mathcal{N}(\alpha_j \mu_t + (1 - \alpha_j) \mu_s, \alpha_j \sigma_t^2 + (1 - \alpha_j) \sigma_s^2)P(t_{j-1} < T_{\text{decay}} < t_j). \quad (8.9)$$

I refer to the first and second components of equation 8.9 as $p_{\tau < T_{\text{decay}}}(\bar{Q}|S = t)$ and $p_{\tau > T_{\text{decay}}}(\bar{Q}|S = t)$, respectively.

The parametric form for $p(\bar{Q})$ is then

$$p_{EM}(\bar{Q}|\theta) = w_1 p(\bar{Q}|S = s) + w_2 p_{\tau < T_{\text{decay}}}(\bar{Q}|S = t) + w_3 p_{\tau > T_{\text{decay}}}(\bar{Q}|S = t), \quad (8.10)$$

where the relevant unknown parameters are $\theta = \{\mu_t, \mu_s, \sigma_t, \sigma_s, \lambda_{T_1}, w_1, w_2, w_3\}$. The likelihood of this distribution is plotted in Fig. 8.3. This statistical derivation yields a similar form to that seen in prior literature [101].

8.2.2 Unsupervised learning via Expectation Maximisation

The parametric density has unknown parameters θ . I now show how θ can be learnt efficiently. After a set of measurements $\{\bar{q}_i\}_{i=1,\dots,N}$ are drawn from $p(\bar{Q})$, it is unknown if a measurement $\bar{q}_i$ corresponds to a singlet or a triplet. Therefore, the model cannot be trained in a supervised fashion, i.e. using sets of measurements and sets of labels. I train the model using an unsupervised approach instead. In this case, Expectation Maximisation (EM) is commonly used [102]. EM estimates expected sample membership (the E step) and then tries to maximise the expected likelihood, (the M step); the E and M steps are performed iteratively.

The function of the model in Eq. 8.10 is similar to that of a Gaussian mixture model. All weights, w_1 , w_2 , and w_3 , can be optimised separately using the traditional Gaussian mixture model derivation (see appendix .14). The mean and covariances can also be estimated in this formalism. All that is required is a way to estimate λ_{T_1} . This could be achieved using an optimiser, but I derive and use an approach that uses w_2 and w_3 in order to speed up computation. Given

$$P(\tau < T_{\text{decay}}) = \frac{w_3}{w_2 + w_3}, \quad (8.11)$$

it can be shown that

$$\lambda_{T_1} = -\frac{\ln \frac{1}{\frac{w_3}{w_2} + 1}}{\tau}. \quad (8.12)$$

This approach yields good performance and is quicker than alternative strategies.

Training the model on the measurements from Fig. 8.2c and d yields the results displayed in Fig. 8.4a and b. The trained model accurately explains the distribution, almost completely reproducing the none parametric distribution method. The model produced by training on the samples at P1 corresponds to $T_1 \approx 28 \mu s$.

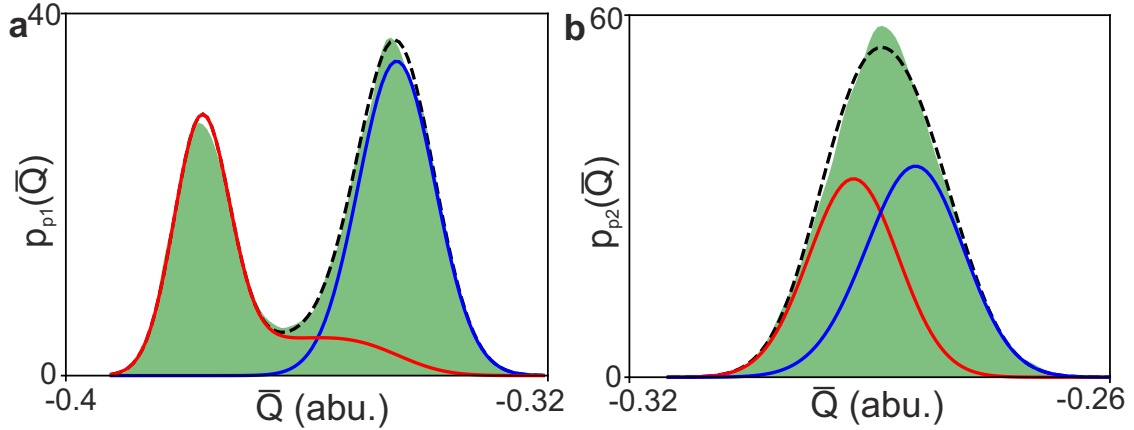


Figure 8.4: Parametric density training. **a** and **b** Parametric densities trained on measurements at gate voltage locations P1 and P2. The sensor response density $P_{p2}(\bar{Q})$ ($P_{p1}(\bar{Q})$) in panel **a** (**b**), approximated via parametric density estimation, is indicated with a black dashed line. Solid blue and red curves show the individual components $p(\bar{Q}|S=s)$ and $p(\bar{Q}|S=t)$, respectively. The Kernel density approximations are plotted as solid green areas for reference.

8.2.3 Validation of model training

To validate the approach I compare it to a commonly used baseline, that of estimating readout density's in a supervised fashion using histograms. This baseline is the standard method for evaluating readout performance and requires labeled training data. The approach taken is to design two experiments that individually generate samples from $p(\bar{Q}|S=s)$ and $p(\bar{Q}|S=t)$. A set of N_{bins} histogram bins are created and samples from the two experiments are used to generate counts for these bins.

In Fig 8.5a and b trained models are displayed after training on samples generated from simulation. Samples generated from simulation are subjected to various levels of Gaussian noise. It is evident from Fig 8.5a that the EM method more accurately recovers the ground truth despite being unsupervised and not having knowledge of labels. Under high noise both methods struggle to recreate the ground truth.

In Fig 8.5c the performance difference between the EM method and the histogram method is measured for different amounts of noise. This is quantified in two ways. The first method computes the KL divergence between the ground truth $p_{\text{actual}}(\bar{Q})$ and the trained model $p_{\text{model}}(\bar{Q})$. The second method computes the predicted fidelity (according to method in section 8.2.5) and actual fidelity of the ground truth

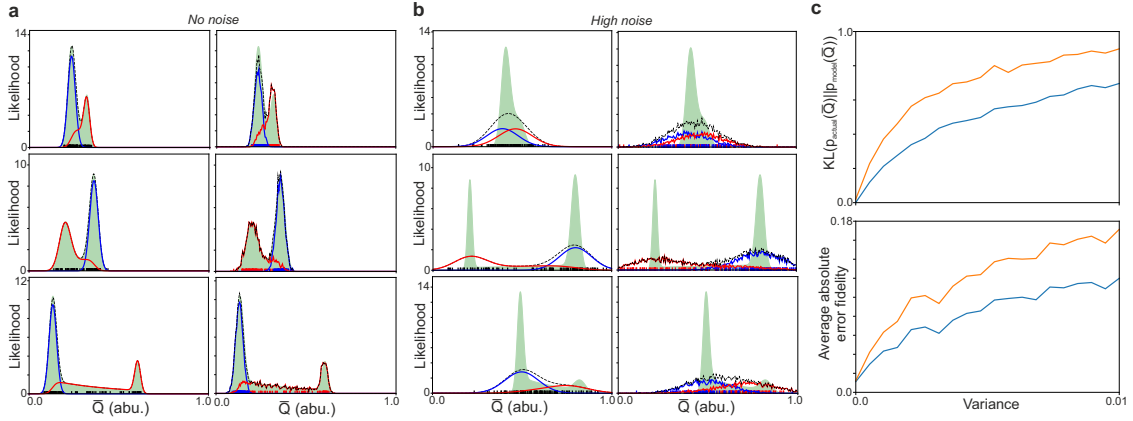


Figure 8.5: Testing on a physics model under noise **a** and **b** Trained sensor response data for randomly selected physics model parameters. Green shaded region corresponds to ground truth, black dashed line corresponds to trained model, blue and red solid lines represent model components, and ticks represent subset of training data. **a** displays trained models with no noisy observations. Left column displays models trained using EM method and right column displays models extracted using traditional supervised histogram method $N_{\text{bins}} = 100$. **b** displays trained models with noisy observations Variance = 0.0079. Left column displays models trained using EM method and right column displays models extracted using traditional supervised histogram method $N_{\text{bins}} = 100$. **c** displays training performance with respect to noise. Solid orange line represents performance of histogram method, solid blue line represents performance of EM method, both averaged over 300 different random physics models for each variance value.

and computes the absolute error between the two. Both these measures are averaged across 300 randomly generated ground truths and plotted as a function of variance. It is seen in Fig 8.5c that for all values of variance the EM method outperforms the histogram method in both measures of performance. The gap in performance is observed to grow as a function of variance in the added Gaussian noise.

8.2.4 Readout detection via model selection

A fit for $p_{EM}(\bar{Q}|\theta)$ can be obtained for any set of measurements regardless of whether they correspond to the states that we want to probe. In Fig. 8.4b the distribution of measurements is clearly uni-modal, yet training still attempts to fit the data. A different model would be more appropriate. To distinguish between sets of measurements that correspond to different states being probed and those that do not, I utilise a model selection approach. For each set of measurements

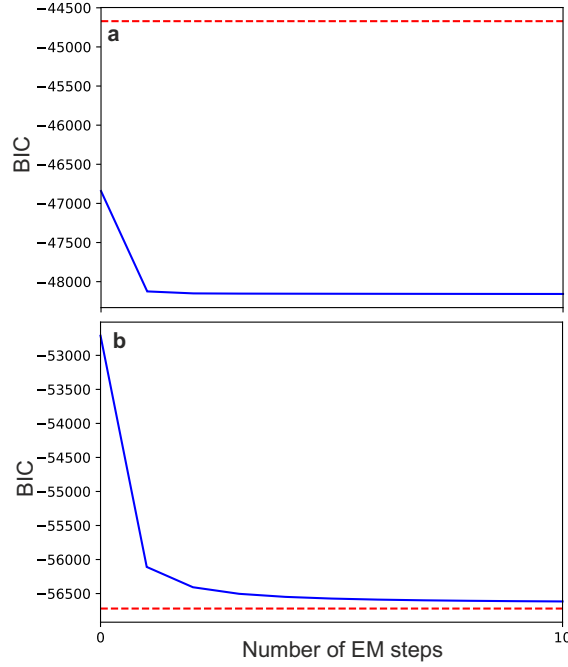


Figure 8.6: Plot of BIC as a function of EM steps **a** BIC_{EM} and BIC_{alt} for trained model in Fig 8.4a. **b** BIC_{EM} and BIC_{alt} for trained model in Fig 8.4b. Red dashed line corresponds to BIC_{alt} , solid blue line corresponds to BIC_{EM}

$\{\bar{q}_i\}_{i=1,\dots,N}$ an alternate model is also trained. The alternate model,

$$p_{\text{alt}}(\bar{Q}|\theta_{\text{alt}}) = N(\mu_a, \sigma_a^2), \quad (8.13)$$

is a single component Gaussian where $\theta_{\text{alt}} = \{\mu_a, \sigma_a\}$. This model captures cases for which loading fails or the measure point is incorrectly chosen. To select between models $p_{\text{alt}}(\bar{Q}|\theta_{\text{alt}})$ and $p_{\text{EM}}(\bar{Q}|\theta)$, I use a Bayesian information criterion [103] (BIC). This criterion allows us to rank the model performances and it is defined as:

$$\text{BIC}_{\text{model}} = k_{\text{model}} \ln N - 2 \ln L_{\text{model}}, \quad (8.14)$$

where L_{model} is the likelihood of a model given the data, and k_{model} is the number of parameters fit by EM. Computing the BIC for both models allows us to compute the evidence,

$$\text{Evidence} = \text{BIC}_{\text{EM}} - \text{BIC}_{\text{alt}}. \quad (8.15)$$

An evidence greater than zero suggests that $p_{\text{EM}}(\bar{Q}|\theta)$ should be selected to explain the data. This condition allows the detection/classification of operating gate voltage

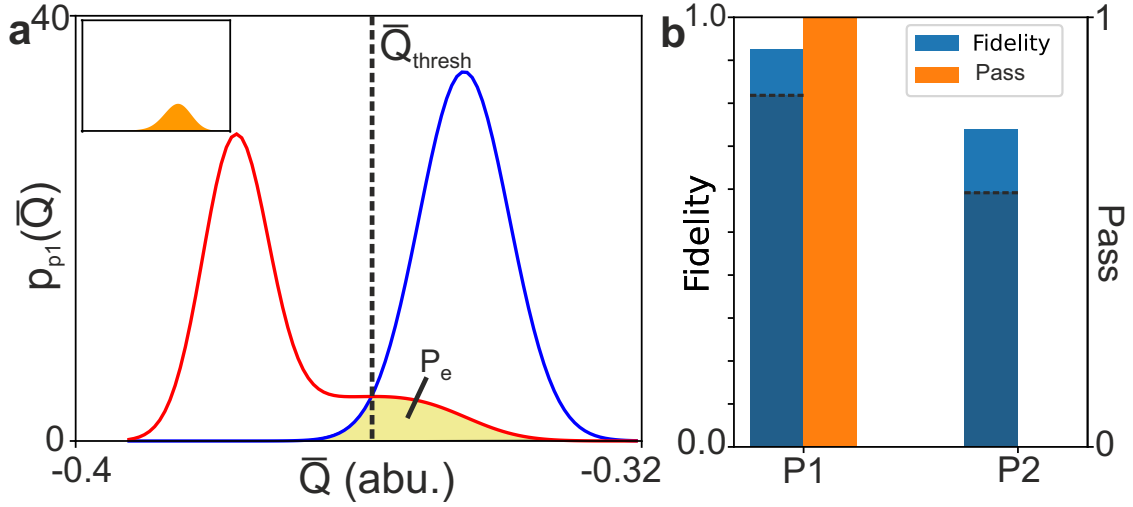


Figure 8.7: Optimal classification and fidelity. **a** Trained sensor response density components $p(\bar{Q}|S = s)$ and $p(\bar{Q}|S = t)$ for gate voltage location P1, with optimal classification threshold $\bar{Q}_{\text{thresh}}$ indicated with a black dashed line. Irreducible Bayes error P_e is highlighted in yellow and the area corresponding to the lower bound fidelity is plotted in the inset (axes are the same as for the main panel). **b** Fidelity, lower bound, and Pass for trained models from Fig. 8.4 at gate voltage locations P1 and P2.

locations that can be used for spin readout, such as P1. I use a pass fail criterion based on this idea which is defined as:

$$\text{Pass} = \begin{cases} 1 & \text{Evidence} \geq 0 \\ 0 & \text{Evidence} < 0 \end{cases}. \quad (8.16)$$

In Fig 8.6 I plot the evolution of the BIC criterion with respect to the number of EM steps.

8.2.5 Distribution overlap and optimal Bayes classification

The readout fidelity is a measure of the ability to distinguish two different quantum states. A fidelity of 1.0 indicates the two states are distinct, and can be classified perfectly with no information loss. Conversely, a readout fidelity of 0.0 indicates states that are indistinguishable from one another.

The fidelity measures described in the literature [23, 104] aim to quantify the probability of miss-classification using a given classification model. I will refer to this probability of miss-classification as P_e . After training a parametric density with the approach specified in section 8.2.2, it can be used to classify new measurements. The optimal strategy to minimise errors (assuming a uniform

prior) is to select the hypothesis with the highest likelihood. This strategy can be employed using the threshold $\bar{Q}_{\text{thresh}}$ displayed for the P1 case in Fig. 8.7a. If distributions $p(\bar{Q}|S = s)$ and $p(\bar{Q}|S = t)$ overlap, then there exists an irreducible classification referred to as the Bayes error,

$$P_e = 1 - \int \max \left\{ \frac{1}{2}p(\bar{Q}|S = s), \frac{1}{2}p(\bar{Q}|S = t) \right\} d\bar{Q}. \quad (8.17)$$

The optimal fidelity is thus

$$Fidelity \equiv 1 - P_e. \quad (8.18)$$

This quantity has no analytic form for almost all distributions but can be computed numerically.

Another useful approach to dealing with Bayes error is to place bounds on it [105]. With this in mind, an upper bound can be set on P_e and, subsequently, a lower bound can be set on the *Fidelity*,

$$Fidelity \geq 1 - \frac{1}{2}BC[p(\bar{Q}|S = s), p(\bar{Q}|S = t)], \quad (8.19)$$

where BC is the Bhattacharyya coefficient equal to

$$BC[p(\bar{Q}|S = s), p(\bar{Q}|S = t)] = \int \sqrt{p(\bar{Q}|S = s)p(\bar{Q}|S = t)} d\bar{Q}. \quad (8.20)$$

This form is intuitive, since the Bhattacharyya coefficient is a f-divergence that measures the similarity between probability distributions by considering their overlap. Equation 8.19 can be used as a surrogate objective. In Fig. 8.7a the area associated with irreducible error P_e is highlighted, and the area of the integrand associated with the lower bound is shown in the inset.

I numerically compute the *Fidelity* and the lower bound fidelity using Monte Carlo integration. Importance sampling from $p(\bar{Q}|\theta)$ is utilised to reduce the variance of the estimates. It is found that, at gate voltage location P1, *Fidelity* $\approx$ 0.924 with a lower bound of *Fidelity* $\geq$ 0.818.

8.2.6 Detection of switches

A uni-modal time varying distribution can appear to be multi-modal due to charge switches, for example. When considering the density of $\bar{Q}$, all temporal correlation between measurements $\{\bar{q}_i\}_{i=1,\dots,N}$ is lost. Switching events will pass the model selection test defined in section 8.2.4, as they will be well explained by $p_{EM}(\bar{Q}|\theta)$. They will also be assigned a desirable fidelity using the criterion in section 8.2.5. To improve the reliability of this methodology, I detail a method for detecting such switching events.

This method uses the model trained in section 8.2.2 to classify measurements $\{\bar{q}_i\}_{i=1,\dots,N}$. Each measurement is assigned a class membership $\{\bar{z}_i\}_{i=1,\dots,N}$. Because there are two classes that are randomly initialised, these should be drawn uniformly from a Bernoulli distribution,

$$p(\bar{z}|p_S) = p_S^{\bar{z}}(1 - p_S)^{1-\bar{z}}, \quad (8.21)$$

where

$$\bar{z} = \begin{cases} 1 & S = s \\ 0 & S = t \end{cases}. \quad (8.22)$$

I can check for abnormalities in the correlation of a distribution by considering how many times during the set of N measurements $\{\bar{z}_i\}_{i=1,\dots,N}$ the condition $\bar{z}_{i-1} \neq \bar{z}_i$ is true. I call these switching events 'flip events' and refer to them with the random variable D_i . The probability of a flip event is also distributed according to a Bernoulli distribution,

$$p(D_i|p_S) = \begin{cases} 2p_S(1 - p_S) & D_i = 1 \\ 2p_S^2 - 2p_S + 1 & D_i = 0 \end{cases}. \quad (8.23)$$

I model the number of flip events observed with a random variable $N_S = \sum_{j=1,\dots,N-1} D_j$. The sum of several Bernoulli trials of equal probability is distributed according to a binomial distribution which, for large N , approaches a normal distribution, hence:

$$p(N_S) = B[N - 1, 2p_S(1 - p_S)] \approx \mathcal{N}(\mu_{p_S}, \sigma_{p_D}), \quad (8.24)$$

where

$$\mu_{p_s} = (N - 1)2p_s(1 - p_s), \quad \sigma_{p_s} = (N - 1)4p_s(1 - p_s)(1 - p_s(1 - p_s)). \quad (8.25)$$

This provides us with a statistical model of how many times classification labels are expected to flip. The value p_s can be estimated from the trained model by Monte Carlo integration.

When switch events are present in measurements, the components of the model from section 8.2.1 capture this time varying behavior instead of the randomly initialised spin state. Flip events are less frequent in these circumstances. To detect time varying behavior, I use the CDF of Eq. 8.24 offset by $m = 3$ times the standard deviation,

$$\text{Flip} = P(n_s > N_{S,\text{worst}} | \mu_{p_s} - m\sigma_{p_s}, \sigma_{p_s}). \quad (8.26)$$

This offset is set so that, when N_S takes expected values, CDF is ~ 1.0 .

8.3 Complete methodology for evaluating readout fidelity

I have now introduced all components required to evaluate readout fidelities. I now combine them into a complete methodology following these steps:

- 1) use a random unload, load, measure pulse sequence to acquire N measurements $\{\bar{q}_i\}_{i=1,\dots,N}$,
- 2) train a parametric density on the measurements to produce a model $p_{EM}(\bar{Q}|\theta)$ (alternative models, such as $p_{\text{alt}}(\bar{Q}|\theta_{\text{alt}})$, should also be trained on the measurements),
- 3) compute BIC for all models and compute Pass,
- 4) compute Bayes error *Fidelity* or lower bound,
- 5) compute Flip,
- 6) compute the full fidelity score defined as

$$\text{Score} = \text{Fidelity} \times \text{Pass} \times \text{Flip}. \quad (8.27)$$

This approach is fast and effective. For a given set of experimental parameters, it can be evaluated in under 1.0 s. The major bottlenecks are the time for data acquisition and the time for training. Sampling $N = 8000$ measurements takes ~ 100 ms, and training $p_{EM}(\bar{Q}|\theta)$ and $p_{\text{alt}}(\bar{Q}|\theta_{\text{alt}})$ takes ~ 500 ms.

8.4 Optimisation of readout fidelity

Now that I have introduced a methodology for quickly evaluating readout fidelities, I will demonstrate its application to select optimal experimental parameters.

8.4.1 Optimising the averaging time

Long averaging times τ allow for better signal-to-noise ratios. However, there is a limit on how long averaging can take, since triplets might relax and appear to be indistinguishable from singlets. The time scale of this relaxation process is given by $\lambda_{T_1} = \frac{1}{T_1}$. Short T_1 times, such as those measured in GaAs devices, require short averaging times.

To optimise the averaging time, I first apply the pulse sequence displayed in Fig. 8.2b at gate voltage location P1. The digitiser is set to acquire measurements during stage M for $49.92 \mu\text{s}$, at a rate of one measurement every 10 ns. This sequence is repeated 8000 times, resulting in an array of 4992×8000 measurements.

The averaging time can then be optimised by manipulating the measurement array. By selecting the first L measurements of the array and averaging all selected measurements along the first axis, an averaging time of $\tau = L/100 \mu\text{s}$ can be considered. In this way, I produce a set of 8000 measurements which can then be treated as $\{\bar{q}_i\}_{i=1,\dots,8000}$ and subjected to the methodology defined in section 8.3.

In Fig. 8.8a I display the evolution of trained models as a function of τ . After an initial settling period of $5 \mu\text{s}$, the variance of modes reduces as a function of τ , but the triplet mode becomes less prominent. By applying the score function as a function of τ , I observe that the *Fidelity* quickly rises in the first $10 \mu\text{s}$ and then falls slowly. Both the *Fidelity* and the lower bound metric perform similarly and predict the similar optima of $10.9 \mu\text{s}$ and $12.3 \mu\text{s}$. respectively. This results

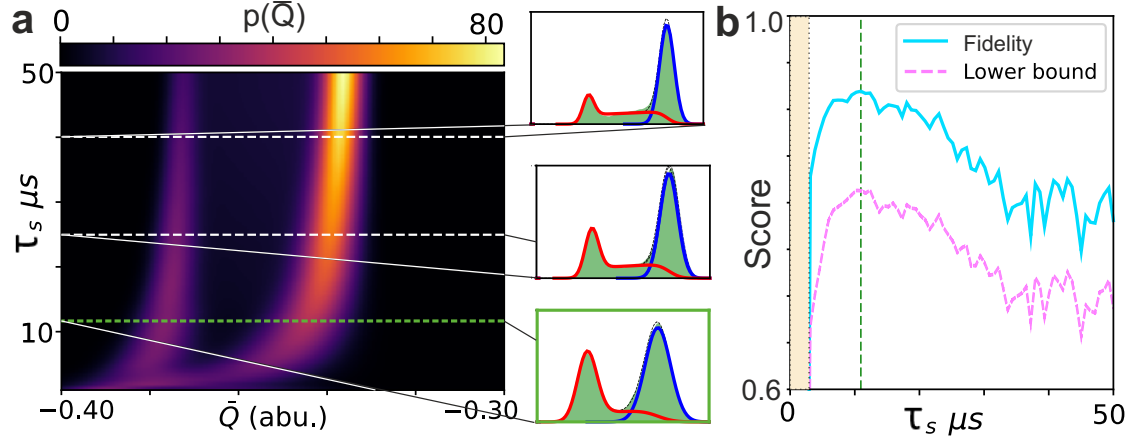


Figure 8.8: Optimising the averaging time. **a** Trained sensor response densities $p(\bar{Q}|\theta)$ as a function of τ . Example of corresponding models are displayed to the right for the optimal averaging time $11.6 \mu s$ (as predicted by *Fidelity*), and other averaging times; $25.1 \mu s$, and $40.1 \mu s$. **b** Fidelity score as a function of τ . The optimal value, selected using *Fidelity*, is indicated with a dashed green line. The shaded yellow area corresponds to values of τ where the other components of the score (Pass, Flip or both Pass and Flip) return low scores.

suggest that the lower bound can be a good surrogate for the *Fidelity* when the Bayes error is too hard to compute.

8.5 Discussion

I introduced a methodology for estimating the fidelity of spin readout. I demonstrated how a load and measure sequence can be distilled into a parametric probability density, and I showed how this density can be trained directly from measurements. Such training does not require labelled observations. The spin readout performance can then be captured using model selection. The effect of charge switches can be detected by considering the correlation of random observations. I showed that the readout fidelity can be quantified from the models produced using Bayes error. This approach is shown to be fast and effective.

I then demonstrate that this methodology can be applied to achieve an improved readout performance. Because of the relatively high readout fidelity that I obtained and the position of the gate voltage location P1 with respect to the inter-dot transition, it is possible that I might have observed latched readout [106]. However, the observed fidelity is still far from the level required for fault tolerant error

correction. Multiple avenues can be explored to realise this. The signal to noise ratio of the readout could be increased by improving the tuning of the charge sensor, improving the tuning of the reflectometry, or by amplifying the signal with a low noise amplifier; an order of magnitude improvement in signal to noise ratio would be required. The relaxation time could be increased by moving to an alternative material system or by adjusting device tuning [107]; an order of magnitude improvement in T_1 would be required.

Finally, I optimised the averaging time via a simple grid scan. This work is ongoing but these promising results indicate the potential of this approach.

9

Conclusion

I reflect on the core question this thesis addresses; ‘Can we develop general purpose, automated strategies for quantum device control that work in real devices?’. Put simply, we can. This thesis has demonstrated that this is possible by identifying multiple novel algorithms and by taking every opportunity to run them on real devices. Of course, this question was not answered in one step. To arrive at this conclusion a number of challenges had to be addressed individually, for example, the ability to perform efficient measurements, automatic tuning, and device characterisation. I will now briefly summarise the journey I went on to address these challenges. Although the core question of this thesis has been answered, there is still much work left to be done, so I will also discuss future directions of this work that I believe should be built upon.

Efficient measurements of quantum devices - the outcome of this work contributes a framework that leverages active learning to improve on the common grid scanning method. It is shown to perform well for multiple different measurements for which grid scans see use, demonstrating its broad applicability. This work also develops the idea of using deep generative models to reconstruct unseen measurements. Finally, this work provides a methodology for bench-marking the performance of the resulting framework. This method can be used when training data is available under the limitation that the distribution of the training data is similar to that of

measurements that it is applied to. Another significant limitation is the approach is limited to making measurements in a two dimensional experimental design space, future work should try to generalise the approach to experimental design spaces with an arbitrary number of dimensions. This could be achieved using a physics model for G and an alternate strategy for optimising the acquisition function.

Automatic coarse tuning of quantum devices - this work provides insights on the structure of the gate voltage space defined by semiconductor quantum devices, and develops the concept of device hypersurface. The algorithm that resulted from this work demonstrates human-level performance (and often the performance is better than that of humans) in the task of tuning different semiconductor devices based on different material systems. The algorithm is a combination of multiple ideas. It uses the idea of searching for transport features near to the hypersurface instead of searching the full gate voltage space. Also, the algorithm treats the search as a multi armed bandit problem. The algorithm was extensively tested to validate all design choices. I developed a methodology for bench-marking the performance of this algorithm or any other search based method. Currently this method is limited to making measurements in transport current on small scale devices. Future work should focus on making measurements in charge sensing and adapting the approach to larger scale devices like arrays of many quantum dots.

Methods for characterising device variability - this work further builds on the concepts developed for the automatic tuning algorithm by showing that they can be reapplied to other challenges. I developed the idea of applying point set registration to measurements of the hypersurface in order to characterise the variability between devices. I was also able to define a volume in parameter space in which a SET is operable and to quantify its size and structure. This method inherits limitations from the automatic coarse tuning algorithm.

Automatic fine tuning of a quantum device - this work developed a methodology for scoring measurements using a distance metric and a small number of target measurements. This algorithm was demonstrated experimentally by using a simple optimiser to improve the quality of double quantum dot transport features, in

particular, bias triangles. The optimiser used is simplistic and susceptible to getting stuck in local minima. This could easily be improved by using an alternate optimiser such as Bayesian optimisation.

Automatic tuning and optimisation of spin qubit readout - I developed a methodology for evaluating qubit readout fidelities. The main outcome of this work is the unsupervised training of a parametric readout density that can be used to optimise the readout fidelities of singlet triplet qubits. This methodology is verified by demonstrating that it can be used to select experimental parameters that improve such readout fidelities. This method is limited to simple optimisation routines based off grid scanning. Future work should focus on intelligently optimising experimental parameters according to measured readout fidelity. This could be achieved through the use of techniques like Bayesian optimisation.

This work has made multiple contributions to the use of machine learning for quantum device control. The clearest contribution of this work in the context of the current literature is the focus on using machine learning methods for making decisions and on the general applicability of the algorithms. In prior work, decisions relied on simple optimisation methods and heuristics. This thesis identifies two decision frameworks that fit naturally into the problem of quantum device tuning; active learning (Chapter 3) and multi-armed bandits (Chapter 4, Chapter 5, Chapter 6). These frameworks, being well established, allow for the identification of powerful abstractions that contribute to the algorithm's general applicability. Of particular surprise is the effectiveness and adaptability of the algorithm presented in Chapter 4, which was adapted to solve different challenges in Chapter 6. At time of writing this, the algorithm is publicly available under the name *AutoDot* and sees regular use in other laboratories around the world.

With this in mind, I now reflect on the future directions of this work in the field of semiconductor spin qubits. It takes years to train an experienced experimentalist, and even they can spend weeks or even months tuning a device into operation. A more efficient method is definitely needed to enable the development of a scalable quantum computer. As is evidenced by this thesis, the automation of spin qubit

control is possible, but it is a challenging task; it requires the development of bespoke machine learning approaches. A secondary issue that is becoming apparent is the stabilisation of devices - once a device has been tuned into an operable regime its operability must be maintained. Devices often need daily re-tuning. Future work should investigate the applicability of machine learning approaches to this task.

The algorithms and methods developed in this thesis provide a major leap towards developing general purpose automated strategies for quantum device control that work in real devices.

Appendices

.1 CVAE architecture

The structure of the of the convolutional neural networks used in the CVAE of chapter 3 is described in Tables 1, 2, and 3. The exponential linear unit is applied after each layer except the final layer of the encoder, decoder, and the discriminator. Batch normalisation is applied after all convolution layers except as separately described. The first and second number in parentheses of the layer names indicate kernel size and stride.

Layer name	output size
Initial	128x128x1
Conv(5,2)	64x64x64
Max pooling(3,2)	32x32x64
Conv(3,1)	32x32x128
Conv(3,2)	16x16x128
Conv(3,1)	16x16x128
Conv(3,2)	8x8x128
Conv(3,1)	8x8x128
Conv(3,2)	4x4x128
Fully connected	200

Table 1: Specification of the encoder.

Layer name	output size
Initial	1x1x(100+64)
Conv'(3,2)	2x2x1,024
Conv(3,1)	2x2x1,024
Conv'(3,2)	4x4x512
Conv(3,1)	4x4x512
Conv'(3,2)	8x8x256
Conv(3,1)	8x8x256
Conv'(3,2)	16x16x128
Conv(3,1)	16x16x128
Conv'(3,2)	32x32x64
Conv(3,1)	32x32x64
Conv'(3,2)	64x64x64
Conv(3,1)	64x64x64
Conv'(3,2)	128x128x32
Conv(3,1)	128x128x32
Conv(1,1,tanh)	128x128x1

Table 2: Specification of the decoder.

Layer name	output size	remark
Initial	128x128x1	
Conv(5,2)	64x64x64	context loss
Conv(3,1)	64x64x128	No BN
Conv(3,2)	32x32x128	context loss
Conv(3,1)	32x32x128	No BN
Conv(3,2)	16x16x128	context loss
Conv(3,1)	16x16x128	No BN
Conv(3,2)	8x8x128	context loss
Conv(3,1)	8x8x128	No BN
Conv(3,2)	4x4x128	context loss
Global average pooling	1x1x128	context loss
Fully connected	2	

Table 3: Specification of the discriminator. In remarks, *NO BN* indicates that batch normalisation is not applied to the layer, and *context loss* indicates that the layer is used to calculate the context loss.

.2 Applying BALD to a CVAE

I derive how the BALD criteria in equation 3.3 can be applied to the CVAE from chapter 3. For brevity I use the definition $p(.) = p(.|Y_n)$ and imply the condition x_{n+1} with the presence of y_{n+1} :

$$\mathbb{E}[KL(p_n(\mathbf{z}|y_{n+1})||p_n(\mathbf{z}))] = \int_{y_{n+1}} p_n(y_{n+1}) \int_{\mathbf{z}} p_n(\mathbf{z}|y_{n+1}) \log \left(\frac{p_n(\mathbf{z}|y_{n+1})}{p_n(\mathbf{z})} \right) d\mathbf{z} dy_{n+1} \quad (1)$$

As mentioned in section 3.5 the continuous distribution is approximated as a discrete distribution $p_n(\mathbf{z}) \approx P_n(m)$. Let n_s denote the iteration when sampling was performed, when $n = n_s$ samples are uniform so for a given m , $P_{n_s}(m) = \frac{1}{M}$. When $n > n_s$ the distribution P_n is directly updated with new measurements hence $P_n(m, n_s) = \frac{p(Y_{(n_s+1:n)}|\hat{Y}_m)}{\sum_m p(Y_{(n_s+1:n)}|\hat{Y}_m)}$. We can also approximate

$$p_n(y_{n+1}) = \int_{\mathbf{z}} p_n(y_{n+1}|\mathbf{z})p_n(\mathbf{z})d\mathbf{z} \approx \sum_m P_n(m)p(y_{n+1}|\mathbf{z}_m) \quad (2)$$

The value of $\hat{y}_m$ at $\mathbf{x}_{n+1}$ can be used for y_{n+1} . Using this equation 1 can be approximated giving

$$\sum_m P_n(m)KL(P_{n+1}(m')||P_n(m')) + c, \quad (3)$$

where c is an arbitrary constant.

.3 Noisy reconstructions

For the real time estimates of $r(n)$, the number of reconstructions is augmented up to 3,000 by adding different noise patterns that are present in typical measured current maps. For the estimation of $\tilde{r}_m(n)$, a single reconstruction $\hat{Y}_m$ is augmented to 30 noisy reconstructions:

$$\hat{Y}_{m,j,\text{SNR}}(\mathbf{x}) = \hat{Y}_m(\mathbf{x}) + \alpha_{m,j,\text{SNR}} \times \mathcal{E}_j(\mathbf{x}) \quad \text{for all } \mathbf{x} \text{ in } X,$$

where $\mathcal{E}_j = \{(\mathbf{x}, e_x^j)|\mathbf{x} \in X\}$ is a noise profile with index j , X is a set of all gate voltage pairs in a two dimensional domain, and $\alpha_{m,j,\text{SNR}}$ is a multiplier that sets the signal-to-noise ratio SNR, where the signal is $\hat{Y}_m$ and the noise is $\mathcal{E}_j$. The noise

profiles characterised correspond to 10 gate voltage ranges close to the Coulomb diamond regime and for which the current is suppressed. SNR is chosen from $\{20^2, 40^2, 80^2\}$, i.e. high, medium, and low noise. This increase in the variability of the reconstructions is needed to avoid an overconfident estimation of $r(n)$.

.4 Applying BALD to a physics model

I derive how the BALD criteria in equation 3.3 can be applied to the physics model from chapter 3. For brevity I use the definition $p(\cdot) = p(\cdot|Y_n)$ and imply the condition x_{n+1} with the presence of y_{n+1} . It can be show that:

$$\mathbb{E}_{y_{n+1}} [KL(p_n(\mathbf{z}|y_{n+1})||p_n(\mathbf{z}))] = \mathbb{E}_{\mathbf{z}} [KL(p_n(y_{n+1}|\mathbf{z})||p_n(y_{n+1}))] \quad (4)$$

Taking y_{n+1} to be the output of the model at $\mathbf{x}_{n+1}$ it follows that:

$$p(y_{n+1}|\mathbf{z} = \mathbf{z}_m) = \mathcal{N}(\hat{y}_{n+1}(\mathbf{z}_m, \mathbf{x}_{n+1}), \sigma^2), \quad (5)$$

and

$$p(y_{n+1}) \approx \frac{1}{M} \sum_m \mathcal{N}(\hat{y}_{n+1}(\mathbf{z}_m, \mathbf{x}_{n+1}), \sigma^2). \quad (6)$$

when $p(\mathbf{z})$ is again approximated by $m = 1, \dots, M$ samples from $\mathbf{z}$. Using this the integral in equation 4 can be approximated with Monte Carlo integration as samples from Gaussian distributions Gaussian mixtures.

.5 Simple tuner

The *simple tuner* is a simple methodology to tune tunnel barriers of an electrostatically defined quantum dot when the devices barrier gates are well diagonalised inputs. The algorithm is not mathematically optimal or ideally formulated. It's benefit is it's simplicity; when in a pinch this algorithm might be the most effective to apply if you believe your device exhibits a cube like hypersurface.

The algorithm operates on a set of N barrier gates $V_{\text{barriers}} = \{V_i\}_{i=1, \dots, N}$ and requires some step size δS , and starting origin $\mathbf{o}_{\text{start}}$. The algorithm iterates between characterising the pinch-off behavior of the gates in V_{barriers} and updating

the location of the origin $\mathbf{o}$ in order tune gates V_{barriers} . The exact algorithm is detailed in algorithm 1.

Algorithm 1 The *Simple tuner* algorithm

```

 $\mathcal{U} \leftarrow \mathbb{1}(|V_{\text{barriers}}|)$  {where  $\mathbb{1}$  is the identity}
 $\mathbf{o} \leftarrow \mathbf{o}_{\text{start}}$ 
while  $I_c(\mathbf{o}) > 0$  do
   $\mathbf{p} \leftarrow \{\}$ 
  for  $\mathbf{u} \in \mathcal{U}$  do
     $\mathbf{p}_i \leftarrow \text{Search}(\mathbf{u}, \mathbf{o})$  {perform a search for the hypersurface}
     $\mathbf{p} \leftarrow \mathbf{p} \cup \mathbf{p}_i$ 
  end for
   $\mathbf{u}_{\text{step}} \leftarrow \frac{\mathbf{p} - \mathbf{o}}{|\mathbf{p} - \mathbf{o}|}$ 
   $\mathbf{o} \leftarrow \mathbf{u}_{\text{step}} \delta S + \mathbf{o}$ 
end while

```

The algorithm uses the same conditions defined in section 4.3. The function *Search* can be a linear search like that defined in section 4.5.1 or a binary search linear search combination like used in section 6.2.4.

Roughly the algorithm can be thought of as a methodology for finding the hyper-corner of a hyper-cube under such ideal conditions δS actually serves to slow down the method. However, when the faces of the hyper-cube are not perfectly aligned with gate unit vectors δS aids the algorithm in converging to the corner.

.6 Gaussian process specification

For the model of $\tilde{p}(r|\mathbf{u})$ initially there is no prior knowledge, except for the maximum and minimum gate voltage bounds. Thus the range of r is: $r \in [0, r_{\text{bound}}] = [0, \sqrt{N} \times 2 \text{ V}]$, where N is the number of gate electrodes. We set the prior distribution in this range: $m_r(\mathbf{u}) = (r_{\text{bound}})/2$, $k_r(\mathbf{u}, \mathbf{u}) = (r_{\text{bound}})^2/4^2$. This prior means that r_{bound} and 0 will be two standard deviations away from $m_r(\mathbf{u})$.

The probability $\tilde{P}_{\text{peak}}(\mathbf{v})$ can be modelled, by factorising it into two components $\tilde{P}_{\text{peak}}(\mathbf{v}) = \tilde{P}_{\text{peak}|\text{valid}}(\mathbf{v})\tilde{P}_{\text{valid}}(\mathbf{v})$, where $\tilde{P}_{\text{valid}}(\mathbf{v})$ is the probability that $\mathbf{v}$ corresponds to a pinch-off location, and not to the boundary of the operable gate voltages $[0, -2] \text{ V}$.

For all Gaussian process models the Matern 5/2 kernel is used for the prior covariance. The prior covariance is defined by length scales l_q with $q = 1, \dots, N$, setting the smoothness of the model predictions. These length scales are periodically optimised to the Maximum a posterior (MAP) using gathered data and a prior gamma distribution. The gamma distribution is set to have mean 0.4, and variance 0.1^2 for $\tilde{p}(r|\mathbf{u})$; mean 500 and variance 100^2 for $\tilde{P}_{\text{peak}|\text{valid}}(\mathbf{v})$; and mean 50, and variance 20^2 for $\tilde{P}_{\text{valid}}(\mathbf{v})$.

.7 Tuning algorithm raw labels

The number of positively labeled current maps from Chapter 4 are provided below for each labeller. Table 4 contains data from section 4.9.4 and Table 5 contains data from section 4.9.6.

Experiment	Iterations	Time (hr)	L1	L2	L3
Device1, run1	500	8.2	7	8	7
Device1, run2	500	9.1	7	8	6
Device1, run3	500	8.2	5	4	6
Device1, run4	500	8.4	5	11	10
Device1, run5	500	7.7	4	6	4
Device2, run1	500	11.2	2	3	1
Device2, run2	500	11.4	1	6	6
Device2, run3	500	11.2	0	4	2
Device2, run4	500	11.4	3	9	7
Device2, run5	500	10.5	3	4	5
Device2, Grouped, run1	500	11.1	12	19	20

Table 4: Labelling result of the experiments in Section 4.9.4. It shows the number of samples classified as successes for each labeller.

Experiment	Iterations	Time (hours)	L1	L2	L3
Pure random, run1	1000	6.9	0	0	0
Pure random, run2	10000	68.9	1	2	1
Uniform surface, run1	500	11.5	0	1	1
Peak weighting, run1	500	35.3	4	4	2
Full decision, run1	500	15.7	2	6	5
Full decision, run2	459	16.2	5	3	4
Full decision + grouped gates, run1	500	14.8	7	15	11

Table 5: Labelling result of the experiments in Section 4.9.6. It shows the number of samples classified as successes for each labeller. The second run of the Full method could not reach 500 iterations, because the algorithm was interrupted by some external noise.

.8 Thompson sampling for device tuning

The objective of Thompson sampling is to select action $k \in \{1, \dots, K\}$ to maximise the expected reward, action k will be selected with probability

$$\int \mathbb{I} \left[\mathbb{E}(r|k, \theta) = \max_{k'} \mathbb{E}(r|k', \theta) \right] P(\theta|\mathcal{D}) d\theta, \quad (7)$$

Where $\mathbb{I}$ is an indicator function, $\mathcal{D} = (X_{1:i}, Y_{1:i})$, and θ parameterises the distribution. The reward r is deterministic being either 0 or 1 with $r = 1$ denoting success, in our case $\theta = \hat{r}$ which is Bernoulli distributed as the probability of success is either 1.0, or 0.0 (note this fundamentally differs from the classic Bernoulli bandit senario where the probability of success takes a value in the domain $[0.0, 1.0]$).

In practice Thompson sampling is implemented by sampling parameters θ^* from $p(\theta|\mathcal{D})$ and selecting the optimal action according to those. For Chapter 4 this means sampling $\hat{r}$ from a Gaussian process classification model $\tilde{P}_{\text{peak}}$ for some set of points in gate voltage space $\{s_k\}_{k=1,\dots,K}$ and if a value of $\hat{r} = 1.0$ then $r = 1$ conversely sampling $\hat{r} = 0.0$ then $r = 0$. A given action s_k will take on a reward of $r = 1$ with probability $\tilde{P}_{\text{peak}}(s_k)$ hence the algorithm must weight decisions by their predicted probability of success $\tilde{P}_{\text{peak}}(s_k)$.

.9 Additional stability diagrams

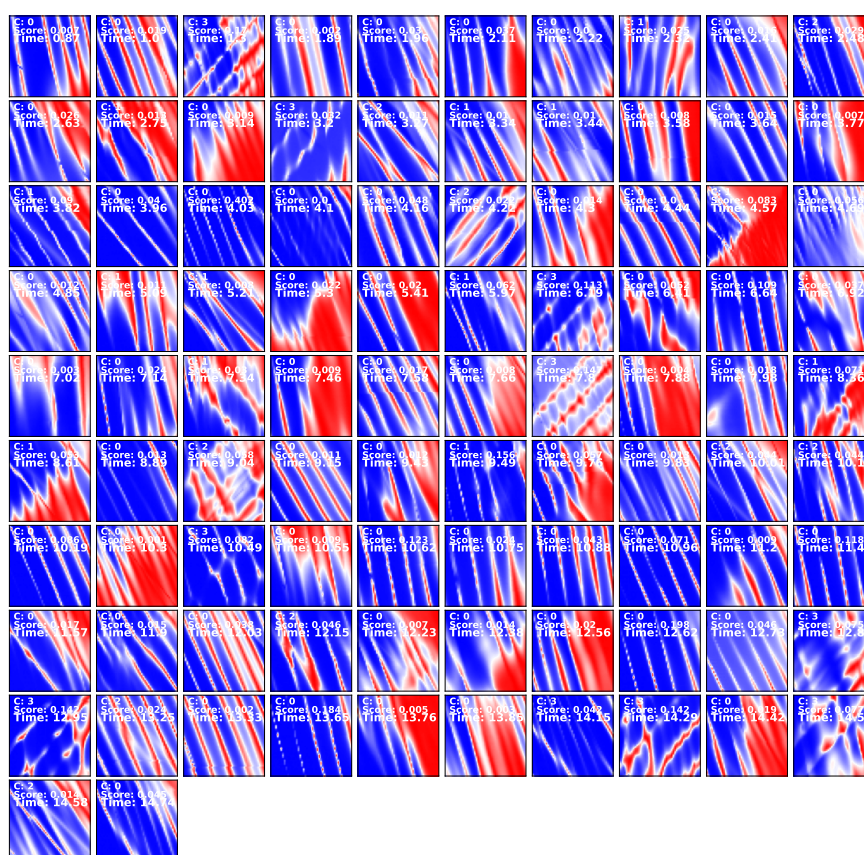


Figure 1: Device 1 run 1

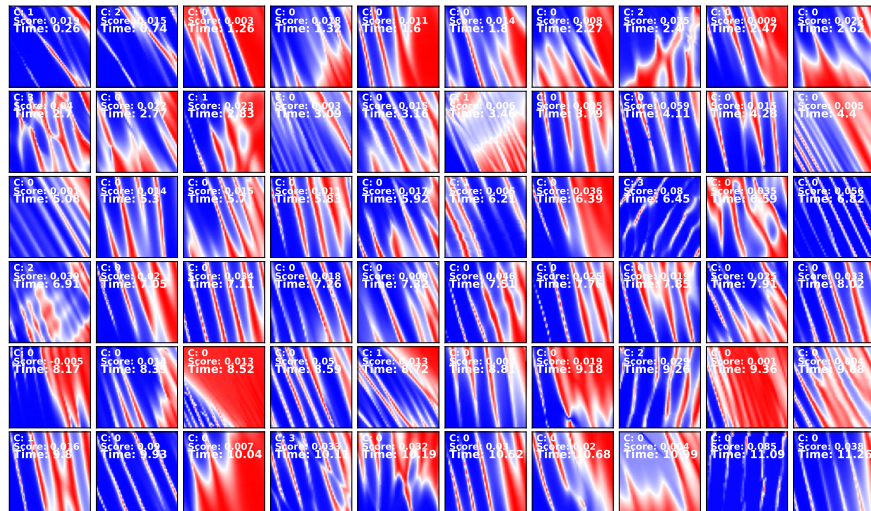


Figure 2: Device 1 run 2

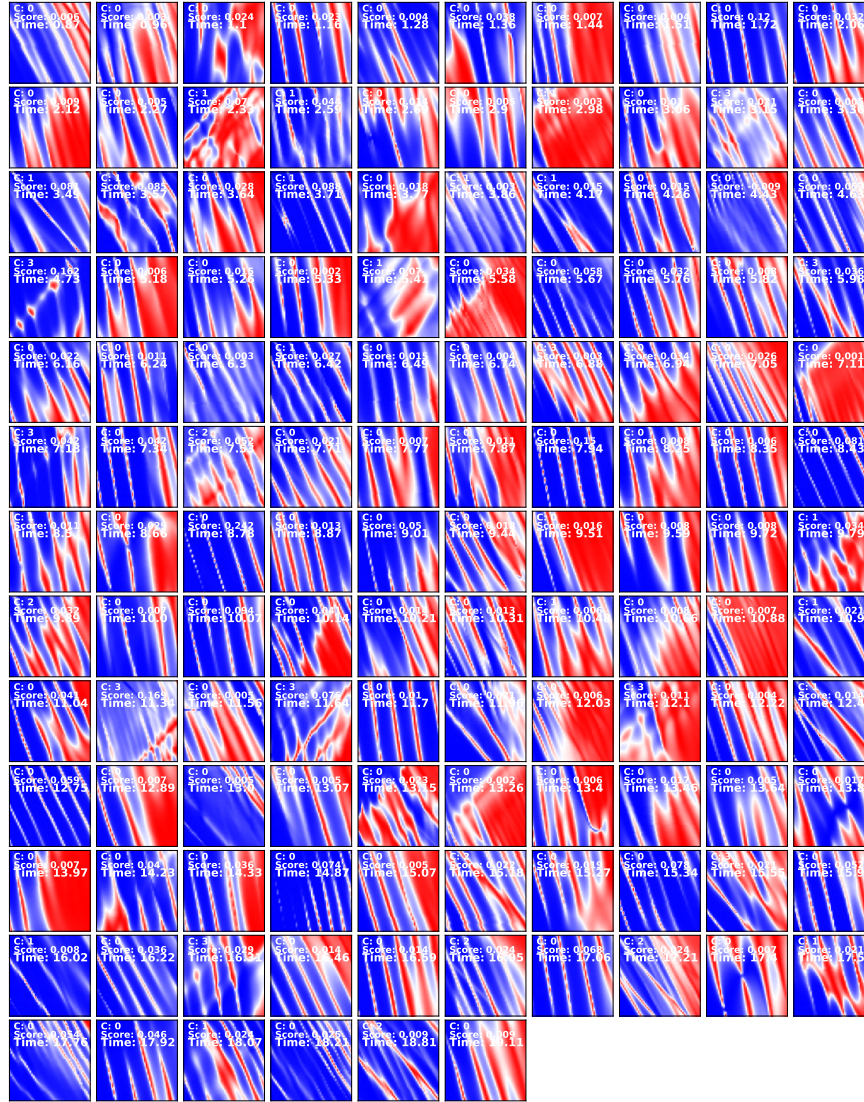


Figure 3: Device 1 run 3

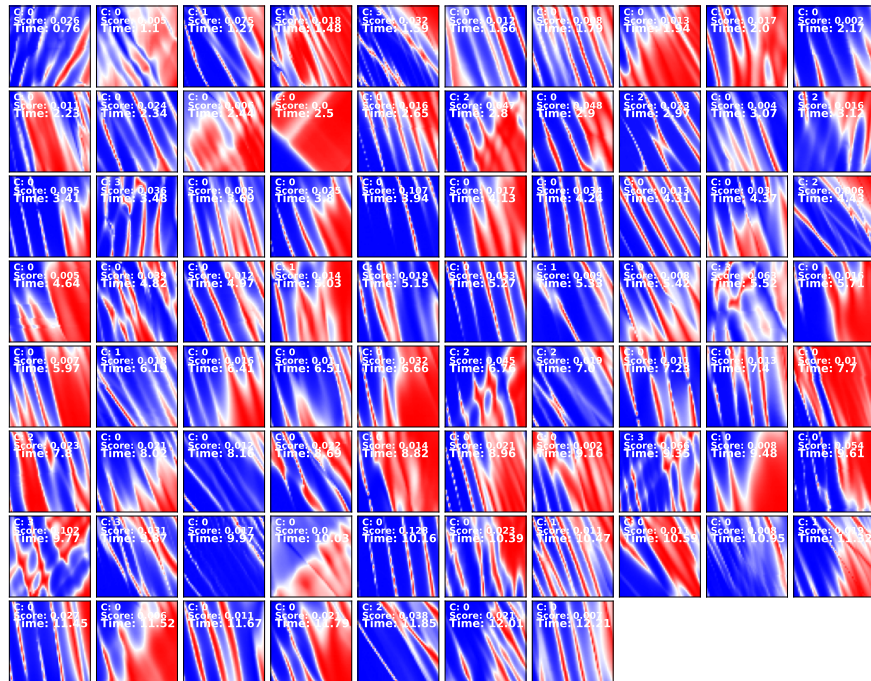


Figure 4: Device 1 run 4

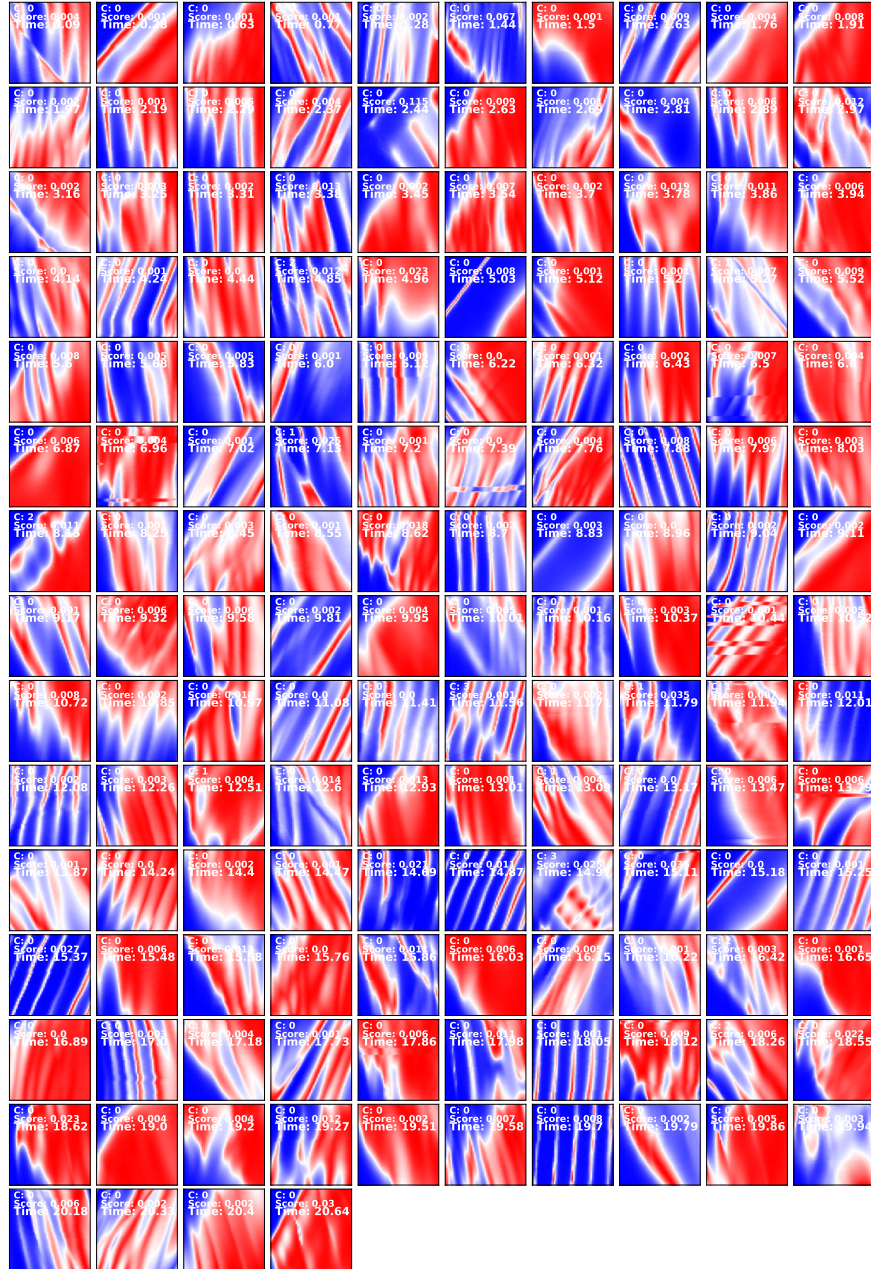


Figure 6: Device 2 run 1

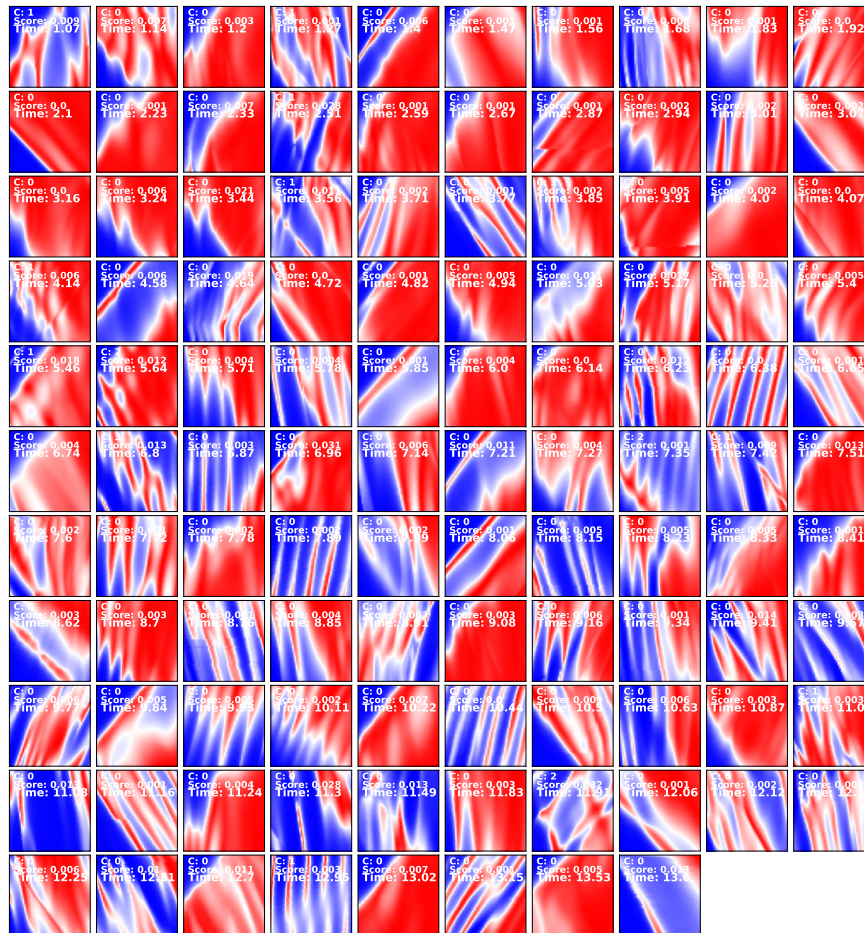


Figure 7: Device 2 run 2

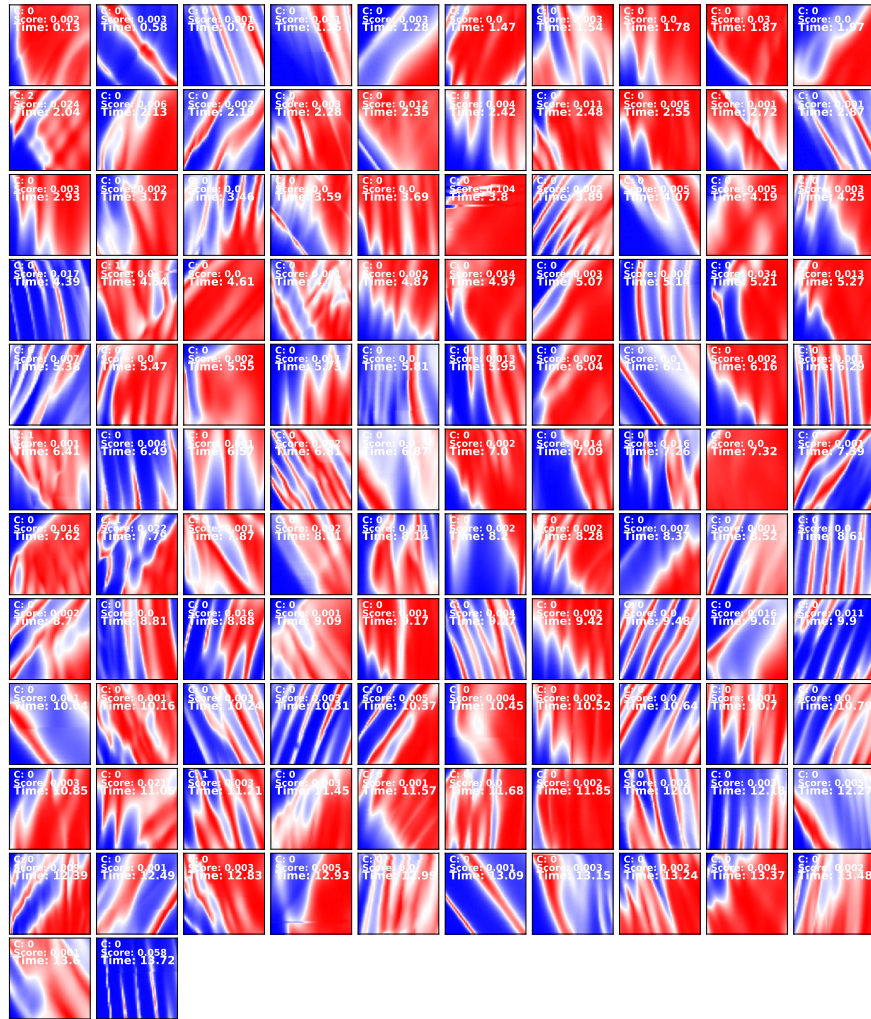


Figure 8: Device 2 run 3

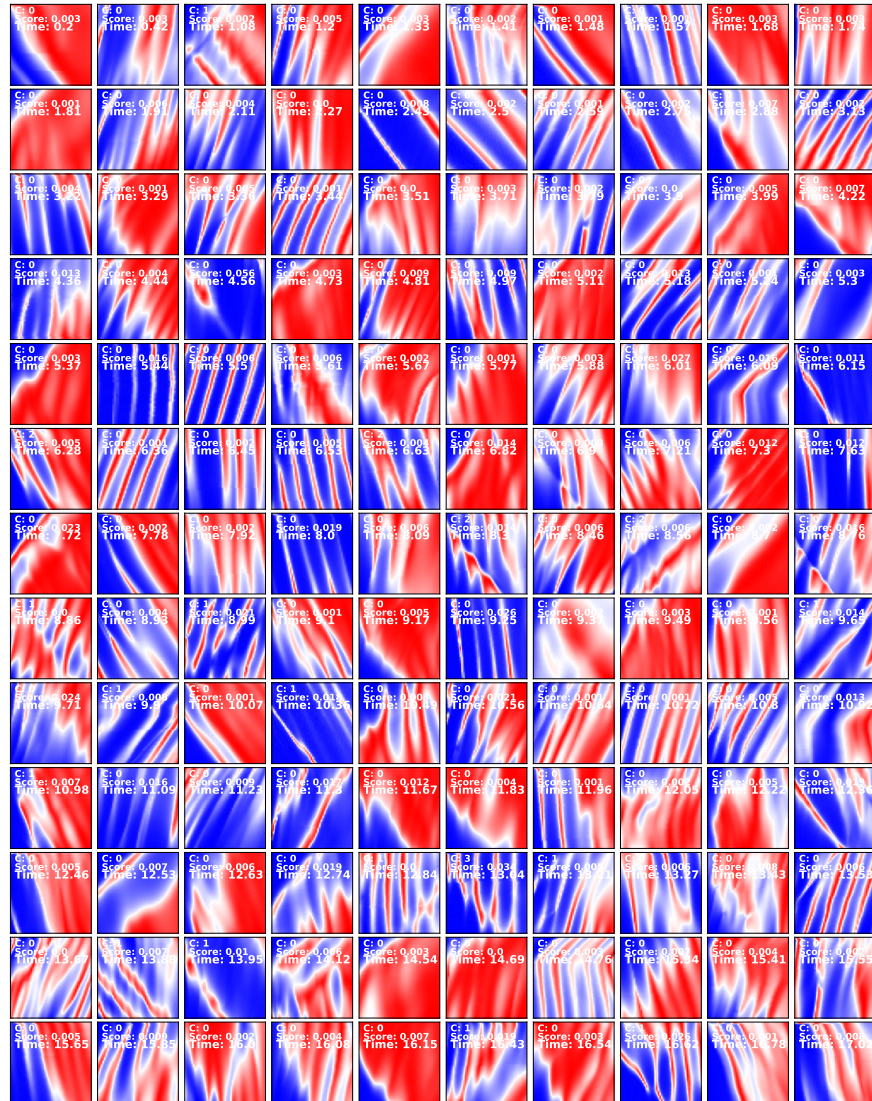


Figure 9: Device 2 run 4

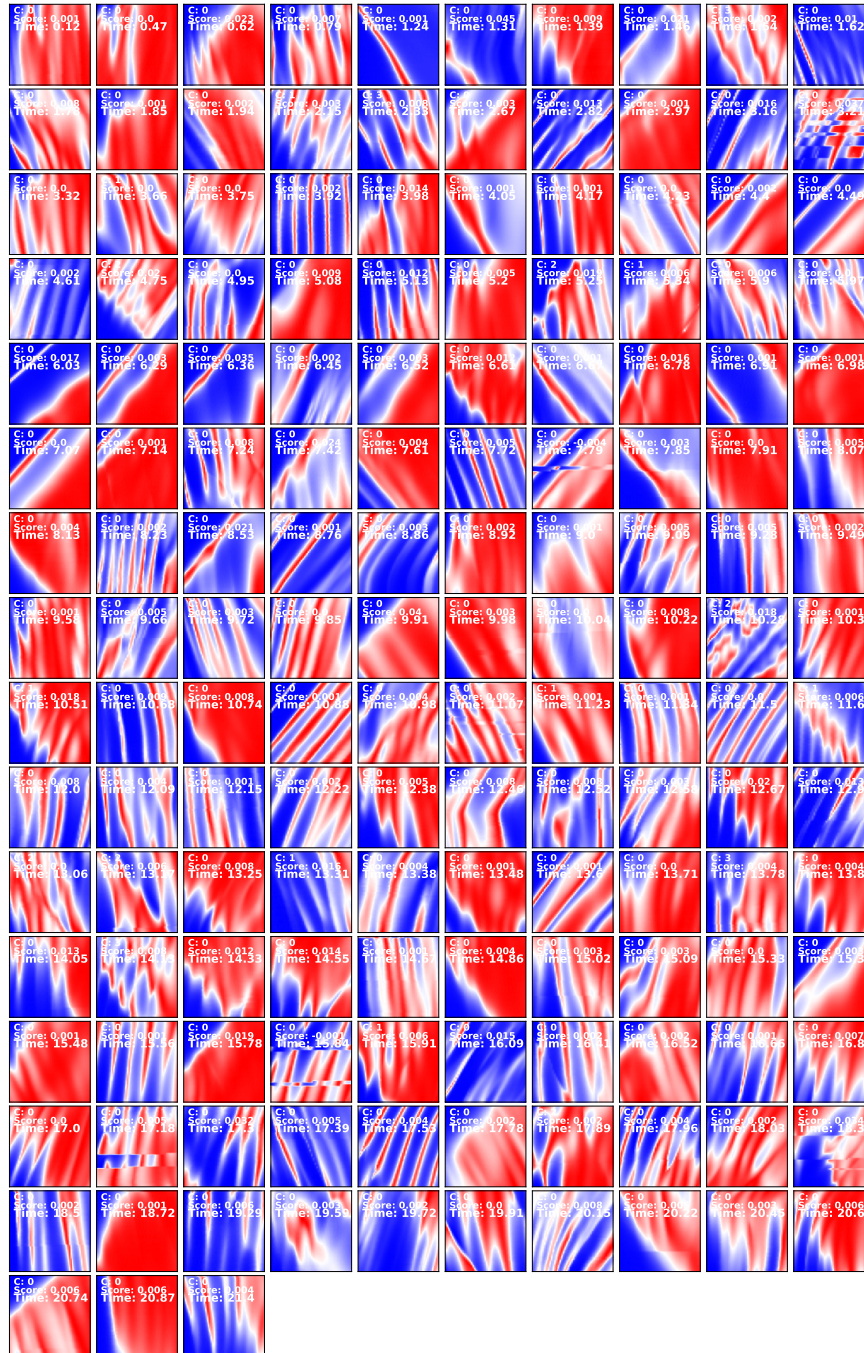


Figure 10: Device 2 run 5

.10 VAE architecture

The structure of the of the convolutional neural networks used in the VAE of chapter 7 is described in Tables 6, 7, and 8. The rectified linear unit (ReLU) is applied after each layer except those indicated. The first and second number in parentheses of the layer names indicate kernel size and stride.

Layer name	output size
Initial	32x32x1
Conv(4,2)	16x16x32
Conv(4,2)	8x8x64
Conv(4,2)	4x4x64
Fully connected	128
Fully connected	20

Table 6: Specification of the encoder from Chapter 7.

Layer name	output size
Initial	10
Fully connected	128
Fully connected	4x4x64
Conv'(4,2)	8x8x64
Conv'(4,2)	16x16x32
Conv'(4,2, Sigmoid)	32x32x1

Table 7: Specification of the decoder from Chapter 7.

Layer name	output size
Initial	10
Fully connected	1000
Fully connected	1000
Fully connected	1000
Fully connected	1000
Fully connected	2

Table 8: Specification of the discriminator from Chapter 7.

.11 Additional stability diagrams

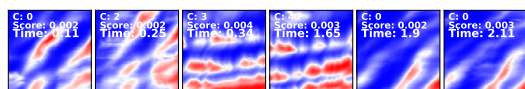


Figure 11: FinFET run 1

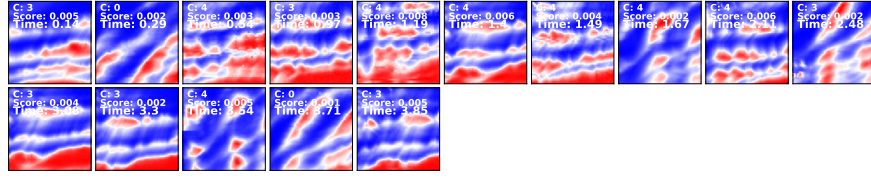


Figure 12: FinFET run 2

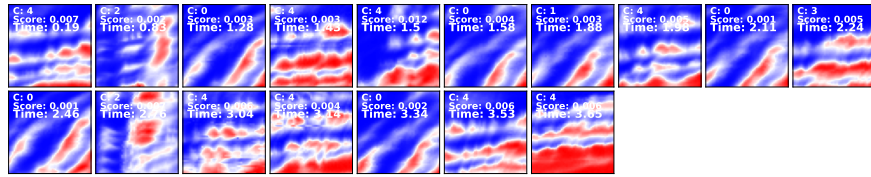


Figure 13: FinFET run 3

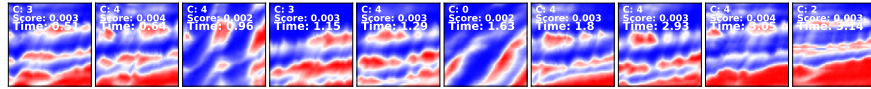


Figure 14: FinFET run 4

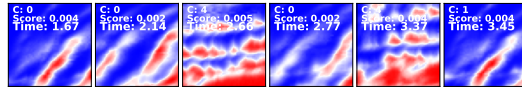


Figure 15: FinFET run 5

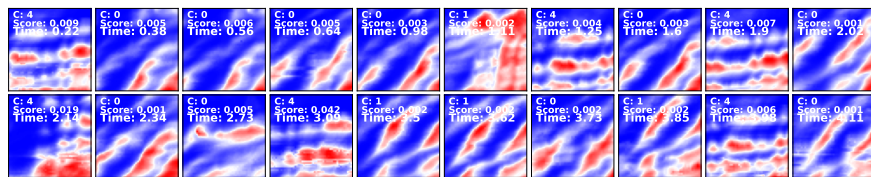


Figure 16: FinFET run 6

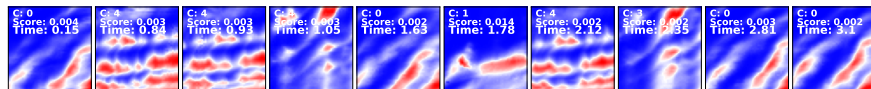


Figure 17: FinFET run 7

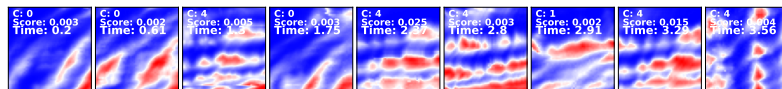


Figure 18: FinFET run 8

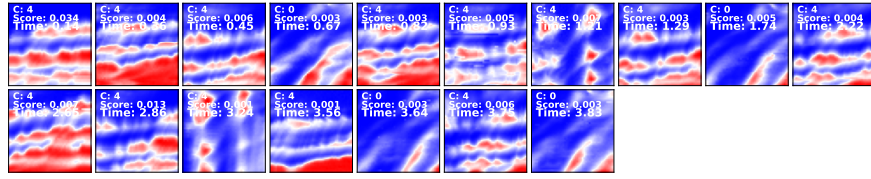


Figure 19: FinFET run 9

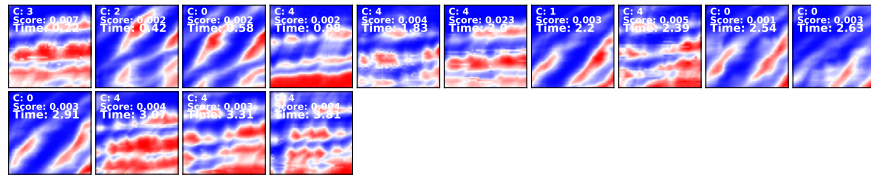


Figure 20: FinFET run 10

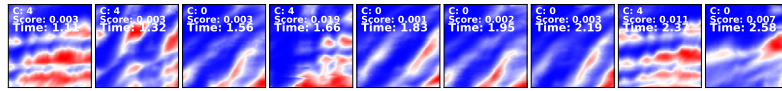


Figure 21: FinFET run 11

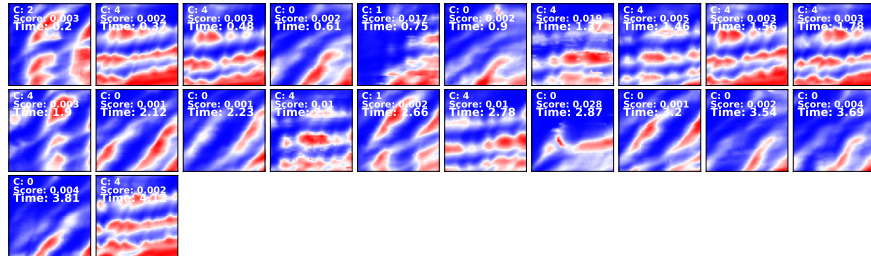


Figure 22: FinFET run 12

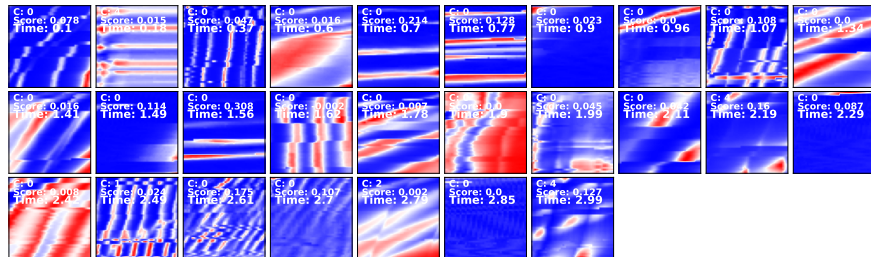


Figure 23: Heterostructure run 1

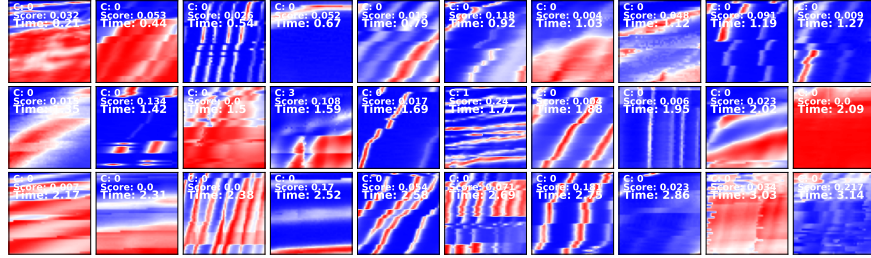


Figure 24: Heterostructure run 2

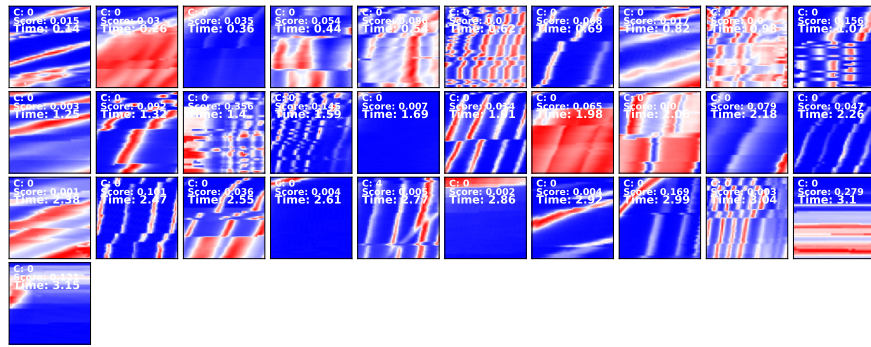


Figure 25: Heterostructure run 3

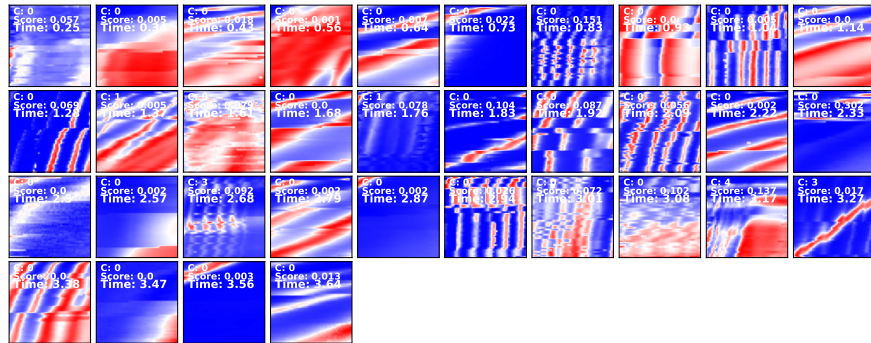


Figure 26: Heterostructure run 4

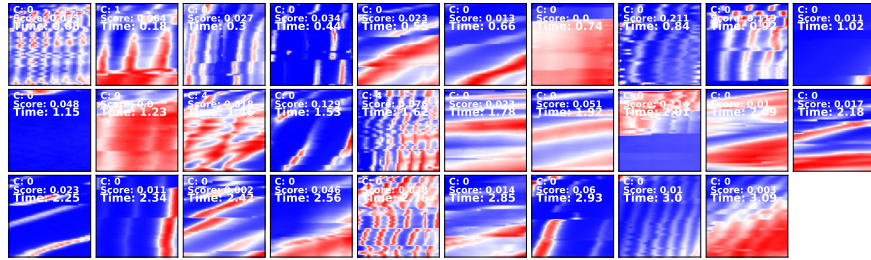


Figure 27: Heterostructure run 5

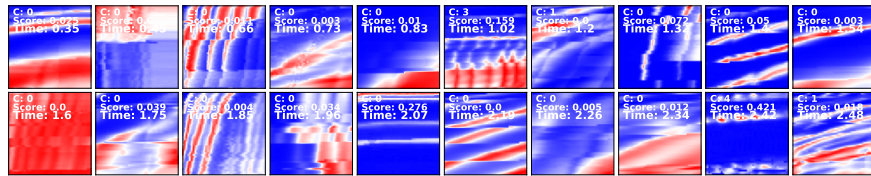


Figure 28: Heterostructure run 6

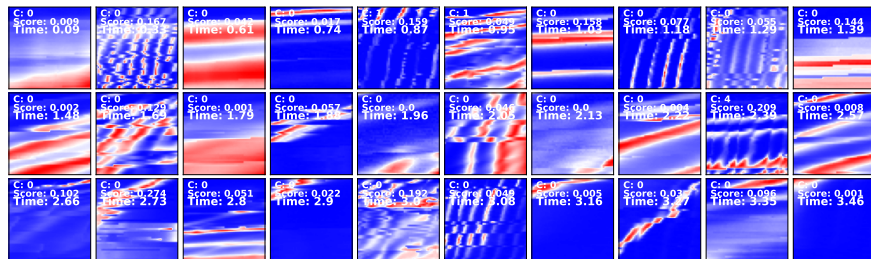


Figure 29: Heterostructure run 7

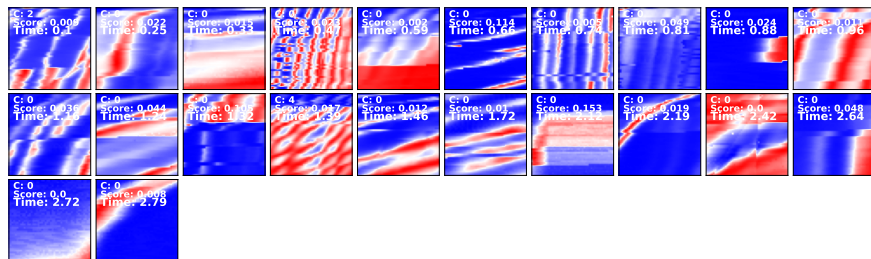


Figure 30: Heterostructure run 8

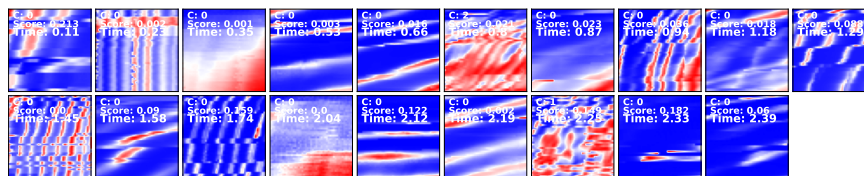


Figure 31: Heterostructure run 9

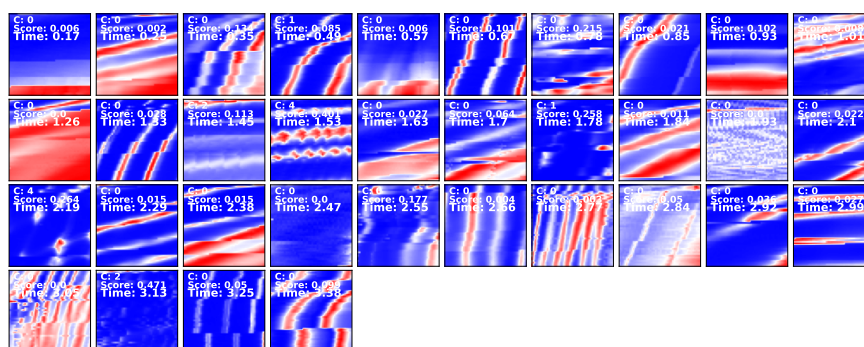


Figure 32: Heterostructure run 10

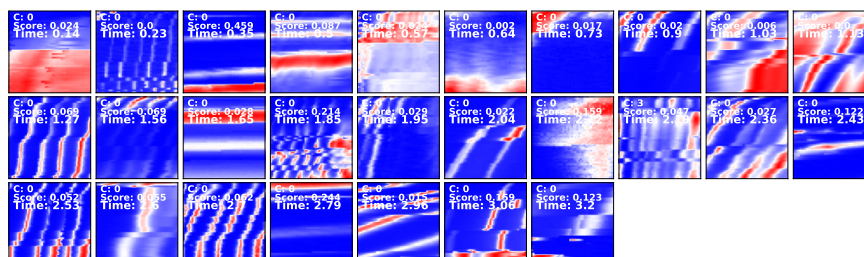


Figure 33: Heterostructure run 11

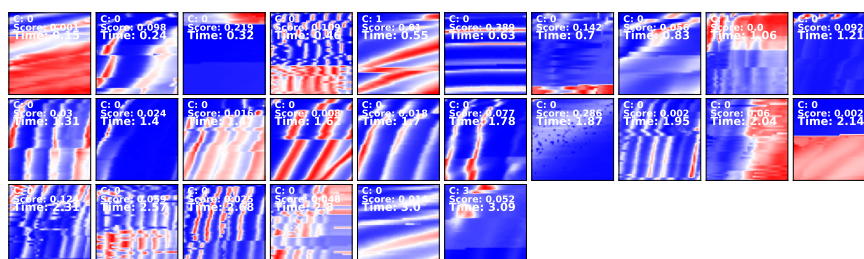


Figure 34: Heterostructure run 12

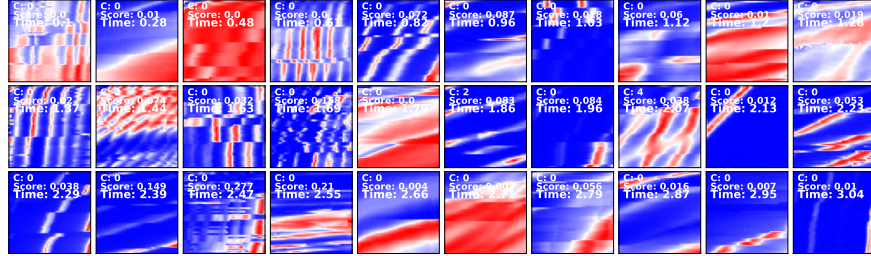


Figure 35: Heterostructure run 13

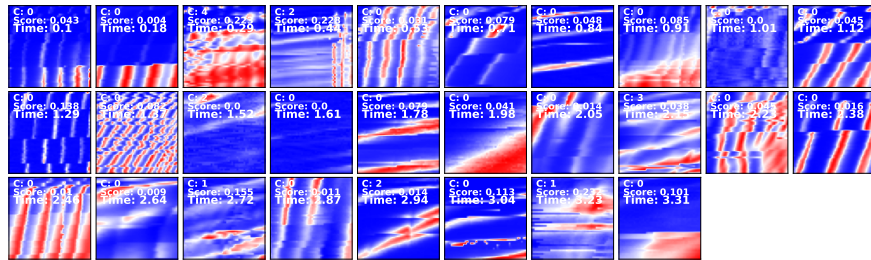


Figure 36: Heterostructure run 14

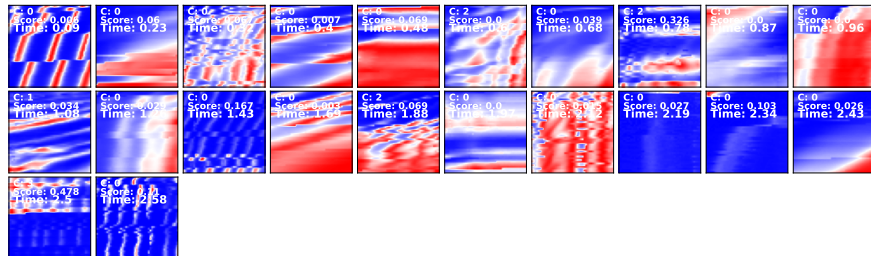


Figure 37: Heterostructure run 15

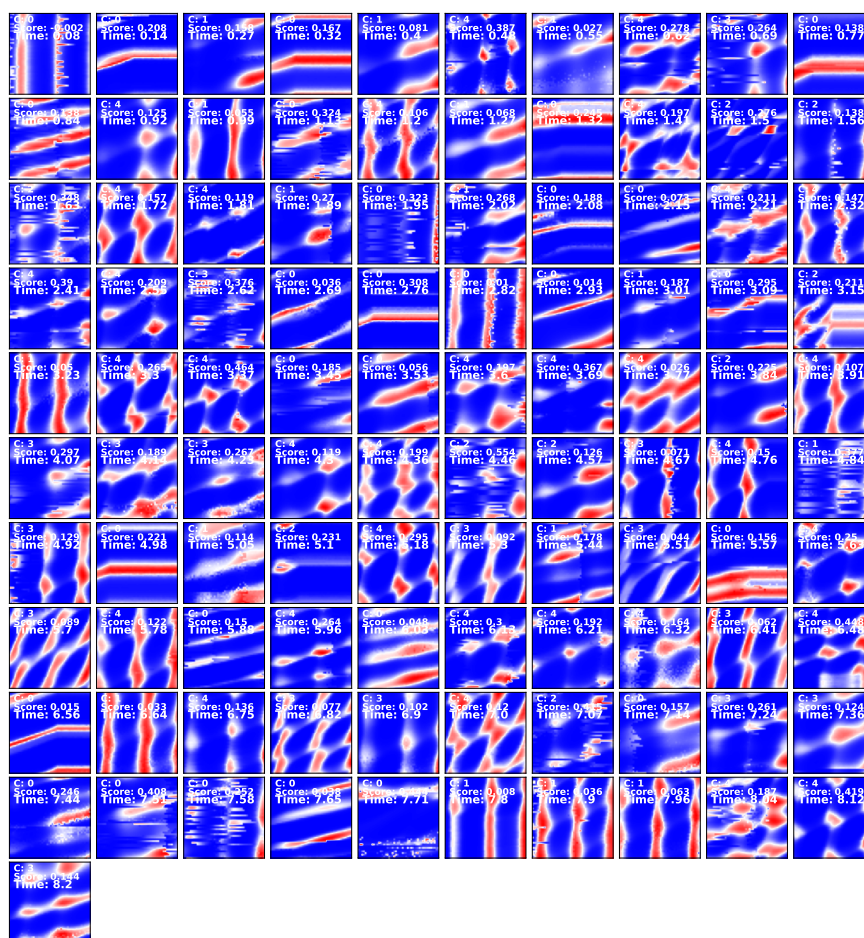


Figure 38: Nanowire run 1

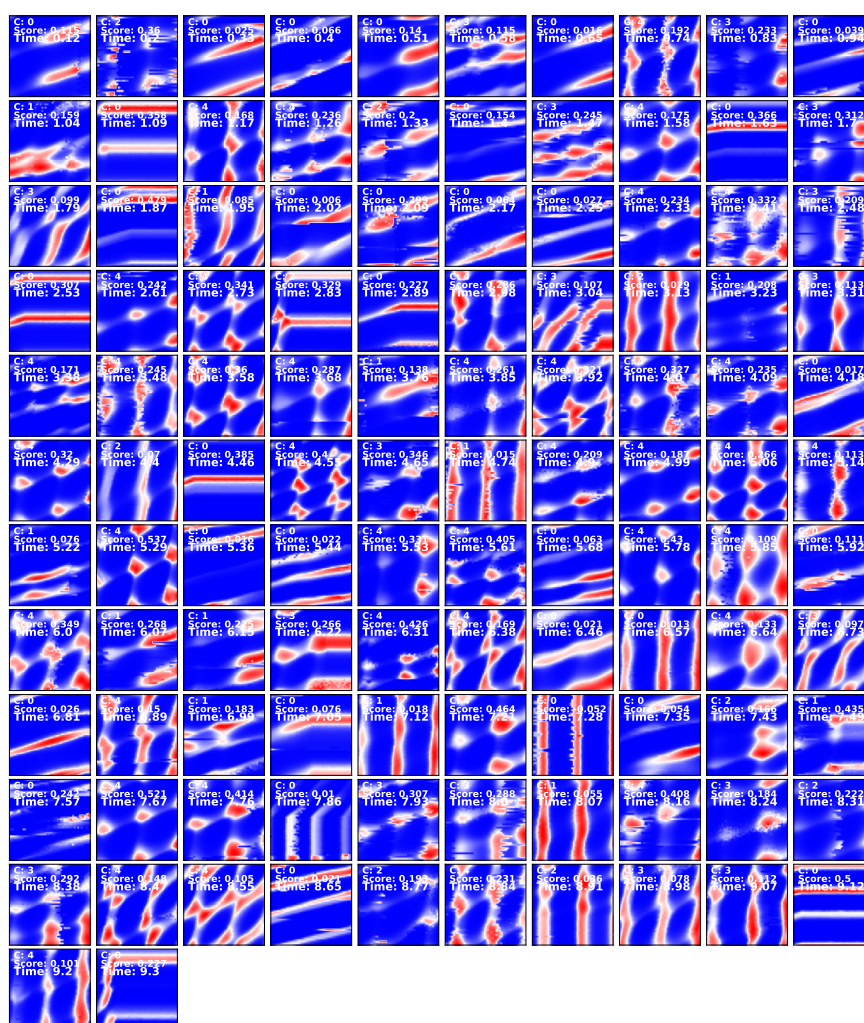


Figure 39: Nanowire run 2

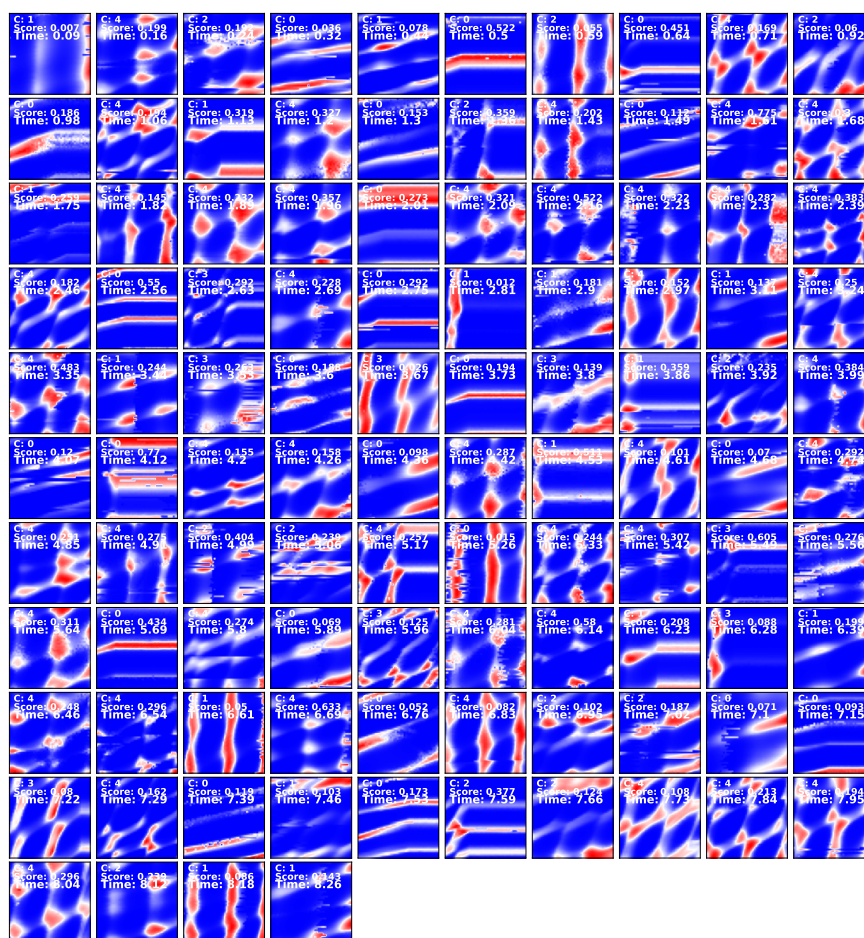


Figure 40: Nanowire run 3

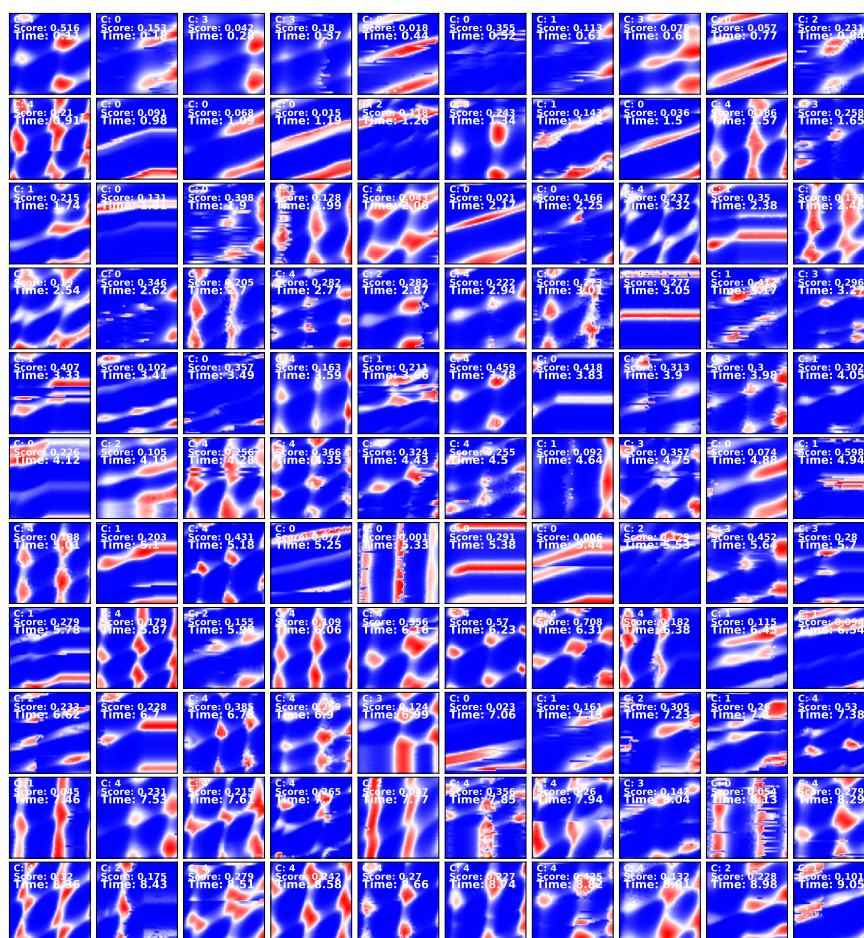


Figure 41: Nanowire run 4

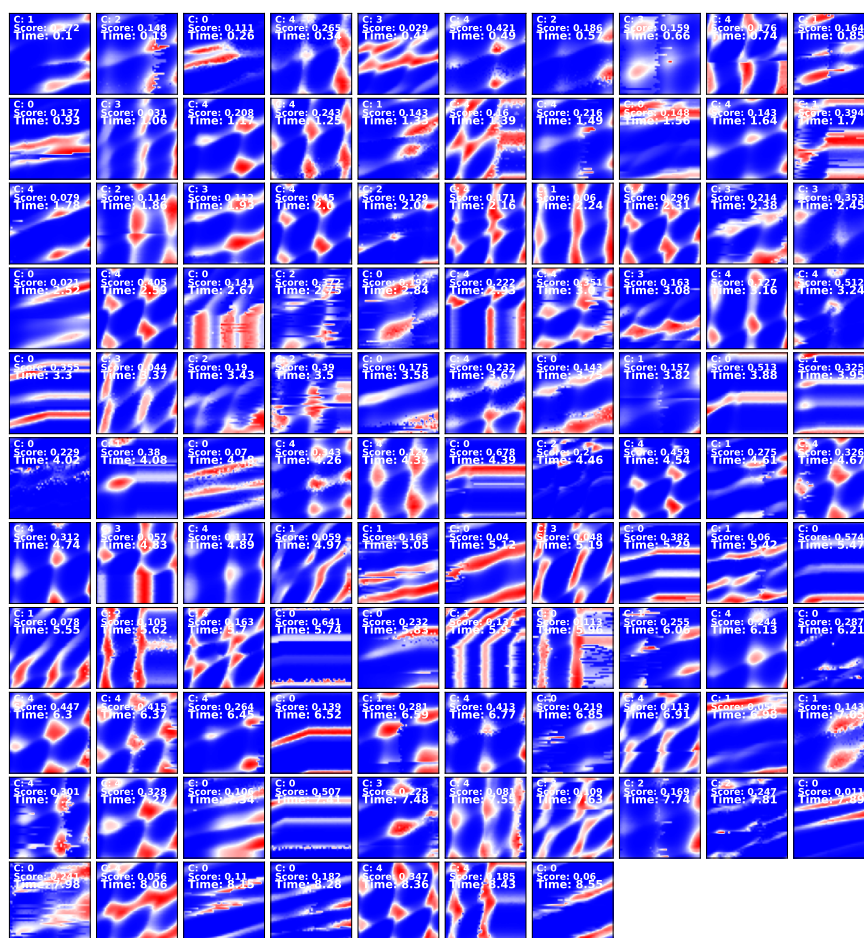


Figure 42: Nanowire run 5

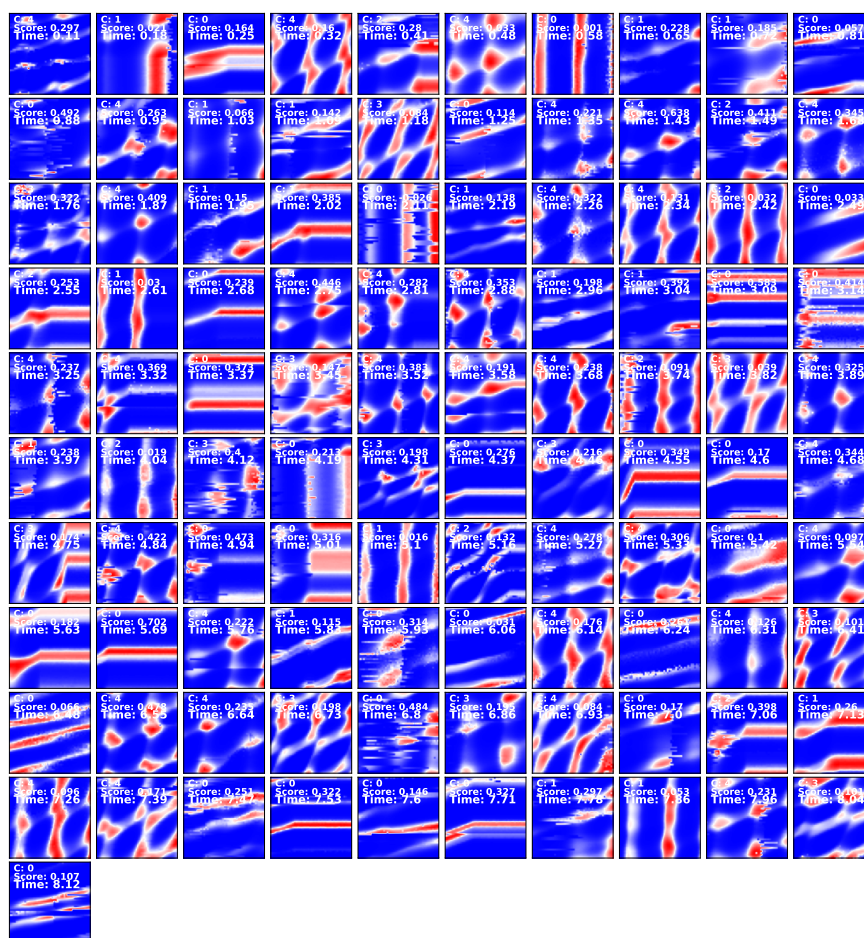


Figure 43: Nanowire run 6

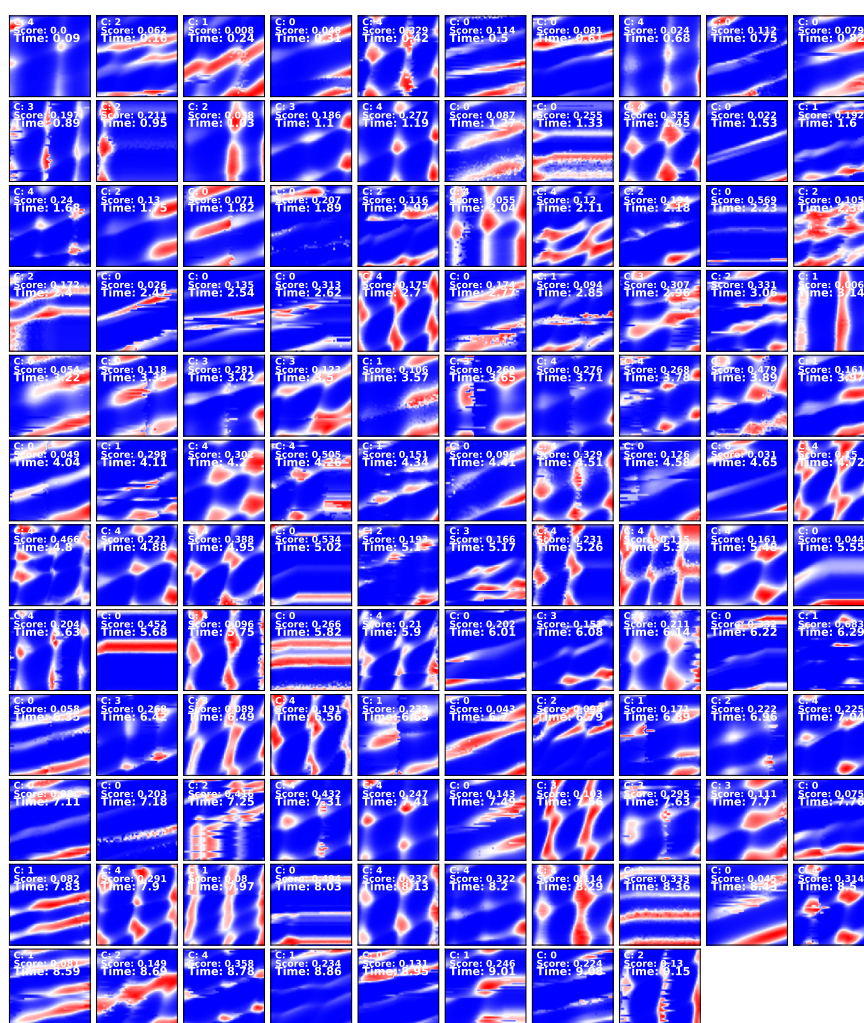


Figure 44: Nanowire run 7

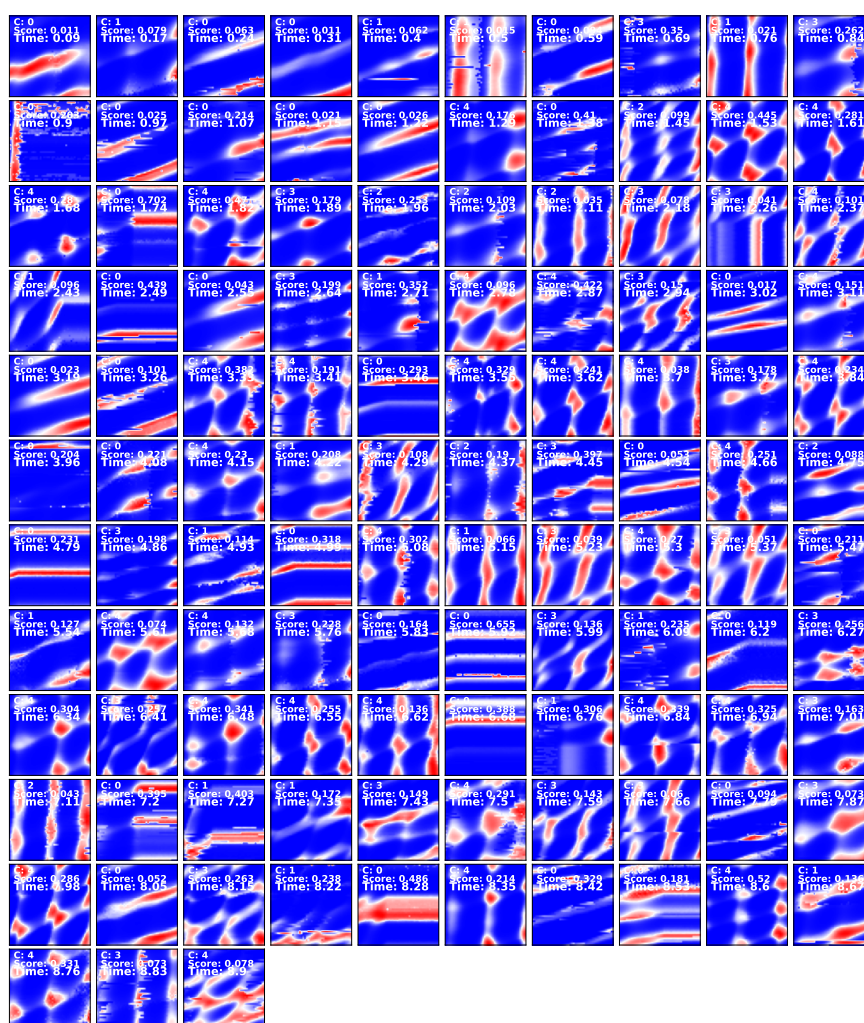


Figure 45: Nanowire run 8

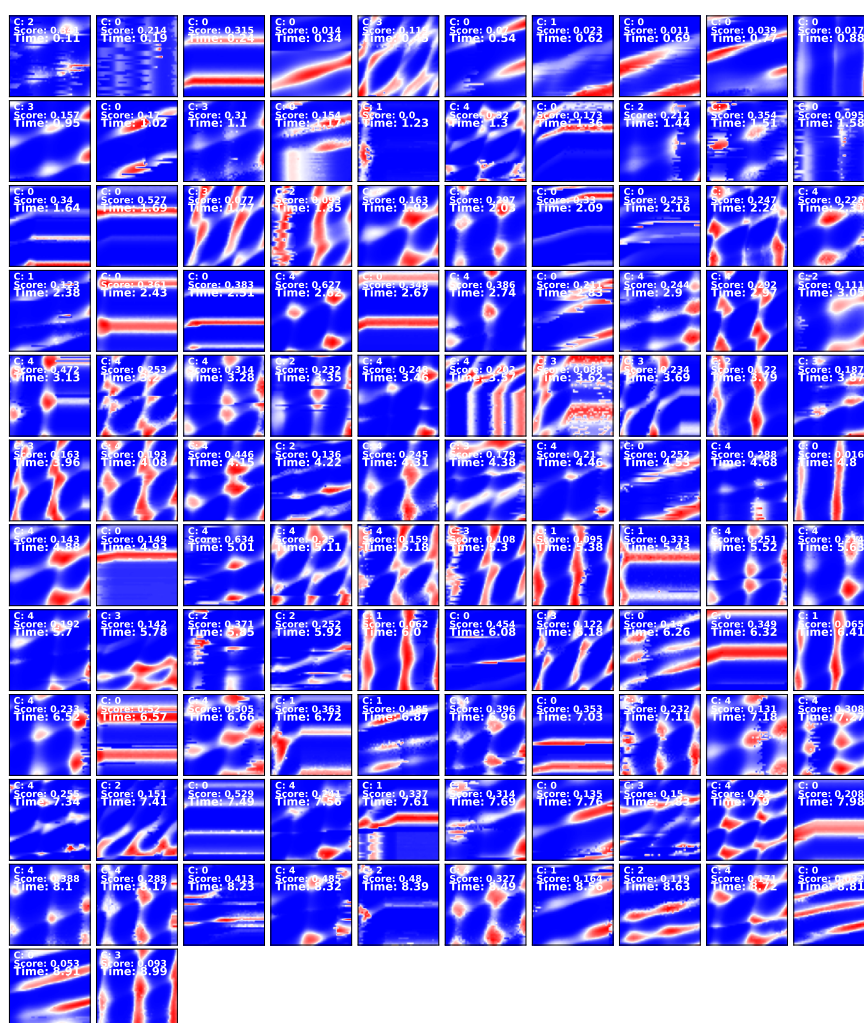


Figure 46: Nanowire run 9

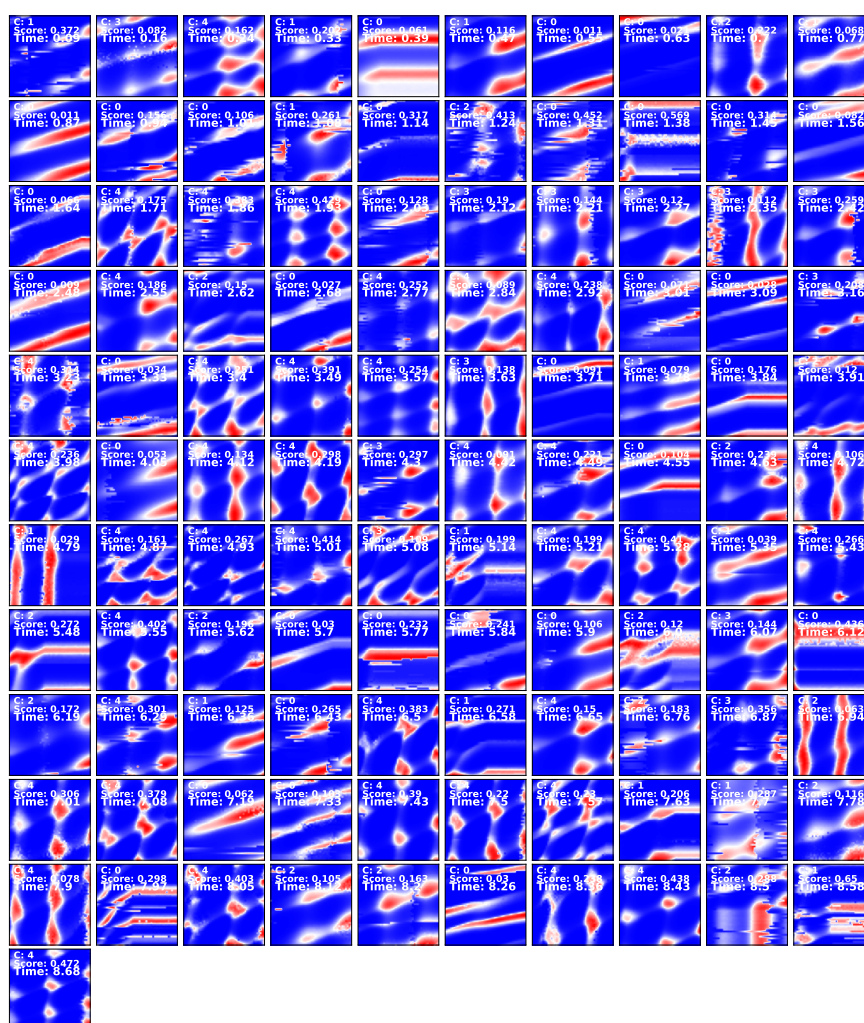


Figure 47: Nanowire run 10

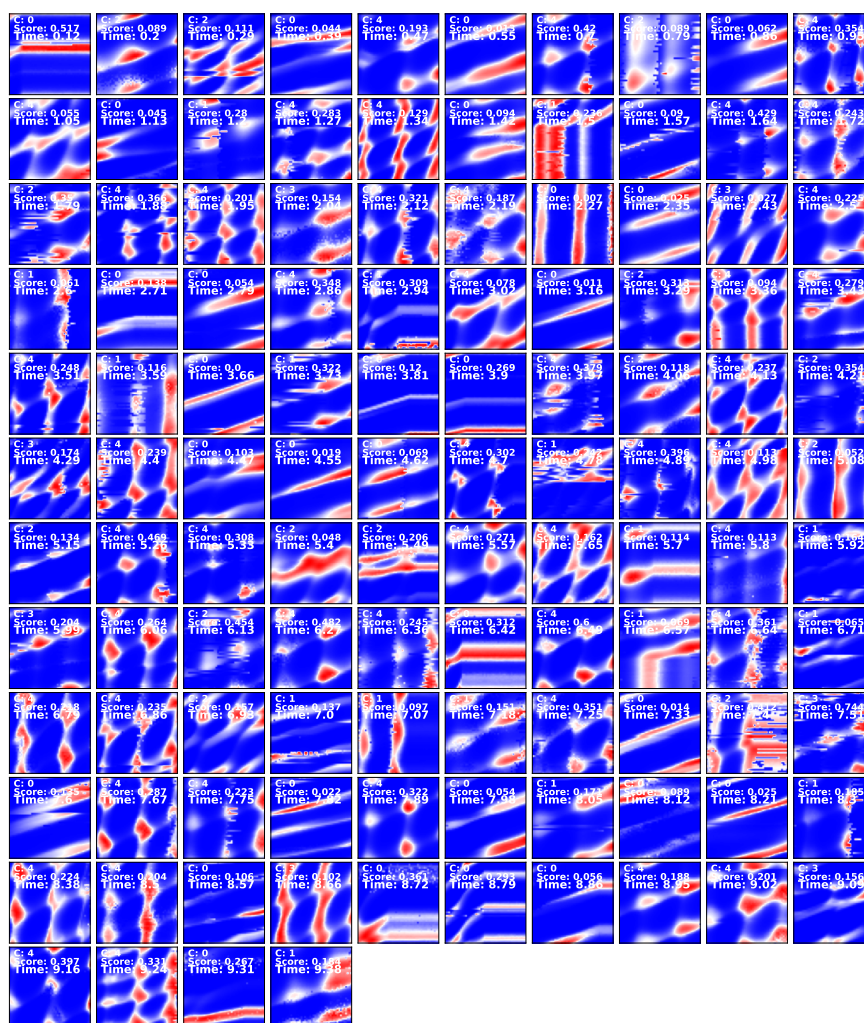


Figure 48: Nanowire run 11

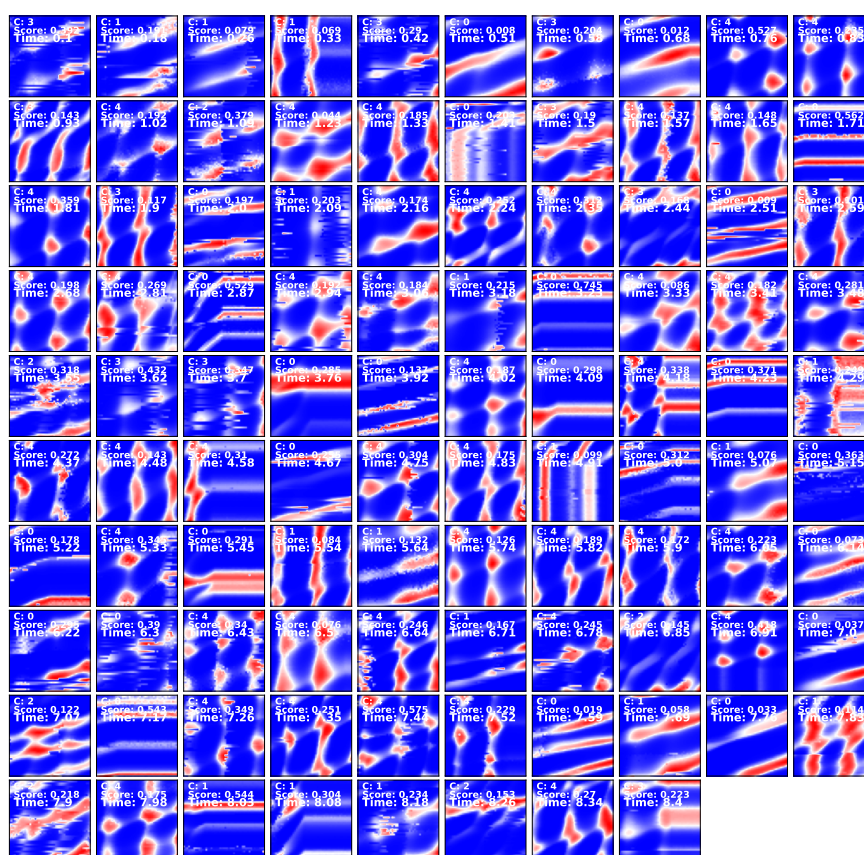


Figure 49: Nanowire run 12

.12 Bounding length scales

Length scales in a Gaussian process govern the degree of correlation between locations in the input space. Large length scales lead to high correlation over large distances making the form of the prediction smooth. Conversely small length scales lead to low correlation over large distances predictions sharper.

Over the gate voltage space a human operator has an informative prior over the length scales of the operability volume. It should be expected to encompass multiple charge transitions. Given an estimate for the lever arm and charging energy a lower bound can be constructed of some $n_{\min}$ times coulomb peak spacing; I assume a spacing of 10 mV in this demonstration. This gives using a value of 2 supplies a lower bound of $V_{lmin} \approx 20 \text{ mV}$. Again, it can be expected that the operable volume will encompass $n_{\max}$ coulomb peaks. Using a value of 30 supplies a upper bound of $V_{lmin} \approx 300 \text{ mV}$. The Gaussian processes take a unit vector as input so their length scales apply to the space of unit vectors. If we consider a Gaussian process with an RBF kernel and aim for a correlation of order 0.01 over stated distances then the following formula can be used to convert back and forth

$$L \approx 3\sqrt{2 - 2\cos \frac{V}{R_{\mu}}}. \quad (8)$$

R_{μ} is an estimate of the average typical distance to the hypersurface, V is an estimated length scale in voltage space and L is the length scale in the unit vector space. This approximation is only valid if r does not change greatly with $\mathbf{u}$.

.13 Coulomb peak classifier

Due to the different types of current noise observed for each of the devices considered, a robust Coulomb peak detector was required. We thus developed a random forest based Coulomb peak classifier.

A set of 128-pixel current traces was obtained on different devices to those for which the algorithm was tested. 6722 current traces were labelled by a single labeller, from which 553 were labelled as positive (current traces containing Coulomb

peaks) and the remainder (6169 current traces) were labelled as negative. 553 negative examples were randomly selected from the shuffled 6169 negative examples, forming a balanced dataset of 1106 current traces. The data was split into a 70/30% train test split. The random forest classifier was trained on were the normalised current values of each trace at each pixel point, thus each sample had 128 characteristic features. The classifier relies on the Scikit-learn's ensemble RandomForestClassifier package [108]. An accuracy of 84 was achieved. The random forest classifier was then retested on 1562 current traces from a 5-gate GeSi heterostructure device and an accuracy of 92 achieved. This high accuracy contrasts the Coulomb peak detector used in Ref Chapter 4, which achieved an accuracy of 20% classifying the same traces.

.14 Expectation maximisation function

The function to be optimised is often referred to in literature as a Q function not to be confused with Q in Chapter 8. I will refer to the function as H to avoid confusion. If the membership of observations was known then training could be achieved by considering the likelihood for a mixture model:

$$L(\theta|\bar{q}_{1:N}, \bar{z}_{1:N}) = \prod_{n=1}^N \prod_{j=1}^3 [f_j(\bar{q}_n|\theta_j)w_j]^{\mathbb{I}(\bar{z}_n=j)} \quad (9)$$

and it's log likelihood:

$$LL(\theta|\bar{q}_{1:N}, \bar{z}_{1:N}) = \sum_{n=1}^N \sum_{j=1}^3 \mathbb{I}(\bar{z}_n = j) [\log(w_j) + \log(f_j(\bar{q}_n|\theta_j))] \quad (10)$$

When the latent membership of each observation $\bar{z}_n$ is known θ can simply be found by optimising equation 10. To circumvent this expectation maximisation introduces a function

$$H(\theta|\theta^{(t)}) = \mathbb{E}_{\bar{Z}|\bar{Q},\theta^{(t)}} [LL(\theta|\bar{q}, \bar{Z})] \quad (11)$$

The expectation introduces an $N \times 3$ matrix $T_{n,j}^{(t)}$

$$T_{n,j}^{(t)} = \frac{w_j f_j(\bar{q}_n|\theta_j)}{\sum_{j=1}^3 w_j f_j(\bar{q}_n|\theta_j)} \quad (12)$$

Giving a final form for H

$$H(\theta|\theta^{(t)}) = \sum_{n=1}^N \sum_{j=1}^3 T_{n,j}^{(t)} [\log(w_j) + \log(f_j(\bar{q}_n|\theta_j))] \quad (13)$$

In the E step the elements of $T_{n,j}^{(t)}$ are populated for the current estimate of $\theta^{(t)}$. In the M step H is maximised. Because w_j appears in its own separate linear term it can be maximised independently.

$$w_j^{(t+1)} = \frac{1}{N} \sum_{n=1}^N T_{n,j}^{(t)} \quad (14)$$

When f_j is d dimensional multivariate Gaussian in form

$$\log(f_j(\bar{q}|\theta_j)) = -\frac{1}{2} \log(|\Sigma_j|) - \frac{1}{2} (\bar{q} - \mu_j)^T \Sigma_j^{-1} (\bar{q} - \mu_j) - \frac{d}{2} \log(2\pi) \quad (15)$$

As μ_j and Σ_j appear in separate linear terms and are quadratic in form optimisation can be achieved without applying an optimiser. It can be derived that:

$$\mu_j^{(t+1)} = \frac{\sum_{n=1}^N T_{n,j}^{(t)} \bar{q}_n}{\sum_{n=1}^N T_{n,j}^{(t)}}, \Sigma_j^{(t+1)} = \frac{\sum_{n=1}^N T_{n,j}^{(t)} (\bar{q}_n - \mu_j^{(t+1)}) (\bar{q}_n - \mu_j^{(t+1)})^T}{\sum_{n=1}^N T_{n,j}^{(t)}} \quad (16)$$

The mixture model used in Chapter 8 contains two gaussian components and one Gaussian mixture model containing many components. For speed only the weight of the mixture component is trained in the manner detailed here. Once the parameters for the next time step are learnt for the gaussian components their parameters are used to directly update the mixture component as detailed in section 8.2.2. This is not optimal but is shown to have good performance.

References

- [1] Richard P Feynman. “Simulating physics with computers”. In: *Feynman and computation*. CRC Press, 2018, pp. 133–153.
- [2] Daniel Loss and David P DiVincenzo. “Quantum computation with quantum dots”. In: *Physical Review A* 57.1 (1998), p. 120.
- [3] David P DiVincenzo. “The physical implementation of quantum computation”. In: *Fortschritte der Physik: Progress of Physics* 48.9-11 (2000), pp. 771–783.
- [4] National Academies of Sciences Engineering, Medicine, et al. *Quantum computing: progress and prospects*. National Academies Press, 2019.
- [5] Rodney Van Meter and Clare Horsman. “A blueprint for building a quantum computer”. In: *Communications of the ACM* 56.10 (2013), pp. 84–93.
- [6] Dave Wecker et al. “Gate-count estimates for performing quantum chemistry on small quantum computers”. In: *Physical Review A* 90.2 (2014), p. 022305.
- [7] Yang Wu et al. “A programmable two-qubit solid-state quantum processor under ambient conditions”. In: *npj Quantum Information* 5.1 (2019), pp. 1–5.
- [8] Jason R Petta et al. “Coherent manipulation of coupled electron spins in semiconductor quantum dots”. In: *Science* 309.5744 (2005), pp. 2180–2184.
- [9] Erika Kawakami et al. “Electrical control of a long-lived spin qubit in a Si/SiGe quantum dot”. In: *Nature nanotechnology* 9.9 (2014), pp. 666–670.
- [10] Florian NM Froning et al. “Ultrafast hole spin qubit with gate-tunable spin–orbit switch functionality”. In: *Nature Nanotechnology* 16.3 (2021), pp. 308–312.
- [11] JWG Van den Berg et al. “Fast spin-orbit qubit in an indium antimonide nanowire”. In: *Physical review letters* 110.6 (2013), p. 066806.
- [12] Edward A Laird, Fei Pei, and Leo P Kouwenhoven. “A valley–spin qubit in a carbon nanotube”. In: *Nature nanotechnology* 8.8 (2013), pp. 565–568.
- [13] ZV Penfold-Fitch, F Sfigakis, and MR Buitelaar. “Microwave spectroscopy of a carbon nanotube charge qubit”. In: *Physical Review Applied* 7.5 (2017), p. 054017.
- [14] R Maurand et al. “A CMOS silicon spin qubit”. In: *Nature communications* 7.1 (2016), pp. 1–6.
- [15] Leon C Camenzind et al. “A spin qubit in a fin field-effect transistor”. In: *arXiv preprint arXiv:2103.07369* (2021).
- [16] Jarryd J Pla et al. “A single-atom electron spin qubit in silicon”. In: *Nature* 489.7417 (2012), pp. 541–545.
- [17] Jarryd J Pla et al. “High-fidelity readout and control of a nuclear spin qubit in silicon”. In: *Nature* 496.7445 (2013), pp. 334–338.

- [18] Yu He et al. “A two-qubit gate between phosphorus donor electrons in silicon”. In: *Nature* 571.7765 (2019), pp. 371–375.
- [19] Frank HL Koppens et al. “Driven coherent oscillations of a single electron spin in a quantum dot”. In: *Nature* 442.7104 (2006), pp. 766–771.
- [20] Katja C Nowack et al. “Coherent control of a single electron spin with electric fields”. In: *Science* 318.5855 (2007), pp. 1430–1433.
- [21] EA Laird et al. “Hyperfine-mediated gate-driven electron spin resonance”. In: *Physical review letters* 99.24 (2007), p. 246601.
- [22] Kenta Takeda et al. “A fault-tolerant addressable spin qubit in a natural silicon quantum dot”. In: *Science advances* 2.8 (2016), e1600694.
- [23] JM Elzerman et al. “Single-shot read-out of an individual electron spin in a quantum dot”. In: *nature* 430.6998 (2004), pp. 431–435.
- [24] Jun Yoneda et al. “A quantum-dot spin qubit with coherence limited by charge noise and fidelity higher than 99.9%”. In: *Nature nanotechnology* 13.2 (2018), pp. 102–106.
- [25] TF Watson et al. “A programmable two-qubit quantum processor in silicon”. In: *nature* 555.7698 (2018), pp. 633–637.
- [26] Jeremy Levy. “Universal quantum computation with spin-1/2 pairs and Heisenberg exchange”. In: *Physical Review Letters* 89.14 (2002), p. 147902.
- [27] Xian Wu et al. “Two-axis control of a singlet–triplet qubit with an integrated micromagnet”. In: *Proceedings of the National Academy of Sciences* 111.33 (2014), pp. 11938–11942.
- [28] Sandra Foletti et al. “Universal quantum control of two-electron spin quantum bits using dynamic nuclear polarization”. In: *Nature Physics* 5.12 (2009), pp. 903–908.
- [29] Jonas Schuff et al. “Identifying Pauli spin blockade using deep learning”. In: *arXiv preprint arXiv:2202.00574* (2022).
- [30] Takumi Ito et al. “Detection and control of charge states in a quintuple quantum dot”. In: *Scientific reports* 6.1 (2016), pp. 1–7.
- [31] Richard O Duda and Peter E Hart. “Use of the Hough transformation to detect lines and curves in pictures”. In: *Communications of the ACM* 15.1 (1972), pp. 11–15.
- [32] Maxime Lapointe-Major et al. “Algorithm for automated tuning of a quantum dot into the single-electron regime”. In: *Physical Review B* 102.8 (2020), p. 085301.
- [33] Giovanni A Oakes et al. “Automatic virtual voltage extraction of a 2xN array of quantum dots with machine learning”. In: *arXiv preprint arXiv:2012.03685* (2020).
- [34] Itzhak Fogel and Dov Sagi. “Gabor filters as texture discriminator”. In: *Biological cybernetics* 61.2 (1989), pp. 103–113.
- [35] Timothy A Baart et al. “Computer-automated tuning of semiconductor double quantum dots into the single-electron regime”. In: *Applied Physics Letters* 108.21 (2016), p. 213104.

- [36] Sandesh S Kalantre et al. “Machine learning techniques for state recognition and auto-tuning in quantum dots”. In: *npj Quantum Information* 5.1 (2019), pp. 1–10.
- [37] Justyna P Zwolak et al. “Autotuning of double-dot devices in situ with machine learning”. In: *Physical review applied* 13.3 (2020), p. 034075.
- [38] Renato Durrer et al. “Automated tuning of double quantum dots into specific charge states using neural networks”. In: *Physical Review Applied* 13.5 (2020), p. 054019.
- [39] Joshua Ziegler et al. “Toward Robust Autotuning of Noisy Quantum Dot Devices”. In: *arXiv preprint arXiv:2108.00043* (2021).
- [40] Jana Darulová et al. “Autonomous tuning and charge-state detection of gate-defined quantum dots”. In: *Physical Review Applied* 13.5 (2020), p. 054005.
- [41] CJ Van Diepen et al. “Automated tuning of inter-dot tunnel coupling in double quantum dots”. In: *Applied Physics Letters* 113.3 (2018), p. 033101.
- [42] AR Mills et al. “Computer-automated tuning procedures for semiconductor quantum dot arrays”. In: *Applied Physics Letters* 115.11 (2019), p. 113501.
- [43] Christian Volk et al. “Loading a quantum-dot based Qubyte register”. In: *npj Quantum Information* 5.1 (2019), pp. 1–8.
- [44] Toivo Hensgens et al. “Quantum simulation of a Fermi–Hubbard model using a semiconductor quantum dot array”. In: *Nature* 548.7665 (2017), pp. 70–73.
- [45] T-K Hsiao et al. “Efficient orthogonal control of tunnel couplings in a quantum dot array”. In: *Physical Review Applied* 13.5 (2020), p. 054018.
- [46] Greg Welch, Gary Bishop, et al. “An introduction to the Kalman filter”. In: (1995).
- [47] Julian D Teske et al. “A machine learning approach for automated fine-tuning of semiconductor spin qubits”. In: *Applied physics letters* 114.13 (2019), p. 133102.
- [48] Warren S McCulloch and Walter Pitts. “A logical calculus of the ideas immanent in nervous activity”. In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133.
- [49] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105.
- [50] Dan Ciresan et al. “Deep neural networks segment neuronal membranes in electron microscopy images”. In: *Advances in neural information processing systems* 25 (2012), pp. 2843–2851.
- [51] Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. “Multi-column deep neural networks for image classification”. In: *2012 IEEE conference on computer vision and pattern recognition*. IEEE. 2012, pp. 3642–3649.
- [52] Dan C Ciregan et al. “Mitosis detection in breast cancer histology images with deep neural networks”. In: *International conference on medical image computing and computer-assisted intervention*. Springer. 2013, pp. 411–418.
- [53] Kuniyiko Fukushima and Sei Miyake. “Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition”. In: *Competition and cooperation in neural nets*. Springer, 1982, pp. 267–285.

- [54] William R Thompson. “On the likelihood that one unknown probability exceeds another in view of the evidence of two samples”. In: *Biometrika* 25.3/4 (1933), pp. 285–294.
- [55] Aleksandrs Slivkins. “Introduction to multi-armed bandits”. In: *arXiv preprint arXiv:1904.07272* (2019).
- [56] Peter Auer and Ronald Ortner. “UCB revisited: Improved regret bounds for the stochastic multi-armed bandit problem”. In: *Periodica Mathematica Hungarica* 61.1-2 (2010), pp. 55–65.
- [57] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. “Finite-time analysis of the multiarmed bandit problem”. In: *Machine learning* 47.2 (2002), pp. 235–256.
- [58] Djallel Bouneffouf et al. “Optimal exploitation of clustering and history information in multi-armed bandit”. In: *arXiv preprint arXiv:1906.03979* (2019).
- [59] Shipra Agrawal and Navin Goyal. “Analysis of thompson sampling for the multi-armed bandit problem”. In: *Conference on learning theory. JMLR Workshop and Conference Proceedings*. 2012, pp. 39–1.
- [60] Olivier Chapelle and Lihong Li. “An empirical evaluation of thompson sampling”. In: *Advances in neural information processing systems* 24 (2011), pp. 2249–2257.
- [61] Siddhartha Chib and Edward Greenberg. “Understanding the metropolis-hastings algorithm”. In: *The american statistician* 49.4 (1995), pp. 327–335.
- [62] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. “Variational inference: A review for statisticians”. In: *Journal of the American statistical Association* 112.518 (2017), pp. 859–877.
- [63] Carl Edward Rasmussen. “Gaussian processes in machine learning”. In: *Summer school on machine learning*. Springer. 2003, pp. 63–71.
- [64] R Radebaugh and JD Siegwarth. “Dilution refrigerator technology”. In: *Cryogenics* 11.5 (1971), pp. 368–384.
- [65] Aaron van den Oord et al. “Wavenet: A generative model for raw audio”. In: *arXiv preprint arXiv:1609.03499* (2016).
- [66] Alireza Makhzani et al. “Adversarial autoencoders”. In: *arXiv preprint arXiv:1511.05644* (2015).
- [67] Huaibo Huang et al. “Introvae: Introspective variational autoencoders for photographic image synthesis”. In: *arXiv preprint arXiv:1807.06358* (2018).
- [68] Jun-Yan Zhu et al. “Unpaired image-to-image translation using cycle-consistent adversarial networks”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2223–2232.
- [69] Yaniv Taigman, Adam Polyak, and Lior Wolf. “Unsupervised cross-domain image generation”. In: *arXiv preprint arXiv:1611.02200* (2016).
- [70] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. “Globally and locally consistent image completion”. In: *ACM Transactions on Graphics (ToG)* 36.4 (2017), pp. 1–14.
- [71] Rafael Gómez-Bombarelli et al. “Automatic chemical design using a data-driven continuous representation of molecules”. In: *ACS central science* 4.2 (2018), pp. 268–276.

- [72] Benjamin Sanchez-Lengeling and Alán Aspuru-Guzik. “Inverse molecular design using machine learning: Generative models for matter engineering”. In: *Science* 361.6400 (2018), pp. 360–365.
- [73] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. “Learning structured output representation using deep conditional generative models”. In: *Advances in neural information processing systems* 28 (2015), pp. 3483–3491.
- [74] Ian Goodfellow et al. “Generative adversarial nets”. In: *Advances in neural information processing systems* 27 (2014).
- [75] Diederik P Kingma and Max Welling. “Auto-encoding variational bayes”. In: *arXiv preprint arXiv:1312.6114* (2013).
- [76] Mikael Sunnåker et al. “Approximate bayesian computation”. In: *PLoS computational biology* 9.1 (2013), e1002803.
- [77] Neil Houlsby et al. “Bayesian active learning for classification and preference learning”. In: *arXiv preprint arXiv:1112.5745* (2011).
- [78] Andreas Kirsch, Joost Van Amersfoort, and Yarin Gal. “Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning”. In: *Advances in neural information processing systems* 32 (2019), pp. 7026–7037.
- [79] Martin Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015.
- [80] Gerrit van den Burg and Chris Williams. “On memorization in probabilistic deep generative models”. In: *Advances in Neural Information Processing Systems* 34 (2021).
- [81] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-net: Convolutional networks for biomedical image segmentation”. In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.
- [82] Nick Kanopoulos, Nagesh Vasanthavada, and Robert L Baker. “Design of an image edge detection filter using the Sobel operator”. In: *IEEE Journal of solid-state circuits* 23.2 (1988), pp. 358–367.
- [83] F. Kuemmeth. personal communication. Dec. 12, 2019.
- [84] William E Lorensen and Harvey E Cline. “Marching cubes: A high resolution 3D surface construction algorithm”. In: *ACM siggraph computer graphics* 21.4 (1987), pp. 163–169.
- [85] DL Craig et al. “Bridging the reality gap in quantum devices with physics-aware machine learning”. In: *arXiv preprint arXiv:2111.11285* (2021).
- [86] Pauli Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272.
- [87] Hariharan Narayanan and Partha Niyogi. “Sampling hypersurfaces through diffusion”. In: *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*. Springer, 2008, pp. 535–548.
- [88] Andreas V Kuhlmann et al. “Ambipolar quantum dots in undoped silicon fin field-effect transistors”. In: *Applied Physics Letters* 113.12 (2018), p. 122107.

- [89] Simon Geyer et al. “Self-aligned gates for scalable silicon quantum computing”. In: *Applied Physics Letters* 118.10 (2021), p. 104004.
- [90] FNM Froning et al. “Strong spin-orbit interaction and g-factor renormalization of hole spins in Ge/Si nanowire quantum dots”. In: *Physical Review Research* 3.1 (2021), p. 013081.
- [91] Daniel Jirovec et al. “A singlet-triplet hole spin qubit in planar Ge”. In: *Nature Materials* (2021), pp. 1–7.
- [92] Andriy Myronenko and Xubo Song. “Point set registration: Coherent point drift”. In: *IEEE transactions on pattern analysis and machine intelligence* 32.12 (2010), pp. 2262–2275.
- [93] Nard Dumoulin Stuyck et al. “An Integrated Silicon MOS Single-Electron Transistor Charge Sensor for Spin-Based Quantum Information Processing”. In: *IEEE Electron Device Letters* 41.8 (2020), pp. 1253–1256.
- [94] Ulrik Hanke, Yu M Galperin, and KA Chao. “Charge sensitivity of a single electron transistor”. In: *Applied physics letters* 65.14 (1994), pp. 1847–1849.
- [95] Herbert Edelsbrunner, David Kirkpatrick, and Raimund Seidel. “On the shape of a set of points in the plane”. In: *IEEE Transactions on information theory* 29.4 (1983), pp. 551–559.
- [96] Hyunjik Kim and Andriy Mnih. “Disentangling by factorising”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 2649–2658.
- [97] Laurens van der Maaten and Geoffrey Hinton. “Visualizing Data using t-SNE”. In: *Journal of Machine Learning Research* 9 (2008), pp. 2579–2605.
- [98] Wilfred G Van der Wiel et al. “Electron transport through double quantum dots”. In: *Reviews of Modern Physics* 75.1 (2002), p. 1.
- [99] Federico Fedele et al. “Simultaneous Operations in a Two-Dimensional Array of Singlet-Triplet Qubits”. In: *PRX Quantum* 2 (4 Oct. 2021), p. 040306.
- [100] Christian Barthel et al. “Fast sensing of double-dot charge arrangement and spin state with a radio-frequency sensor quantum dot”. In: *Physical Review B* 81.16 (2010), p. 161308.
- [101] Christian Barthel et al. “Rapid single-shot measurement of a singlet-triplet qubit”. In: *Physical Review Letters* 103.16 (2009), p. 160503.
- [102] Todd K Moon. “The expectation-maximization algorithm”. In: *IEEE Signal processing magazine* 13.6 (1996), pp. 47–60.
- [103] Gideon Schwarz. “Estimating the dimension of a model”. In: *The annals of statistics* (1978), pp. 461–464.
- [104] Patrick Harvey-Collard et al. “High-fidelity single-shot readout for a spin qubit via an enhanced latching mechanism”. In: *Physical Review X* 8.2 (2018), p. 021046.
- [105] Thu Pham-Gia, Noyan Turkkan, and Andriëtte Bekker. “Bounds for the Bayes error in classification: A Bayesian approach using discriminant analysis”. In: *Statistical Methods and Applications* 16.1 (2007), pp. 7–26.
- [106] SA Studenikin et al. “Enhanced charge detection of spin qubit readout via an intermediate state”. In: *Applied Physics Letters* 101.23 (2012), p. 233101.

- [107] Leon C Camenzind et al. “Hyperfine-phonon spin relaxation in a single-electron GaAs quantum dot”. In: *Nature communications* 9.1 (2018), pp. 1–6.
- [108] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.