Investigating Graph Neural Networks and Classical Feature-Extraction Techniques in Activity-Cliff and Molecular Property Prediction

Markus Ferdinand Dablander

Mansfield College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

October 2023

Acknowledgements

First and foremost, I would like to say thank you to my scientific supervisors Prof. Garrett M. Morris, Prof. Renaud Lambiotte, and Dr Thierry Hanser, who always took the time to have deep discussions, who supported me in my years of doctoral research to explore novel ideas, and who encouraged me in the right moments to push forward.

I would like to thank Garrett for sharing with me his vast knowledge of the cheminformatics literature, teaching me important chemistry facts, providing exceptionally detailed and dedicated feedback on my work, helping me to navigate all the formal milestones of my doctoral journey, and always making scientific meetings with me and his other doctoral students a priority. Throughout the years, I have immensely benefitted from Garrett's knowledge and have more than once felt the genuine care he actively shows for the well-being of his students.

I would like to thank Renaud for offering his profound and rigorous mathematical expertise, continually encouraging me throughout the ups and downs of my research, reliably finding kind and motivational words for me during the most intensive parts of my doctoral studies, and sharing his insights with me about how to develop an effective and creative mindset for advanced research. One of the most impactful pieces of advice I received from Renaud during my early days as a doctoral student was that the best ideas often come when working on something concrete.

I would like to thank Thierry for consistently giving precise advice on the deepest technical aspects of my work, supporting me to find the right research direction whenever I was facing a difficult crossroad, cordially welcoming me into his industrial research group when I visited him at my partner company Lhasa, and always being interested in talking about new and fascinating ideas from science and beyond. I will warmly remember the long in-person conversations I had with Thierry that did not only

revolve around my scientific work, but also around philosophy, cosmology, consciousness, history, music, technological progress and the future of artificial intelligence.

In addition to thanking my scientific supervisors, I would like to express further gratitude to: The InFoMM CDT directors, Chris Breward and Colin Please, for granting me the privilege to do research at the Mathematical Institute and for guiding me and many other students through our formal doctoral path. The EPSRC (EP/L015803/1) as well as my industrial partner Lhasa for their financial support. Mansfield College for enriching my life at Oxford with a great community, engaging events and a beautiful place to retreat. Stéphane Werner and Jean-Francois Marchaland from Lhasa for always being approachable and staying up to date with my scientific work. My mother and my father for the sacrifices they made without which I would have never been able to become a mathematician. My grandparents and my aunt for their continuous loving support. My brother for always having my back and believing in me. My sister-in-law and my little niece for brightening the mood after a long day. My cousin who is also a mathematician for going on long walks with me on the Austrian countryside to discuss science and life. My old friends from UCL, who have since scattered around the world but have still managed to preserve and foster the seed of our friendship. My new friends from Oxford who have accompanied me along my path. All my other friends from Austria, the UK, and elsewhere who have grown with me throughout the years. And finally, my girlfriend who has deeply shared with me all the highs and lows of my time as a doctoral student, and who has always been there for me with her whole heart.

Abstract

Molecular featurisation refers to the process of transforming molecular data into numerical feature vectors. It is one of the key research areas in molecular machine learning and computational drug discovery. Recently, message-passing graph neural networks (GNNs) have emerged as a novel method to learn differentiable features directly from molecular graphs. While such graph-based techniques hold great theoretical promise, further investigations are needed to clarify if and when they indeed manage to definitively outcompete classical molecular featurisations such as extended-connectivity fingerprints (ECFPs) and physicochemical-descriptor vectors (PDVs).

In this thesis, we systematically explore and further develop classical as well as graph-based molecular featurisation methods for two important tasks: the well-studied problem of molecular property prediction, in particular quantitative structure-activity relationship (QSAR) prediction, and the largely unexplored challenge of activity-cliff (AC) prediction. We first give a mathematical description and critical analysis of PDVs, ECFPs and message-passing GNNs, with a focus on graph isomorphism networks (GINs). We then conduct a rigorous computational study to compare the performance of PDVs, ECFPs and GINs for QSAR and AC-prediction. Following this, we mathematically describe a novel twin neural network model for AC-prediction and experimentally evaluate ECFP-based and GIN-based versions of this dual architecture. In an additional project, we introduce *substructure pooling* as a general mathematical operation for the vectorisation of structural fingerprints that represents a natural counterpart to graph-pooling in GNN-architectures. We propose *Sort & Slice* as a simple substructure-pooling technique for ECFPs that robustly outperforms hashing at molecular property prediction. Finally, we outline two ideas for future research: (i) a graph-based self-supervised learning strategy to make classical molecular featurisations trainable, and (ii) trainable substructure-pooling via differentiable self-attention.

Contents

1 Introduction **1**

1.1 Outline of Thesis and Research Contributions 4

2 Molecular Representations and Featurisations for Machine Learning **8**

2.1 Overview 8

2.2 Molecular Graphs 9

2.3 SMILES Strings 12

2.4 Physicochemical-Descriptor Vectors 14

2.4.1 Mathematical Description 14

2.4.2 Critical View 19

2.5 Extended-Connectivity Fingerprints 21

2.5.1 A Short Background on Structural Fingerprints 21

2.5.2 Mathematical Description 22

2.5.3 Standard and Pharmacophoric Atom Features 26

2.5.4 Critical View 27

2.6 Message-Passing Graph Neural Networks 29

2.6.1 Mathematical Description 29

2.6.2 Graph Convolutional Networks 32

2.6.3 Graph Isomorphism Networks 35

2.6.4 Theoretical Expressivity of Graph Neural Networks 35

2.6.5 Critical View 40

2.7 Molecular Featurisations: Critical Overview 44

3 Exploring Molecular Featurisations for QSAR and Activity-Cliff Prediction: A Computational Study **46**

3.1 Overview 46

3.2 Introduction to Activity Cliffs and Activity-Cliff Prediction 49

3.3 Experimental Methodology 52

3.3.1 Molecular Data Sets 52

3.3.2 Definition of Binary Activity-Cliff Classification-Tasks 56
3.3.3 Developed Pair-Based Data Splitting Technique 58
3.3.4 Prediction Strategies 61
3.3.5 Performance Metrics 63
3.3.6 Model Training and Hyperparameter Optimisation 65
3.4 Results and Discussion 67
3.4.1 QSAR-Prediction Performance 67
3.4.2 AC-Classification Performance 67
3.4.3 PD-Classification Performance 75
3.4.4 Linear Relationship between QSAR-MAE and AC-MCC . . . 75
3.5 Conclusions 76

4 A Twin Neural Network Model for Activity-Cliff Prediction 79
4.1 Overview 79
4.2 Twin Neural Network: Mathematical Description 81
4.2.1 Neural Architecture and Symmetry Properties 81
4.2.2 Loss Function and Model Training 88
4.2.3 Molecular Featurisations: Four Model Versions 91
4.3 Computational Experiments 93
4.3.1 Experimental Methodology 93
4.3.1.1 Molecular Data Set 93
4.3.1.2 Data Splitting Technique and Prediction Tasks . . . 93
4.3.1.3 Prediction Tasks and Prediction Strategies 94
4.3.1.4 Performance Measures 95
4.3.1.5 Evaluated Models 97
4.3.1.6 Model Training and Hyperparameter Settings 97
4.3.2 Results and Discussion 100
4.3.2.1 AC-Classification Performance 100
4.3.2.2 PD-classification Performance 106
4.4 Conclusions 108

5 Beyond Hashing: Substructure-Pooling Techniques to Robustly Improve Extended-Connectivity Fingerprints 111
5.1 Introduction 111
5.2 Methods and Experimental Methodology 115
5.2.1 Substructure Pooling: Mathematical Description 116
5.2.2 Investigated Substructure-Pooling Techniques 120

5.2.2.1 Hashing 122
5.2.2.2 Sort & Slice 123
5.2.2.3 Filtering 126
5.2.2.4 Mutual-Information Maximisation 129
5.2.3 Experimental Setup 130
5.3 Results and Discussion 134
5.4 Conclusions 144

6 Future Directions **148**
6.1 A Graph-Based Self-Supervised Learning Strategy to Make Classical Molecular Featurisations Trainable 148
6.2 Trainable Substructure Pooling via Differentiable Self-Attention . . . 150

7 Conclusions and Further Thoughts **156**

Summary of Research Contributions **160**

Bibliography **163**

List of Figures

2.1 Generation of a SMILES string. 13
2.2 Circular chemical subgraphs of varying radii. 23
2.3 Molecular featurisation via message-passing GNN. 32
2.4 Non-isomorphic graphs that cannot be distinguished by the 1-WL test. 37
3.1 Example of an AC for factor Xa. 49
3.2 Protein structure of dopamine receptor D2. 53
3.3 Protein structure of factor Xa. 54
3.4 Protein structure of SARS-CoV-2 main protease. 55
3.5 Data splitting strategy for AC-prediction study. 60
3.6 Overview of investigated QSAR models for AC-prediction study. . . . 65
3.7 QSAR and AC-prediction results for dopamine receptor D2. 68
3.8 QSAR and AC-prediction results for factor Xa. 69
3.9 QSAR and AC-prediction results for SARS-CoV-2 main protease. . . 70
3.10 QSAR and PD-prediction results for for dopamine receptor D2. . . . 71
3.11 QSAR and PD-prediction results for factor Xa. 72
3.12 QSAR and PD-prediction results for SARS-CoV-2 main protease. . . 73
4.1 Twin neural network architecture for AC and PD-classification. . . . 83
4.2 AC-classification results for twin neural network models. 101
4.3 PD-classification results for twin neural network models. 102
5.1 Overview of investigated substructure-pooling methods. 121
5.2 Substructure-pooling experiments (lipophilicity). 135
5.3 Substructure-pooling experiments (aqueous solubility). 136
5.4 Substructure-pooling experiments (SARS-CoV-2 main protease). . . . 137
5.5 Substructure-pooling experiments (mutagenicity). 138
5.6 Substructure-pooling experiments (estrogen receptor alpha antagonism).139
5.7 Overview of results of all substructure-pooling experiments. 140
6.1 Self-supervised pre-training and fine-tuning strategy for GNNs. 149

List of Tables

2.1 Atom and bond features for molecular graphs. 11
2.2 Physicochemical descriptors for PDVs. 18
2.3 Atom features for ECFPs. 27
2.4 Strengths and weaknesses of PDVs, ECFPs and message-passing GNNs. 44
3.1 Data sets for AC-prediction study. 57
4.1 Hyperparameters of twin neural networks and baseline QSAR models. 98
5.1 Data sets for substructure-pooling experiments. 131
5.2 RF and MLP hyperparameters for substructure-pooling experiments. 132

List of Abbreviations

- AC = Activity Cliff
- AUPRC = Area Under the Precision Recall Curve
- AUROC = Area Under the Receiver Operating Characteristic Curve
- ECFP = Extended-Connectivity Fingerprint
- GCN = Graph Convolutional Network
- GIN = Graph Isomorphism Network
- GNN = Graph Neural Network
- InChI = International Chemical Identifier
- kNN = k-Nearest Neighbour
- MAE = Mean Absolute Error
- MCC = Matthews Correlation Coefficient
- MIM = Mutual-Information Maximisation
- MLP = Multilayer Perceptron
- MMP = Matched Molecular Pair
- NFP = Neural Fingerprint
- PD = Potency Direction
- PDV = Physicochemical-Descriptor Vector
- QSAR = Quantitative Structure-Activity Relationship
- RF = Random Forest
- SAR = Structure-Activity Relationship
- SELFIE = Self-Referencing Embedded String
- SMILES = Simplified Molecular-Input Line-Entry System
- TPE = Tree-structured Parzen Estimator
- WL = Weisfeiler-Lehman

Chapter 1

Introduction

How to best represent a chemical compound in computational form is one of the most fundamental questions in cheminformatics and molecular machine learning. Molecules are complex physical entities whose properties depend on a large number of intertwined factors such as the types of their involved atoms and bonds, their graph-theoretic connectivity structure, their three-dimensional geometry, and the quantum-mechanical behaviour of their associated electrons. Whether a given computational molecular representation is useful intricately depends on the task one wants to solve, on its required data format and on its level of abstraction

In this thesis, we are interested in the representation of chemical compounds as real-valued feature vectors for the purpose of molecular machine learning. We focus on the process of *molecular featurisation* which describes the computational transformation of a non-vectorial molecular representation (often a detailed string of symbols or mathematical graph that fully encodes the chemical composition and structure of an input compound) into an abstract representation as a numerical vector than can readily be processed by a machine-learning system. As we will see, the impact of the chosen molecular featurisation on the predictive performance of a machine-learning system is often profound.

In the field of computer vision, the advent of convolutional neural networks has led to dramatic performance breakthroughs in the last decade [1, 2, 3, 4, 5]. These breakthroughs were caused by the remarkable ability of convolutional architectures to automatically extract relevant high-level features directly from raw visual data. For machine-learning tasks on images, this has since led to an almost universal shift from traditional expert-based *feature engineering* towards differentiable and automatic *feature learning*. This can be seen in contrast to the area of molecular machine learning, where trainable feature learning techniques that operate directly on molecular graphs without intermediate feature-engineering steps have only emerged in the

last few years and have still not managed to conclusively outperform classical non-trainable featurisation methods in a variety of studies [6, 7, 8, 9, 10, 11, 12, 13]. For machine learning on molecular data, breakthroughs comparable to the ones caused by the feature-extraction capabilities of convolutional neural networks in computer vision remain elusive.

For chemical prediction tasks, molecules have classically been represented via physicochemical-descriptor vectors (PDVs) [14] or via structural fingerprints such as extended-connectivity fingerprints (ECFPs) [15, 16]. Both featurisation types rely on non-trainable algorithms to compute specific properties of an input molecule. In the case of physicochemical descriptors, these properties are typically high-level characteristics of an input compound such as the predicted molecular partition coefficient $\log(P)$ that quantifies lipophilicity; in the case of structural fingerprints, these properties are frequently binary features that indicate the presence or absence of certain chemical substructures. The computed properties are collected in numerical feature vectors that can be subsequently used for a downstream machine-learning task. While certain descriptors and fingerprints can lead to considerable predictive performance [6, 7, 8, 10], they also have some serious drawbacks. The most significant of these is that the algorithmic transformation of a detailed molecular representation into a mere vector of selected structural or physicochemical properties can lead to a loss of crucial chemical information. Classical descriptor and fingerprint-based featurisations thus form an information-bottleneck which separates detailed molecular data on one side from its computationally processed form on the other side.

Recently, message-passing graph neural networks (GNNs) [17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28] have entered the field of cheminformatics as a potential solution for this limitation. GNNs are trainable deep-learning architectures that allow the differentiable extraction of continuous features directly from molecular graphs. These graphs can fully specify the connectivity structure and chemical composition of a molecule; even simple stereochemical properties such as tetrahedral R-S chirality and E-Z double-bond geometry can easily be encoded in the atom and bond features of molecular graphs. Such graphs thus form highly explicit and detailed molecular representations. Learning differentiable features directly from molecular graph structures holds the promise of overcoming potential information-bottlenecks during molecular featurisation. Unfortunately, GNNs come with their own set of technical challenges: Some popular GNN-models lack theoretical expressivity and thus cannot learn to distinguish certain simple graph structures [29]; many GNN-architectures cannot be made deep due to a tendency of successively convolved node

features to become indistinguishable [30]; GNNs have to learn a reasonable embedding of chemical space from scratch every time they are trained on a novel task; almost all current GNN-models are based on a local neighbourhood-aggregation scheme [17] which hinders information flow between distant graph nodes; and GNNs require a global pooling step to eventually reduce the graph to a vector that can potentially form a dangerous information-bottleneck in and of itself [31]. A variety of studies have found GNNs to exhibit superior predictive performance compared to the simpler and more computationally efficient fingerprint and descriptor-based featurisations [17, 20, 22, 32, 33, 34, 35, 36]; however, other experiments have pointed towards the exact opposite [6, 7, 8, 9, 10, 11, 12, 13, 37]. The superiority of GNNs over classical techniques thus remains questionable and requires more investigation.

In this work, we aim to explore and further develop classical as well as graph-based molecular featurisation methods with a focus on two important challenges in computational drug discovery: the canonical problem of molecular property prediction, in particular quantitative structure-activity relationship (QSAR) prediction, and the largely unexplored task of activity-cliff (AC) prediction. Molecular property prediction encompasses a diverse number of tasks such as the prediction of lipophilicity, aqueous solubility, toxicity, or mutagenicity of a chemical compound. QSAR-prediction represents a special case of molecular property prediction and refers to the problem of predicting the biological activity of a compound with respect to a given pharmacological target from its chemical structure. QSAR modelling often takes the form of predicting the binding affinity of a molecule for a specific protein target such as an enzyme or a receptor. ACs refer to pairs of molecular compounds whose chemical structures only differ by a small change at a specific site but which exhibit an unexpectedly high difference in their binding affinity for a given pharmacological target [38, 39, 40, 41, 42, 43, 44]. AC-prediction does not only refer to the classification whether a given compound pairs forms and AC or not but usually also implicitly encompasses the prediction of the potency direction (PD) of the pair (i.e. which of both compounds is more active). Molecular property prediction, QSAR-prediction, and AC-prediction represent three important challenges of high relevance to the *in silico* identification and optimisation of novel pharmacological compounds. In particular, our work aims to contribute to the development of computational tools that can accurately predict important properties of novel compounds and whether they will bind tightly to a biological target. This could help to tackle one of the central questions in drug discovery: what should a medicinal chemist synthesise next?

1.1 Outline of Thesis and Research Contributions

The rest of thesis is organised as follows:

- In Chapter 2 we first introduce SMILES strings and molecular graphs which both constitute detailed molecular representations that fully encode the chemical composition and structure of an input compound. We then give a technical introduction to PDVs, ECFPs and GNNs as the three most frequently used molecular featurisation methods in the current literature. We critically discuss all three featurisations and contrast their respective strengths and weaknesses. In the case of GNNs, we also shortly discuss the issue of theoretical expressivity in mathematical terms and give a description of the graph isomorphism network (GIN) as perhaps the simplest and most prototypical GNN in the 1-Weisfeiler-Lehman class.

- In Chapter 3 we conduct a rigorous computational study to explore the relative performance of PDVs, ECFPs and GINs for QSAR and AC-prediction. We investigate nine separate QSAR models by integrating each of the three studied featurisations with a random forest, a multilayer perceptron, and a k-nearest neighbour regressor. We use each model to classify whether a compound pair forms an AC, to classify which compound in the pair is more active, and to predict the binding affinities of individual molecules for three pharmacological targets: dopamine receptor D2, factor Xa, and SARS-CoV-2 main protease. We further develop a novel pair-based data-splitting strategy for the evaluation of distinct AC-prediction scenarios.

  Our results strongly support the hypothesis that QSAR models frequently fail to predict ACs. We observe low AC-sensitivity when the activities of both compounds are unknown, but a substantial increase in AC-sensitivity when the actual activity of one of the compounds is given. GIN features are found to be competitive with or superior to classical molecular featurisations for AC-classification. For general QSAR-prediction, however, ECFPs still consistently deliver the best performance amongs the tested featurisations.

  Our study represents the first work that investigates the capabilities of QSAR models to classify between ACs and non-ACs. Moreover, it provides additional evidence that standard message-passing GNNs may need to be improved further to robustly outperform classical ECFPs. Our study results have been published

as a peer-reviewed research paper [45] in the Journal of Cheminformatics and as a scientific poster [46] at the 10th International Congress on Industrial and Applied Mathematics (ICIAM 2023, Tokyo).

- In Chapter 4 we design and evaluate a novel twin neural network model for AC-prediction. We provide a mathematical description of our proposed dual architecture along with proofs of its built-in symmetry properties. We then conduct computational experiments to compare the AC-prediction performance of the twin architecture and the strongest QSAR models from the previous chapter on a data set of SARS-CoV-2 main protease inhibitors. We explore four distinct molecular featurisations for the twin model, based on either ECFP, GINs, or two transfer learning techniques we developed.

  Our experiments suggest that the twin architecture outperforms standard QSAR models at AC-prediction in a variety of scenarios. In particular, twin networks appear to strike a superior balance between AC-sensitivity and AC-precision which increases their practical utility.

  To the best of our knowledge, this work represents the first application of twin neural networks to AC-prediction and the first investigation of a novel AC-prediction technique that includes important control experiments based on standard QSAR models. We have presented our twin architecture as a scientific poster [47] at the 4th RSC-BMCS / RSC-CICAG Artificial Intelligence in Chemistry Symposium (2021, virtual) where we were subsequently awarded the prize for the best scientific poster.

- In Chapter 5 we introduce a mathematical operation called *substructure pooling* that formalises the transformation of sets of substructures (i.e. unordered set representations of structural fingerprints) into numerical vectors. We draw an analogy between substructure pooling for structural fingerprints and global graph-pooling for GNN-architectures. However, unlike global graph-pooling, substructure pooling remains largely unexplored. We show that the standard hashing procedure for the vectorisation of ECFPs is a form of substructure pooling. We go on to describe a straightforward alternative to hashing for ECFP-vectorisation called *Sort & Slice*. This technique is simply based on first sorting all detected circular substructures according to their frequency of appearance in the training set and then only accepting the most frequent substructures into the final vectorial fingerprint. This naturally leads to a higher

interpretability than hashing due to an absence of bit collisions in the final vector-representation.

Under reasonable theoretical assumptions that hold approximately true for real-world data sets, we mathematically prove that Sort & Slice only selects the most informative substructures from an entropic point of view. Furthermore, we demonstrate via a set of strict computational experiments that Sort & Slice robustly leads to higher predictive performance than hashing as well as two advanced supervised substructure selection schemes for molecular property prediction across a large number of settings. In particular, we observe a predictive advantage of Sort & Slice over hashing across varying data sets, data splitting techniques, machine-learning regressors, and ECFP hyperparameters. This advantage seems to increase with the expected number of bit collisions in the hashed ECFP.

Based on our current knowledge, the work in this chapter constitutes the first study that robustly demonstrates the existence of a technically simple and more interpretable alternative to hashing for the vectorisation of ECFPs that leads to higher predictive performance at supervised molecular property prediction. We have only recently become aware of one other work [48] that has investigated a technique similar to Sort & Slice which we acknowledge. We intend to submit the results in this chapter for publication in the near future.

- In Chapter 6 we shortly present two potential ideas for future research in the domain of molecular featurisation:

  Firstly, we describe a self-supervised graph-based learning strategy that can intuitively be interpreted as a method to make classical molecular featurisations differentiable and trainable. Our central idea involves first training a modern GNN to predict classical featurisations such as ECFPs or PDVs directly from molecular graphs and then fine-tune the pre-trained GNN on a supervised task.

  Secondly, we outline a trainable substructure-pooling method based on a differentiable self-attention mechanism that might enable the learning of contextual substructural features. This technique could for instance be combined with MACCS fingerprints or ECFPs. It could potentially provide a useful inductive bias for the learning of task-specific compound-level featurisations that do not only depend on individually present substructures but also on their interactions.

- In Chapter 7 we give some final conclusions and further thoughts on our work.

Chapter 2

Molecular Representations and Featurisations for Machine Learning

2.1 Overview

In its most general interpretation, *molecular representation* refers to the problem of describing a molecular compound in an abstract machine-readable format for the purpose of computational processing. In this Chapter, we focus on three of the most common types of molecular representation:

1. **Graph-based representation methods** that represent molecules as mathematical graphs with numerical node and edge features.

2. **String-based representation methods** that represent molecules as linear chains of symbols.

3. **Feature-based representation methods** that represent molecules as vectors of real numbers.

There are other types of molecular representations, including image-based representations [49, 50, 51] and three-dimensional coordinate-based representations [52, 53]. However, the above three categories cover a large part of modern use cases in cheminformatics and currently dominate the field of molecular machine learning.

The key advantage of most non-feature-based representations like graphs or strings is that they allow for highly explicit representations of molecular compounds. Such representations are generally designed to fully encode the chemical structure of real molecules and as a result contain high levels of information. However, standard machine learning techniques such as random forests and k-nearest neighbours cannot directly utilise this rich information content because they are only able to process

feature-based representations that consist of vectors of real numbers. One of the central challenges for machine learning in chemistry is thus the developments of *molecular featurisation methods.*

Definition 2.1 (Molecular Featurisation)**.** *A mathematical or algorithmic technique that turns a non-feature-based molecular representation $\mathcal{R}$ (such as a graph or a string) into a feature-based representation $\mathcal{F}$,*

$$\mathcal{R} \mapsto \mathcal{F} = (f_1, ..., f_l) \in \mathbb{R}^l,$$

is called a molecular featurisation method.

The usual purpose of molecular featurisation is to encode useful predictive information within the feature vector $\mathcal{F}$ for a downstream molecular machine learning task such as activity prediction. The chosen molecular featurisation method is often the key ingredient that determines the relative success or failure of a predictive algorithm. Current molecular featurisation methods can be subdivided into two distinct categories: *trainable-* and *non-trainable.* Non-trainable methods are based on fixed non-differentiable algorithms that compute specific predefined properties of an input molecule. The most widely-used examples in this category are physicochemical-descriptor vectors and structural fingerprints. Trainable methods, on the other hand, are based on recently developed deep-learning architectures that can learn to extract molecular features in a differentiable and task-specific manner. At present, message-passing GNNs that operate on top of molecular graphs are the most widely-used method in this class.

We start off this chapter by describing molecular graphs as mathematical representations of chemical compounds. We then go on to discuss SMILES strings which are the prevalent string-based molecular representation in cheminformatics. Finally, we give a technical description and critical discussion of the three most important molecular featurisation methods in the current literature: physicochemical descriptors [14], extended-connectivity fingerprints [16] and message-passing GNNs [17]. In recent years, all three of these competing techniques have led to state-of-the-art results in a wide variety of molecular machine learning tasks [6, 7, 8, 9, 10, 11, 12, 13, 17, 19, 20, 22, 24, 28].

2.2 Molecular Graphs

The molecular graph of a chemical compound is a formal representation of its chemical structure via the language of mathematical graph theory.

Definition 2.2 (Molecular Graph)**.** *Let $\mathcal{M}$ be a molecule composed of n atoms and let*

$$A = \{a_1, ..., a_n\}$$

be a mathematical set of elements representing these atoms. Furthermore, let

$$B = \{\{a, \tilde{a}\} \mid a, \tilde{a} \in A \text{ and there is a chemical bond between } a \text{ and } \tilde{a} \text{ in } \mathcal{M}\}$$

be a set of unordered pairs of elements of A which describe existing chemical bonds between the atoms in $\mathcal{M}$. Then the pair $\mathcal{G} = (A, B)$ is called the molecular graph of $\mathcal{M}$. If all hydrogen atoms are removed from A before constructing $\mathcal{G}$, then $\mathcal{G}$ is called hydrogen-depleted.

Those familiar with the field of graph theory will recognise a molecular graph as a connected, undirected, unweighted graph without self-loops and without multiple edges. The idea to view molecules as abstract networks of linked entitites has deep historical roots and was already employed in the 19th century by the British mathematician Arthur Cayley [54]. Using the hydrogen-depleted version of a molecular graph usually reduces the number of involved atoms significantly while preserving the essential chemical connectivity pattern between the heavy atoms. Hydrogen-depleted molecular graphs are therefore routinely preferred over their complete counterparts, especially in settings where computational cost plays a role. In this work, we too exclusively use hydrogen-depleted molecular graphs unless otherwise specified.

In almost all cases, the pure structural information contained in the molecular graph is enriched via atom and bond feature vectors.

Definition 2.3 (Atom and Bond Feature Vectors)**.** *Let $\mathcal{G} = (A, B)$ be the molecular graph of a molecule $\mathcal{M}$ and let*

$$f : A \to \mathbb{R}^k$$

be a function that assigns real-valued vectors in $\mathbb{R}^k$ to the atoms in A. Then f is called an atom featurisation function. For each atom $a \in A$, the vector $f(a) \in \mathbb{R}^k$ is called its atom feature vector. Similarly, let

$$g : B \to \mathbb{R}^j$$

be a function that assigns real-valued vectors in $\mathbb{R}^j$ to the chemical bonds in B. Then g is called a bond featurisation function. For each chemical bond $\{a, \tilde{a}\} \in B$ the vector $g(\{a, \tilde{a}\}) \in \mathbb{R}^j$ is called its bond feature vector.

Selected Atom and Bond Features for Molecular Graphs		
Atom Feature	**Encoding**	**Dimensions**
element type	one-hot encoding	43
number of heavy neighbours	one-hot encoding	6
number of hydrogen neighbours	one-hot encoding	6
formal charge	one-hot encoding	8
hybridisation type	one-hot encoding	7
R-S chirality type	one-hot encoding	4
is part of a ring	binary integer	1
is aromatic	binary integer	1
atomic mass	min-max scaled real number	1
van der Waals radius	min-max scaled real number	1
covalent radius	min-max scaled real number	1
Bond Feature	-	-
bond type	one-hot encoding	4
E-Z double-bond geometry	one-hot encoding	4
is part of a ring	binary integer	1
is conjugated	binary integer	1

Table 2.1: Overview of the chosen atom and bond features for the molecular graphs used in our experiments.

The purpose of atom and bond feature vectors is to add extra physicochemical information to the molecular graph. At the very least, the atom feature vectors should allow for a distinction between different atomic numbers, i.e. element types

$$\{\text{C, N, O, S, F, Si, Cl, Br,...}\}$$

and the bond feature vectors should allow for a distinction between different bond types

$$\{\text{single, double, triple, aromatic}\}.$$

However, the exact composition of atom and bond feature vectors beyond basic indications of atom and bond types can vary substantially from author to author. For example, Kearnes et al. [19] include tetrahedral R-S chirality in their atom feature vectors while Gilmer et al. [17] do not. Such differences can make an objective comparison of results across studies challenging. Pocha et al. [55] have recently made a

systematic effort to compare the effects of differing atom feature vectors on the performance of a popular GNN-architecture, a graph convolutional network (GCN) [18], in several molecular property prediction tasks. They found that the overall influence of the chosen atom featurisation on downstream performance was modest. However, representations that included more atomic features (and thus more information) tended to deliver better results.

Motivated by this, we decided to extensively leverage the information-storing capacities of molecular graphs in our own work and employ 11 atom and four bond features. An overview of the chosen features that were used throughout our experiments is given in Table 2.1. In our atom featurisation, we considered 42 common element types of heavy atoms, alond with a generic *other-element* type for atoms whose element types were not on our predefined list. Element types were thus represented via 43-dimensional one-hot encoded vectors. Note that while we used the usual hydrogen-depleted form of the molecular graph, we still implicitly considered hydrogen-related information by including the number of hydrogen neighbours in the list of atom features.

2.3 SMILES Strings

The most frequently used method to store molecules in digital form is via Simplified Molecular-Input Line-Entry System (SMILES) strings [56]. A SMILES string is a sequence of ASCII characters which is generated by a fixed set of rules in order to specify the chemical identity of an input molecule. The most common version of the SMILES string can be interpreted as a sequential encoding of a (hydrogen-depleted) molecular graph that takes into account atom and bond types. The SMILES string of a molecule is thus fully sufficient to construct a molecular graph equipped with basic atom and bond features. On the other hand, every molecular graph with basic atom and bond features can be used to generate a SMILES string.

Description 2.1 (SMILES String)**.** A valid SMILES string can be obtained from a molecular graph by turning the graph into a spanning tree via breaking all existing cycles, and then printing out the (capitalised or lower case) atom symbols encountered in a depth-first traversal of the spanning tree. Branches are specified via parentheses while double and triple bonds are written using the symbols $\{=, \#\}$ respectively; the existence of aromatic bonds is either inferred from the context or is expressed via the use of lower case for the involved atoms. The SMILES algorithm for an example

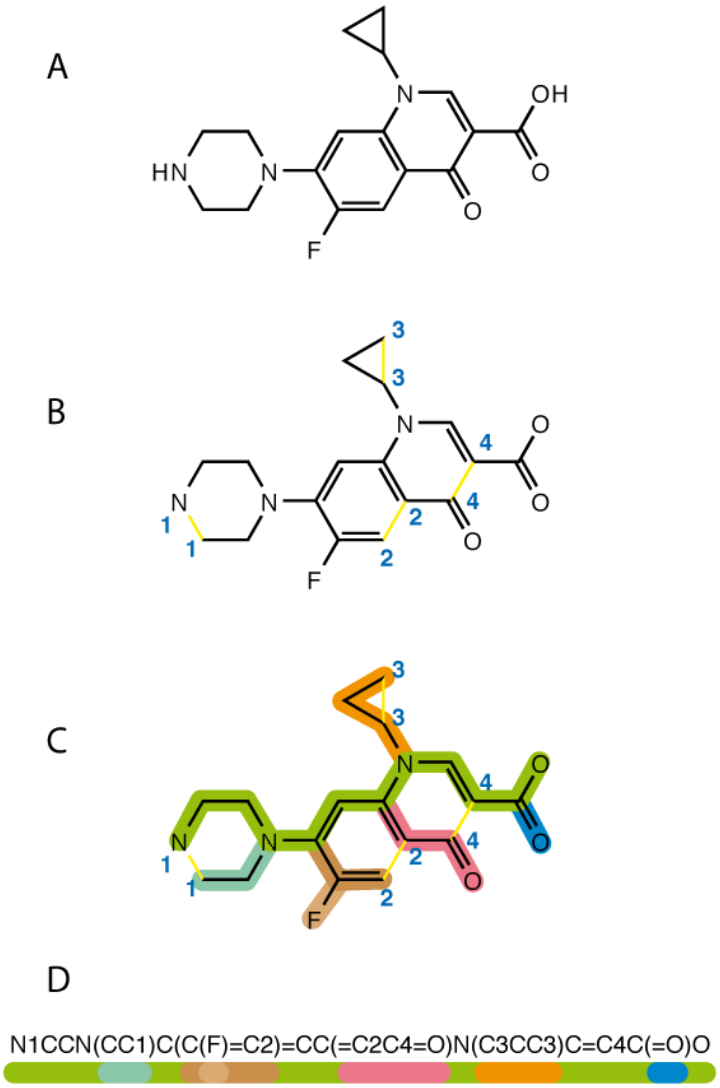

Figure 2.1: Generation of a Simplified Molecular-Input Line-Entry System (SMILES) string from the molecular graph of the antibiotic molecule Ciprofloxacin. First the molecular graph is reduced to its hydrogen-depleted version. Then cycles are broken to turn the graph into a spanning tree. Finally, a depth-first traversal of the spanning tree (here starting with the leftmost nitrogen atom as a root) produces the SMILES string whereby branches are specified via parentheses. The integers in the SMILES string indicate which ring bonds were broken to produce the spanning tree and the equality signs indicate double bonds. Image source: [59].

molecule is illustrated in Figure 2.1. For the exact procedure we refer the reader to the original paper series of Weininger et al. [56, 57, 58].

The SMILES string of a molecule depends on the bonds chosen to break rings, the atom chosen as a starting point for the depth-first traversal of the spanning tree, and the order in which encountered branches are listed. Thus, a molecule can have many distinct and valid SMILES strings. However, canonicalisation algorithms exist that

make the mapping from compound to SMILES string unambiguous. Basic SMILES strings do not contain explicit information about the three-dimensional positioning of the atoms in a molecule, but extensions of the SMILES system such as *isomeric* SMILES strings exist which are able to express tetrahedral R-S chirality and E-Z double-bond geometry. Throughout our work, we exclusively use isomeric SMILES strings that include specifications for these two important stereochemical features.

The SMILES notation has become the default molecular representation in cheminformatics due to its relative simplicity, readability and expressivity. It also allows for the computational storage of large numbers of molecular graphs in a highly compressed and efficient format. However, the SMILES methodology does have some technical drawbacks. For example, since the SMILES string of a molecule is not unique, one has to arbitrarily commit to a fixed canonicalisation algorithms in order to guarantee the consistency of the SMILES representation across applications. In comparison, the international chemical identifier (InChI) [60] string which was initially developed by the International Union of Pure and Applied Chemistry (IUPAC) is more expressive and is guaranteed to produce a unique string-label for every chemical structure. These advantages, however, come at the cost of reduced simplicity and readability.

Another weakness of the SMILES notation is that a slight perturbation of symbols can easily lead to either a syntactically invalid expression or an impossible molecular structure that violates the basic laws of chemistry. This fragility may cause SMILES-based deep generative models trained for *de novo* molecular design to output large fractions of broken and invalid SMILES strings; a challenge that can already be observed in the pioneering work of Gómez-Bombarelli et al. [61]. To address this issue, more robust string representations such as the self-referencing embedded string (SELFIE) notation [62] and the DeepSMILES notation [63] have recently been developed. These string systems have been specifically designed to facilitate the generation of valid molecules by deep generative models and may thus be preferrable to SMILES strings in this context.

2.4 Physicochemical-Descriptor Vectors

2.4.1 Mathematical Description

Physicochemical descriptors [14] are likely the oldest and most established molecular featurisation method and play a fundamental role in classical cheminformatics. Such descriptors are able quantify an abundant number of important characteristics of a

molecule such as lipophilicity, druglikeness, molecular refractivity, electrotopological state, molecular graph-structure, fragment-profile, molecular charge, and molecular surface properties. Descriptor-based molecular featurisations have been used in QSAR-modelling [64], toxicity prediction [65], virtual screening [66], combinatorial chemistry [66], the organisation of latent spaces of deep generative models [61] by chemical properties of interest, and countless other areas of cheminformatics. In the context of our work, we formally define physicochemical-descriptor vectors (PDVs) as follows:

Definition 2.4 (Physicochemical-Descriptor Vector)**.** *Let $\mathcal{R}$ be a non-feature-based molecular representation (such as a SMILES string or a molecular graph). A physicochemical-descriptor function is an algorithmic procedure f^{desc} that transforms $\mathcal{R}$ into a real number $f^{desc}(\mathcal{R}) \in \mathbb{R}$ that quantitatively describes a physicochemical property of the input molecule. A vector $\mathcal{F}$ composed of the outputs of $l \in \mathbb{N}$ distinct physicochemical-descriptor functions,*

$$\mathcal{F} := (f_1^{desc}(\mathcal{R}), ..., f_l^{desc}(\mathcal{R})) \in \mathbb{R}^l,$$

is called a physicochemical-descriptor vector for the molecule represented by $\mathcal{R}$.

Ideally physicochemical descriptors should be interpretable, conceptually simply, and computationally efficient to generate. Moreover, they should preferably give distinct values for distinct input molecules, not require experimental measurements, and vary continuously with continuous changes in molecular structure. Several thousand physicochemical descriptors have been developed in the last decades [67] which fulfill these desired properties to varying degrees. Below we take a closer look at two well-known descriptors that have been frequently used throughout the literature.

Example 2.1 (Predicted $\log(P)$)**.** The 1-octanol-water partition coefficient P of a molecule $\mathcal{M}$ describes the ratio of its concentrations in an (immiscible) mixture of 1-octanol and water at equilibrium:

$$P := \frac{[\mathcal{M}]_{\text{1-octanol}}}{[\mathcal{M}]_{\text{water}}}.$$

Since 1-octanol is a fatty alcohol, the decadic logarithm $\log(P)$ is used as a measure of the lipophilicity of $\mathcal{M}$. Lipophilicity in turn plays an important role in drug discovery since orally active drugs cannot be too lipophilic; the famous Lipinski's rule of five [68] states that a $\log(P)$ over 5 is a risk factor for poor absorption or permeation of an oral drug.

It is common to determine $\log(P)$ experimentally, but it can also be treated as a physicochemical descriptor that can be approximately calculated *in-silico*. A simple method for the computational estimation of $\log(P)$ has been given by Wildman and Crippen [69] who employed a summation over the contributions of individual atoms:

$$\log(P) = \sum_{a \in A} c(f(a)).$$

Here A is a set containing symbolic representations of the atoms in $\mathcal{M}$. The function

$$f : A \to \{1, ..., 68\}$$

assigns each atom to one of 68 developed classes according to properties related to element type, neighbouring atoms and aromaticity. The function

$$c : \{1, ..., 68\} \to \mathbb{R}$$

then maps each atomic class to its additive contribution towards the overall $\log(P)$ of $\mathcal{M}$. The number of atomic classes as well as the functions f and c were constructed and calibrated on the basis of a training set comprising of experimental $\log(P)$-data.

While it might seem simplistic at first sight to try to predict the lipophilicity of a compound merely via a sum over atomic contributions, this method has been shown to provide reasonably accurate and computationally fast $\log(P)$-estimations for many compounds. The intuitive motivation behind the summation of atomic contributions is that the $\log(P)$ may be approximately thought of as reflecting a balance between hydrophobic and hydrophilic parts of a compound whose contributions can be summed up. However, there are limitations to this strategy, and predictions might be less accurate for molecules that significantly deviate from the training set. The Wildmann-Crippen method has been implemented in the `Python` cheminformatics-library `RDKit` [70]. This implementation performs a series of substructure searches on an input compound (usually given via its SMILES string) to assign the appropriate atom classes used for the calculation of the $\log(P)$ estimate.

Example 2.2 (Balaban Index)**.** Let $\mathcal{M}$ be a molecule represented via an hydrogen-depleted molecular graph $\mathcal{G} = (A, B)$ with $|A| := n$ atoms and $|B| := m$ bonds and let $g : B \to \mathbb{R}$ be a scalar bond featurisation function that maps single bonds to 1, double bonds to 1/2, triple bonds to 1/3 and aromatic bonds to 2/3. For $a, \tilde{a} \in A$ let

$$d_g(a, \tilde{a}) := \min \left\{ \sum_{\{a_1, a_2\} \in P} g(\{a_1, a_2\}) \mid P \text{ is a shortest path between } a \text{ and } \tilde{a} \text{ in } B \right\}$$

be the minimal bond-order-weighted graph distance between a and $\tilde{a}$ in $\mathcal{G}$ and let

$$s_a := \sum_{\bar{a} \in A} d_g(a, \bar{a}).$$

Then the Balaban index [71] of $\mathcal{M}$ is defined via

$$J^{\text{bal}}(\mathcal{G}, g) := \frac{m}{m - n + 2} \sum_{\{a, \tilde{a}\} \in B} \frac{1}{\sqrt{s_a s_{\tilde{a}}}}.$$

It may not be straightforward to interpret the quantity $J^{\text{bal}}(\mathcal{G}, g)$ in structural terms. However, it can be shown to be extraordinary useful in detecting the existence of structural differences: for two distinct molecular graphs $\mathcal{G}_1 \neq \mathcal{G}_2$ it almost always holds that

$$J^{\text{bal}}(\mathcal{G}_1, g) \neq J^{\text{bal}}(\mathcal{G}_2, g).$$

The feature $J^{\text{bal}}(\mathcal{G}, g)$ may in some cases encode useful structural information for a downstream prediction task and could then help machine-learning models to detect relevant differences even between very similar input molecules. The Balaban index of a compound can be calculated quickly from its SMILES string using `RDKit` [70].

For the machine-learning tasks in our work, we employed a 200-dimensional PDV composed of a diverse selection of $l = 200$ descriptors that together give a robust overall physicochemical profile of a molecule. Our chosen descriptors are identical to the ones used in the study of Fabian et al. [72] who successfully used a self-supervised transformer-architecture to learn a useful latent embedding of chemical space via the prediction of physicochemical properties from SMILES strings of chemical compounds. Each selected descriptor can be computed quickly from the compound SMILES-string using the `Python` cheminformatics-package `RDKit` [70]. A full list of our chosen descriptors, which includes the predicted $\log(P)$ from Example 2.1 and the Balaban index from Example 2.2, can be found in Table 2.2.

Before using the PDV for any prediction task, we canonically derived the cumulative distribution function for each individual descriptor from the training set and used it to normalise the associated descriptor-values. The final PDV $\mathcal{F}$ was thus always contained in the hypercube $[0, 1]^{200}$. This normalisation-step prevents certain machine-learning algorithms such as multilayer perceptrons from putting disproportionate attention on large-range features while ignoring small-range features.

Selected `RDKit` Physicochemical Descriptors
BalabanJ, BertzCT, Chi0, Chi0n, Chi0v, Chi1, Chi1n, Chi1v, Chi2n, Chi2v, Chi3n, Chi3v, Chi4n, Chi4v, EState_VSA1, EState_VSA10, EState_VSA11, EState_VSA2, EState_VSA3, EState_VSA4, EState_VSA5, EState_VSA6, EState_VSA7, EState_VSA8, EState_VSA9, ExactMolWt, FpDensityMorgan1, FpDensityMorgan2, FpDensityMorgan3, FractionCSP3, HallKierAlpha, HeavyAtomCount, HeavyAtomMolWt, Ipc, Kappa1, Kappa2, Kappa3, LabuteASA, MaxAbsEStateIndex, MaxAbsPartialCharge, MaxEStateIndex, MaxPartialCharge, MinAbsEStateIndex, MinAbsPartialCharge, MinEStateIndex, MinPartialCharge, MolLogP, MolMR, MolWt, NHOHCount, NOCount, NumAliphaticCarbocycles, NumAliphaticHeterocycles, NumAliphaticRings, NumAromaticCarbocycles, NumAromaticHeterocycles, NumAromaticRings, NumHAcceptors, NumHDonors, NumHeteroatoms, NumRadicalElectrons, NumRotatableBonds, NumSaturatedCarbocycles, NumSaturatedHeterocycles, NumSaturatedRings, NumValenceElectrons, PEOE_VSA1, PEOE_VSA10, PEOE_VSA11, PEOE_VSA12, PEOE_VSA13, PEOE_VSA14, PEOE_VSA2, PEOE_VSA3, PEOE_VSA4, PEOE_VSA5, PEOE_VSA6, PEOE_VSA7, PEOE_VSA8, PEOE_VSA9, RingCount, SMR_VSA1, SMR_VSA10, SMR_VSA2, SMR_VSA3, SMR_VSA4, SMR_VSA5, SMR_VSA6, SMR_VSA7, SMR_VSA8, SMR_VSA9, SlogP_VSA1, SlogP_VSA10, SlogP_VSA11, SlogP_VSA12, SlogP_VSA2, SlogP_VSA3, SlogP_VSA4, SlogP_VSA5, SlogP_VSA6, SlogP_VSA7, SlogP_VSA8, SlogP_VSA9, TPSA, VSA_EState1, VSA_EState10, VSA_EState2, VSA_EState3, VSA_EState4, VSA_EState5, VSA_EState6, VSA_EState7, VSA_EState8, VSA_EState9, fr_Al_COO, fr_Al_OH, fr_Al_OH_noTert, fr_ArN, fr_Ar_COO, fr_Ar_N, fr_Ar_NH, fr_Ar_OH, fr_COO, fr_COO2, fr_C_O, fr_C_O_noCOO, fr_C_S, fr_HOCCN, fr_Imine, fr_NH0, fr_NH1, fr_NH2, fr_N_O, fr_Ndealkylation1, fr_Ndealkylation2, fr_Nhpyrrole, fr_SH, fr_aldehyde, fr_alkyl_carbamate, fr_alkyl_halide, fr_allylic_oxid, fr_amide, fr_amidine, fr_aniline, fr_aryl_methyl, fr_azide, fr_azo, fr_barbitur, fr_benzene, fr_benzodiazepine, fr_bicyclic, fr_diazo, fr_dihydropyridine, fr_epoxide, fr_ester, fr_ether, fr_furan, fr_guanido, fr_halogen, fr_hdrzine, fr_hdrzone, fr_imidazole, fr_imide, fr_isocyan, fr_isothiocyan, fr_ketone, fr_ketone_Topliss, fr_lactam, fr_lactone, fr_methoxy, fr_morpholine, fr_nitrile, fr_nitro, fr_nitro_arom, fr_nitro_arom_nonortho, fr_nitroso, fr_oxazole, fr_oxime, fr_para_hydroxylation, fr_phenol, fr_phenol_noOrthoHbond, fr_phos_acid, fr_phos_ester, fr_piperdine, fr_piperzine, fr_priamide, fr_prisulfonamd, fr_pyridine, fr_quatN, fr_sulfide, fr_sulfonamd, fr_sulfone, fr_term_acetylene, fr_tetrazole, fr_thiazole, fr_thiocyan, fr_thiophene, fr_unbrch_alkane, fr_urea, qed

Table 2.2: Overview of the 200 distinct molecular descriptors selected for the 200-dimensional physicochemical-descriptor vectors (PDVs) used in our experiments. Each descriptor is named after its associated command in `RDKit` [70]. The descriptors are identical to the ones used in the work of Fabian et al. [72] and cover a broad spectrum of molecular properties related to lipophilicity, druglikeness, electrotopological state, molecular refractivity, molecular surface, molecular-graph structure, charge, and fragment count.

2.4.2 Critical View

Below we give a list of advantages (+) and disadvantages (−) of PDVs as a molecular featurisation method.

(+) **Low computational cost.** PDVs can be computed quickly, even for large molecular data sets.

(+) **Interpretability (in some cases).** Certain descriptors quantify straightforward physicochemical properties of a compound such as its molecular weight or its heavy atom count. Such descriptors can be immediately understood and interpreted by chemical experts. However, it is important to note that there also exists a large number of potentially useful descriptors whose chemical interpretation is not obvious at all, such as the Balaban index discussed in Example 2.2.

(+) **Simplicity of implementation.** PDVs are easy to use and can be automatically generated without sophisticated technical knowledge via publicly available cheminformatics libraries such as the `Python`-package `RDKit` [70].

(+) **Chemically reasonable initial embedding of chemical space.** PDVs tend to automatically calculate basic physicochemical features of molecules and thus immediately provide an *a priori* embedding of chemical space that is useful across a wide range of cheminformatics tasks. In particular, this means that basic chemical features that are relevant across many applications do not need to be inferred statistically from a molecular representation from scratch every time a model is trained on a new data set. This can be seen in contrast to trainable methods like GNNs which continuously need to relearn a reasonable embedding of chemical space at every new training cycle (unless they have been combined with suitable self-supervised or transfer learning approaches).

(+) **Low dimensionality.** PDVs with a dimensionality $l \leq 200$ often already provide useful feature vectors for molecular machine learning tasks. This can be seen in contrast to extended-connectivity fingerprints (ECFPs) which we will introduce in Section 2.5 and which normally require a length of at least $l \geq 1024$ to reach acceptable performance. The relatively low dimensionality of PDVs can decrease the risk of costly downstream-computations and overfitting.

(+) **Global receptive field.** While ECFPs and message-passing GNNs, which we will both introduce in Section 2.5 and Section 2.6, are only able to extract

features from local circular substructures, PDVs can directly express global high-level properties of a molecule such as its predicted $\log(P)$.

(−) **Non-differentiability.** PDVs are generally based on fixed algorithms that cannot be trained in a differentiable manner to learn features from more explicit molecular representations. In this particular sense we refer to PDVs as non-trainable. It is worth noting though that computed PDVs can still be adapted to a given data set to some extent via the application of additional downstream methods such as data-dependent feature selection and normalisation procedures (like the removal of low-variance or correlated descriptors). Strictly speaking such common procedures can certainly also be considered forms of training (i.e. forms or learning from input data). The non-differentiability of PDVs might cause an information bottleneck through which important chemical information cannot pass.

(−) **Necessity for feature selection.** Since PDVs are not differentiable and normally do not contain tunable hyperparameters, one of the few ways to adapt them to a given prediction task is via the choice of physicochemical-descriptor functions that compose the final PDV. However, the optimal descriptor-choice is almost never obvious and usually requires the application of expert-knowledge or technical feature selection algorithms.

(−) **Finite number of descriptors.** Trainable molecular featurisation methods like message-passing GNNs are parametric families of functions that can represent an infinite number of potential feature mappings into a continuous real vector space. The composition of a PDV on the other hand is restricted to a finite number of a few thousand human-engineered non-trainable non-tunable physicochemical-descriptor functions. As a result, PDVs can only represent a finite number of distinct feature mappings defined by the initial descriptor-choices. This finite family of feature mappings might not always be sufficient to describe all of the relevant information for a chemical prediction task; a suitable descriptor-function for the problem might simply not be available.

2.5 Extended-Connectivity Fingerprints

2.5.1 A Short Background on Structural Fingerprints

In this section, we will discuss the extended-connectivity fingerprint (ECFP) [16, 73], which is part of a larger family of molecular featurisation methods called *structural fingerprints*. The essential idea behind structural fingerprints is to algorithmically detect and encode features that express parts of the chemical structure of a molecule. Often, structural fingerprints can be represented as bit vectors that express the presence or absence of certain chemical substructures within the input compound.

Well-known structural fingerprints include the 166-bit MACCS fingerprint [74] and the 881-bit PubChem fingerprint [75, 76]. Both of these examples belong to a family of *dictionary-based* fingerprints that check the presence or absence of substructures from a predefined dictionary. The dictionaries underlying MACCS and PubChem fingerprints include 166 and 881 unique substructures respectively, which explains the dimensionalities of these featurisations.

Another important family of structural fingerprints is given by *enumeration-based* fingerprints. These fingerprints exhaustively enumerate all substructures of a molecule that can be detected via a certain search strategy. An example are Daylight fingerprints [77] which enumerate all substructures within an input compound that correspond to groups of atoms and bonds that form paths up to a chosen length. As we will see in this section, ECFPs too fall into the category of enumeration-based fingerprints since they enumerate all circular substructures in a molecule up to a chosen radius. Because enumeration-based fingerprints are not constrained by a fixed-sized dictionary of substructures, they can often detect an extremely large number of distinct fragments. It is thus not immediately obvious how the substructures found within a particular compound should be transformed into a reasonably-sized bit-vector. The standard technique to address this issue is by using a hash function to fold the detected fragments into a bit-vector representation of feasible dimensionality and accept the fact that colliding bits might lead to a certain level of ambiguity. In Chapter 5 and in Section 6.2, we will explore some interesting alternatives to this hashing procedure for the vectorisation of structural fingerprints.

In our work, we focus on ECFPs out of all available structural fingerprints. Our reasons for this include that ECFPs have been used widely in the area of cheminformatics and molecular machine-learning and are well-known to perform well on a diverse set of tasks [17, 20, 78, 79, 80, 81]. Moreover, as we will discover in Section 2.6,

ECFPs also exhibit some striking similarities with modern message-passing GNN architectures. In some sense, ECFPs can be interpreted as a non-differentiable version of message-passing GNNs, which makes a comparison between these two approaches interesting. We will use the rest of this section to give a mathematical description of ECFPs along with a critical discussion of their strengths and weaknesses.

2.5.2 Mathematical Description

The ECFP algorithm is a molecular featurisation method that maps a SMILES string (or its associated molecular graph) to a structural fingerprint that can be represented as a bit vector. The fundamental ideas underlying ECFPs were originally introduced by Morgan [73] in 1965; however, modern implementations of the ECFP are largely based on a detailed technical description given by Rogers and Hahn [16] in 2010. ECFPs provide a simple yet powerful technique to encode information about the chemical structure of an input compound. The algorithm is dependent on two predefined hyperparameters: the desired fingerprint length $l \in \mathbb{N}$ and the maximum radius $R \in \mathbb{N}_0$ of the receptive field. An ECFP of length l takes the form of a binary vector

$$\mathcal{F} = (f_1, ..., f_l) \in \{0, 1\}^l.$$

Up to a certain level of ambiguity due to bit collisions which we will discuss below, each component f_i in $\mathcal{F}$ is associated with the presence or absence of a particular *circular subgraph*, equipped with specific atom and bond features inherited from the input molecule, centered around a given atom.

Definition 2.5 (Circular Subgraph)**.** *Let $\mathcal{G} = (A, B)$ be a molecular graph with atom set A and bond set B and let $a \in A$. Denote with*

$$N(a) \coloneqq \{\tilde{a} \in A \mid \{\tilde{a}, a\} \in B\}$$

the set of neighbouring atoms of a and denote with

$$M(a) \coloneqq \{\{a_1, a_2\} \in B \mid a \in \{a_1, a_2\}\}$$

the set of bonds that are attached to a. Now let (A_r^a, B_r^a) be a sequence of subgraphs of (A, B) for $r \in \mathbb{N}_0$ constructed via the following iterative scheme:

$$(A_0^a, B_0^a) \coloneqq (\{a\}, \{\}),$$

$$A_r^a = A_{r-1}^a \cup \bigcup_{\tilde{a} \in A_{r-1}^a} N(\tilde{a}), \quad B_r^a = B_{r-1}^a \cup \bigcup_{\tilde{a} \in A_{r-1}^a} M(\tilde{a}).$$

Then (A_r^a, B_r^a) *is called the circular subgraph of* $\mathcal{G}$ *with center atom* a *and radius* r*. If* $\mathcal{G}$ *is equipped with atom and bond feature vectors, then these are inherited by* (A_r^a, B_r^a)*.*

Circular subgraphs with varying radii for an example compound are depicted in Figure 2.2. The fingerprint hyperparameter R defines the maximum radius of any

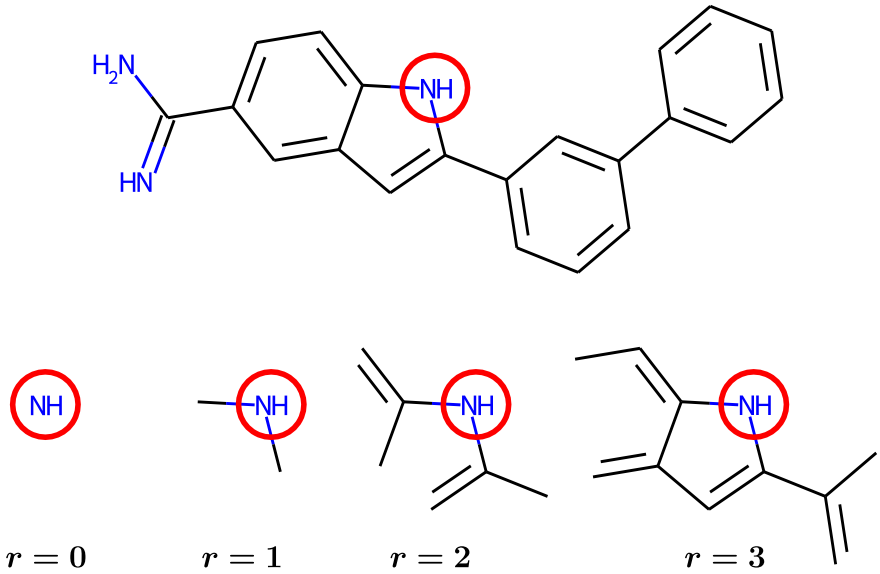

Figure 2.2: Circular subgraphs of varying radii for a central nitrogen atom in an example molecule.

circular subgraph whose presence or absence is indicated in the fingerprint $\mathcal{F}$. Circular subgraphs that are structurally isomorphic are further distinguished according to their inherited atom and bond features, i.e. two structurally isomorphic circular subgraphs with distinct atom or bond features correspond to different components of $\mathcal{F}$. For chemical bonds, this distinction is made on the basis of simple bond types: single, double, triple, or aromatic. To distinguish atoms, ECFPs usually either use standard or pharmacophoric atom features and we discuss both variants below.

There also exists a non-binary version of the ECFP in which each component of $\mathcal{F}$ does not simply indicate the presence or absence of a circular subgraph, but the exact number of occurrences of the subgraph within the input compound. This version is sometimes referred to as *ECFP with counts* as opposed to the binary version mentioned above which is also called *ECFP without counts*. Throughout this thesis, unless specifically stated otherwise, we focus on ECFPs without counts, as they are used significantly more frequently in practice and allow for an easier algorithmic description. However, it is straightforward to extend the methods described in this section to ECFPs with counts.

We now give a technical description of the algorithm used for the generation of ECFPs as specified in the article of Rogers and Hahn [16].

ECFP algorithm

List of inputs:

- A molecular compound $\mathcal{M}$, usually represented via a SMILES string $\mathcal{S}$. Note that $\mathcal{S}$ contains the structural information of the molecular graph $\mathcal{G} = (A, B)$.
- A function f specifying the atom feature vectors for A using either standard or pharmacophoric features.
- A function g specifying the bond types for B using encodings for labels in the set {single, double, triple, aromatic}.
- A fingerprint length $l \in \mathbb{N}$, often chosen to be in $\{1024, 2048\}$.
- A fingerprint radius $R \in \mathbb{N}_0$, often chosen to be in $\{1, 2, 3\}$.

Initialisation: A deterministic hash function h is chosen which maps integer vectors of arbitrary lengths into a large index set such as

$$\{1, ..., 2^{32}\}$$

in a pseudo-random and uniformly distributed manner. The map h is then used to hash each atom feature vector $f(a)$ to a single integer $h(f(a)) \in \{1, ..., 2^{32}\}$. The integers in the set

$$I_0 \coloneqq \{h(f(a)) \mid a \in A\} \subseteq \{1, ..., 2^{32}\}$$

are called *initial atom identifiers.*

Iterative phase for substructure-enumeration: The initialisation step is followed by R iterative steps, whereby at each step all atom identifiers are updated and saved. At each step $r \in \{1, ..., R\}$, each atom $a \in A$ first starts off equipped with its respective atom identifier in I_{r-1}. The new integer atom identifier for a is generated as follows:

1. A vector of integers $J_{a,r}$ is created that contains the integer r in the first position and the current atom identifier of a in the second position.

2. All neighbouring atoms that have a bond to a are ordered in lexicographic order according to their bond types (single = 1, double = 2, triple = 3, aromatic = 4) and then according to their current integer atom identifiers.

3. A loop is performed over all neighbouring atoms according to this established order; for each neighbouring atom, first the bond type and then the current atom identifier is appended to the end of the integer vector $J_{a,r}$.

4. The integer $h(J_{a,r}) \in \{1, ..., 2^{32}\}$ is computed and interpreted as the new atom identifier for a.

After the set of new atom identifiers

$$I_r := \{h(J_{a,r}) \mid a \in A\} \subseteq \{1, ..., 2^{32}\}$$

has been computed, the current atom identifiers are simultaneously updated to correspond to the elements in I_r. Note that by construction each atom identifier in I_r represents an integer label that can be mapped to a particular circular subgraph with radius r and associated atom and bond features. Thus, if two circular subgraphs receive the same integer label they can be assumed to be isomorphic (ignoring rare ambiguities caused by possible hash collision that could lead to two distinct circular subgraphs having the same integer label).

After the completion of R iterations, the atom identifiers from all iterative stages are collected in one set:

$$\mathcal{I} := \bigcup_{r=0}^{R} I_r \subseteq \{1, ..., 2^{32}\}.$$

Structural-duplicate removal: Note that it is possible for several distinct atom identifiers in $\mathcal{I}$ to correspond to the same circular subgraph, for example when an iteratively constructed circular subgraph can be traced back to two or more possible center atoms. To eliminate this redundancy, all but one of the structural-duplicate-identifiers are systematically removed from $\mathcal{I}$ using a method based on sets of bond features. For sake of brevity, further details of this technical removal process are omitted from this description, but full details can be found in the article from Rogers and Hahn [16].

Creation of hashed vectorial fingerprint: After the elimination of all structural duplicates in $\mathcal{I}$, yet another (arbitrary) standard hash function

$$\tilde{h} : \{1, ..., 2^{32}\} \to \{1, ..., l\}$$

is chosen to map the integers in $\mathcal{I}$ into the much smaller set

$$\{1, ..., l\}.$$

This folding procedure creates a fingerprint-set

$$F := \{\tilde{h}(i) \mid i \in \mathcal{I}\} \subseteq \{1, ..., l\}.$$

Finally, F is transformed into a binary fingerprint-vector

$$\mathcal{F} := (f_1, ..., f_l) \in \{0, 1\}^l$$

by setting

$$f_i := \begin{cases} 1 & i \in F, \\ 0 & i \notin F, \end{cases} \qquad \forall i \in \{1, ..., l\}.$$

The total number of detected substructures in a chemical data set naturally increases with the fingerprint radius R. Note that the larger the fingerprint dimension l gets compared to the number of detected substructures (as controlled by R), the more likely it becomes that there are no bit collisions. In other words, the more likely it becomes that each dimensional component f_i of $\mathcal{F}$ informs about the presence or absence of one particular atom identifier in $\mathcal{I}$ and thus about the presence or absence of one unambiguous circular subgraph with atom and bond features within the input compound (ignoring rare ambiguities where two distinct circular subgraphs end up with the same atom identifier due to hash collisions). The bit f_i is then set to 1 if and only if the circular substructure is present anywhere in the molecule, otherwise f_i is set to 0. However, if l becomes small relative to the number of detected substructures then more and more hash collisions start to occur, which degrades the quality of the fingerprint. Such hash collisions cause a fingerprint-component f_i to become ambiguous and correspond to one out of several possible atom identifiers and thus to distinct circular substructures. Therefore, l must be chosen sufficiently large as to guarantee the expressivity of the ECFP.

In the literature, ECFP-featurisations with radius R are often written in the form ECFP2R with $2R$ being the fingerprint diameter. For example, the frequently used 1024-bit ECFP4-featurisation describes an ECFP with radius $R = 2$ and length $l = 1024$. In our work, we used the `Python` cheminformatics-library `RDKit` [70] to generate ECFPs from SMILES strings.

2.5.3 Standard and Pharmacophoric Atom Features

To distinguish atoms, ECFPs as implemented in `RDKit` [70] use six standard features. Optionally, the algorithm also allows for the stereochemical distinction between atoms with respect to tetrahedral R-S chirality. There also exist alternative binary atom

features that were designed to be more reflective of the abstract function that an atom might play in pharmacological chemistry. When these *pharmacophoric* atom features [16] are used instead of the standard features, then one speaks of functional-connectivity fingerprints (FCFPs). The standard and pharmacophoric atom features for the ECFP-algorithm are listed in Table 2.3.

Standard Atom Features	**Pharmacophoric Atom Features**
Atomic number	Hydrogen-bond acceptor (yes/no)
Total degree	Hydrogen-bond donor (yes/no)
Number of hydrogen neighbours	Negatively ionisable (yes/no)
Formal charge	Positively ionisable (yes/no)
Isotope	Aromatic (yes/no)
Part of a ring (yes/no)	Halogen (yes/no)
Optional: tetrahedral R-S chirality	Optional: tetrahedral R-S chirality

Table 2.3: Standard and pharmacophoric atom features used for the two versions of the ECFP algorithm.

As can be seen, there is an overlap between the standard atom features for ECFPs and the atom features in our molecular graphs. In certain molecular machine learning applications, replacing standard with pharmacophoric atom features might lead to increased performance and decreased learning time since important high-level atomic properties are presented to the learning model from the start and do not need to be inferred statistically. However, standard atom features contain more detailed information that could still be relevant for the prediction task and thus be leveraged by the learning algorithm.

2.5.4 Critical View

Below we give a list of advantages (+) and disadvantages (−) of the ECFP-algorithm as a molecular featurisation method.

(+) **Low computational cost.** ECFPs can be computed rapidly, even for large molecular data sets.

(+) **Interpretability.** Since ECFPs contain information about the presence or absence of concrete chemical substructures (up to hash collisions) they can often be understood and interpreted in a straightforward manner.

(+) **Simplicity of implementation.** ECFPs are easy to use and can be automatically generated without sophisticated technical knowledge via publicly available cheminformatics libraries such as the `Python`-package `RDKit` [70].

(+) **Chemically reasonable initial embedding of chemical space.** ECFPs automatically express basic structural features of molecules and thus immediately provide an *a priori* embedding of chemical space that is useful across a wide range of cheminformatics tasks. In particular, this implies that basic structural features that are important across many applications do not need to be relearned from a molecular representation from scratch every time a model is trained on a new data set. This can be seen in contrast to trainable methods like GNNs which continuously need to relearn a reasonable embedding of chemical space at every new training cycle (unless they have been combined with suitable self-supervised or transfer learning approaches).

(+) **Arbitrary circular subgraphs.** ECFPs do not simply check the existence of substructures from a predefined finite list of chemical substructures (like dictionary-based structural fingerprints do), but are able to distinguish between an essentially infinite number of chemical subgraphs, albeit only circular ones.

(−) **Non-differentiability.** After its hyperparameters have been chosen, the ECFP-transformation becomes a fixed method that produces the same task-agnostic features for a given compound across data sets. In particular, ECFPs cannot be trained in a differentiable manner to learn features from more explicit molecular representations. In this particular sense we refer to ECFPs as non-trainable. It should be noted though that computed ECFPs can nevertheless be adapted to a given data set to some degree via the application of additional downstream methods such as data-dependent feature selection and normalisation procedures; strictly speaking such procedures can certainly also be considered forms of training (i.e. forms or learning from input data) and we will explore some of these techniques in Chapter 5. The non-differentiability of ECFPs might cause an information bottleneck through which important chemical information cannot pass.

(−) **Trade-off between dimensionality and hash collisions.** Decreasing the fingerprint dimension l relative to the number of detected substructures increases the number of hash collisions. These collisions cause a loss of information

and interpretability due to an increasing inability of the fingerprint to distinguish between non-identical circular substructures. Thus, lengths of $l \geq 1024$ are often a necessity to reach an acceptable performance. In a machine learning setting, this high dimensionality can lead to costly downstream-computations and increased risk of overfitting.

(−) **Locality of receptive field.** ECFPs are based on a local neighbourhood-aggregation scheme for atoms in a molecule. By design, they are thus only capable of indicating the existence of local circular subgraphs. In particular, they cannot directly express global properties of a molecule.

2.6 Message-Passing Graph Neural Networks

2.6.1 Mathematical Description

In 2015, Duvenaud et al. [20] published an influential article where they proposed an adaptive counterpart to ECFPs. Their goal was to overcome some of the inherent limitations of the ECFP-featurisation such as non-differentiability and high dimensionality. Their work resulted in a trainable GNN-architecture that could directly process molecular graphs of varying sizes as input; they reported competitive performance of their method relative to classical ECFPs at several canonical molecular prediction tasks. After this initial step, a multitude of other promising GNN-architectures appeared rapidly within the molecular machine-learning community [18, 19, 21, 22, 23, 24, 25, 26, 27, 28]. In a seminal article from 2017, Gilmer et al. [17] managed to subsume almost all modern GNN-architectures under an overarching mathematical framework which was further generalised and brought into its modern form by Bronstein et al. [82, 83]. Below we give a brief technical description of this framework known as *message-passing*.

Message-passing-GNN algorithm

List of inputs:

- A chemical compound $\mathcal{M}$, represented via its molecular graph $\mathcal{G} = (A, B)$.
- An function f specifying the atom feature vectors for A.

- A function g specifying the bond feature vectors for B.
- A fingerprint length $l \in \mathbb{N}$, often chosen to be in $\{50, ..., 500\}$.
- A fingerprint radius $R \in \mathbb{N}_0$, often chosen to be in $\{1, 2, 3, 4, 5\}$.

Message-passing phase: The first part of the algorithm consists of a message-passing phase in which the atom feature vectors of $\mathcal{G}$ are iteratively updated over $r \in \{1, ..., R\}$ steps via a local neighbourhood-aggregation scheme. We denote the set of neighbours of an atom $a \in A$ as

$$N(a) \coloneqq \{\tilde{a} \in A : \{a, \tilde{a}\} \in B\}.$$

At each step r, every atom $a \in A$ updates its associated feature vector from $f_{r-1}(a)$ to $f_r(a)$ according to the following recursive relations:

$$f_0(a) \coloneqq f(a),$$

$$m_r(a) = \bigoplus \{w_r(f_{r-1}(a), f_{r-1}(\tilde{a}), g(\{a, \tilde{a}\})) \mid \tilde{a} \in N(a)\}_{\text{mul}}, \tag{2.1}$$

$$f_r(a) = u_r(f_{r-1}(a), m_r(a)).$$

The vector-valued functions $(w_r)_{r=1}^R$ are called message-passing functions and the vector-valued functions $(u_r)_{r=1}^R$ are called atom-updating functions; both types of maps frequently contain trainable neural networks. The vector $m_r(a)$ can be interpreted as an aggregated message that the atom a receives at step r from neighbouring atoms along its associated chemical bonds to update its current feature vector.

Multisets, i.e. sets that can contain multiple instances of the same element, denoted via $\{\}_{\text{mul}}$ instead of $\{\}$, are required in the above recursion. This is because two atom neighbours $\tilde{a}_1, \tilde{a}_2$ of a could in theory produce identical messages

$$w_r(f_{r-1}(a), f_{r-1}(\tilde{a}_1), g(\{a, \tilde{a}_1\})) = w_r(f_{r-1}(a), f_{r-1}(\tilde{a}_2), g(\{a, \tilde{a}_2\})),$$

but two such messages should still be counted as separate elements.

The $\bigoplus$-symbol represents a placeholder for a vector-valued and permutation-invariant multiset-function. Note that permutation-invariance is a basic requirement that each multiset functions must possess in order to be a well-defined function in the first place. The permutation-invariance is necessary to guarantee that the aggregated message $m_r(a)$ is invariant under any (arbitrary) ordering imposed on the neighbours of a. Possible operators for $\bigoplus$ include summation, averaging, or the componentwise computation of maxima. In most cases, however, the sum-operator is selected,

$$\bigoplus \coloneqq \sum, \tag{2.2}$$

and we assume this choice by default unless stated otherwise.

The multiset of atom feature vectors at a particular step $r \in \{0, ..., R\}$,

$$f_r(A)_{\text{mul}} \coloneqq \{f_r(a) \mid a \in A\}_{\text{mul}},$$

is often imagined to be located at the r-th layer of a multilayer graph with $R+1$ layers. Note that the multiset of bond feature vectors

$$g(B)_{\text{mul}} \coloneqq \{g(\{a, \tilde{a}\}) \mid \{a, \tilde{a}\} \in B\}_{\text{mul}}$$

usually does not get updated, although this rule was successfully broken by Kearnes et al. [19].

Global-pooling step: The message-passing phase is followed by a *global-pooling* step at which the multisets of computed atom feature vectors $(f_r(A)_{\text{mul}})_{r=0}^{R}$ are summarised to a single l-dimensional vectorial representation of $\mathcal{G}$. In the most general case, this pooling operation is accomplished via the application of a sequence of vector-valued multiset-functions $(\tilde{\bigoplus}_r)_{r=0}^{R}$ along the sequence of graph layers. The layerwise outputs can then be combined via another vector-valued function q to a final graph-level fingerprint $\mathcal{F}$ of length l. In mathematical terms the global pooling operation thus takes the following general form:

$$q(\tilde{\bigoplus}_0 f_0(A)_{\text{mul}}, ..., \tilde{\bigoplus}_R f_R(A)_{\text{mul}}) =: \mathcal{F} \in \mathbb{R}^l. \tag{2.3}$$

Similar to before, the pooling maps $(\tilde{\bigoplus}_r)_{r=0}^{R}$ are chosen to be permutation-invariant multiset functions as to guarantee their invariance under any (arbitrary) atom ordering imposed on the input graph $\mathcal{G}$ and to guarantee that multiple instances of identical atom feature vectors are treated as separate elements. Often each $\tilde{\bigoplus}_r$ simply represents a summation or averaging operator, or the componentwise computation of maxima, although more powerful and expressive pooling maps exist. An overview of the total molecular-featurisation process via message-passing GNNs is depicted in Figure 2.3.

We can see that message-passing GNNs exhibit some striking similarities with ECFPs. The fingerprint radius R of a GNN defines the maximum diameter of its circular receptive field and plays an analogous role to the hyperparameter R of the same name for ECFPs. In both featurisation methods a local neighbourhood-aggregation scheme is applied to iteratively update the atom feature vectors of the molecular graph. Unlike ECFPs, however, message-passing GNNs can learn from graph-shaped input data in a differentiable manner: the functions $(w_r)_{r=1}^{R}, (u_r)_{r=1}^{R}$ normally contain a trainable deep-learning component. The graph-level feature vector $\mathcal{F}$ can be fed into a neural prediction head (such as a multilayer perceptron) which enables the resulting end-to-end architecture to train its weights on a supervised task via backpropagation

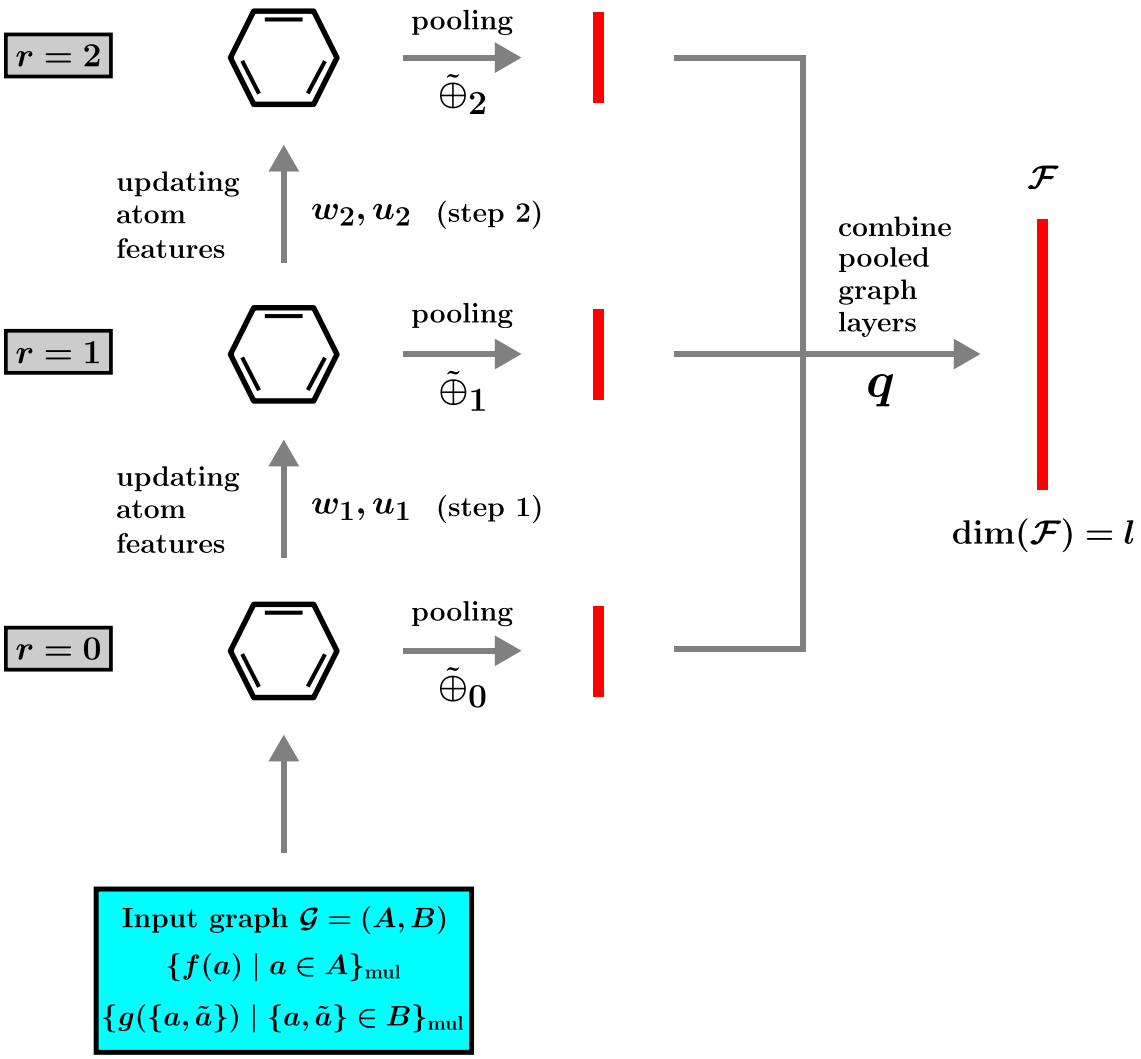

Figure 2.3: Schematic overview of the molecular-featurisation mechanism of a message-passing graph neural network (GNN) with radius $R = 2$. All depicted functions may contain trainable deep-learning components.

and standard gradient-based optimisation algorithms.

2.6.2 Graph Convolutional Networks

We now describe an early GNN-model that has been used frequently in the literature due to its relative simplicity and computational efficiency: the graph convolutional network (GCN) introduced in 2016 by Kipf and Welling [18]. Let

- $S \in \{0, 1\}^{n \times n}$ be the adjacency matrix of a molecular graph $\mathcal{G} = (A, B)$ with atoms $A = \{a_1, ..., a_n\}$,
- $\bar{S} := S + I$ be a modified version of the adjacency matrix with 1s on the diagonal

(here I is the n-dimensional identity-matrix which is added to guarantee self-loops in the graph so that during the node-feature updating-process each node does not only take into account the features of its neighbouring nodes but also its own features),

- $\bar{D}$ be the diagonal degree-matrix of $\bar{S}$ defined via $\bar{D}_{i,i} := \sum_{j=1}^{n} \bar{S}_{i,j}$ and $\bar{D}_{i,j} = 0$ for $i \neq j$,
- $F_r \in \mathbb{R}^{n \times k}$ be a matrix whose rows contain the transposed atom feature vectors $(f_r(a_i)^T)_{i=1}^{n}$ at step r,
- $W_r \in \mathbb{R}^{k \times l}$ be a matrix of trainable weights associated with the r-th graph layer, and
- $\mathrm{ReLU}(x) := \max\{0, x\}$ be the well-known rectified linear unit (ReLU) activation-function.

Then the atom feature vector updates in a GCN are governed by the following dynamics:

$$F_r := \mathrm{ReLU}(\bar{D}^{-\frac{1}{2}} \bar{S} \bar{D}^{-\frac{1}{2}} F_{r-1} W_{r-1}). \tag{2.4}$$

This updating rule can be motivated by a first-order approximation of localised spectral filters on graphs. For details on this derivation, we refer the reader to the original article [18]. We now show that GCNs are indeed message-passing GNNs.

Proposition 2.1 (Message-Passing for GCNs)**.** *The GCN-model falls into the class of message-passing GNNs, i.e. the atom feature vector updating process of GCNs can be mathematically expressed via the message-passing scheme described in Equations 2.1.*

Proof. Note that for each $a_i \in A$ the associated entry of the diagonal matrix $\bar{D}$ takes the form

$$\bar{D}_{i,i} = \sum_{j=1}^{n} \bar{S}_{i,j} = \deg(a_i) + 1$$

and we can thus write $\bar{D}_{i,i} := \bar{D}(a_i)$. Now let $a_i, a_j \in A$ be two neighbouring atoms in $\mathcal{G}$, i.e. $a_j \in N(a_i)$. We choose

$$w_r(f_{r-1}(a_i), f_{r-1}(a_j), g(\{a_i, a_j\})) := \bar{D}(a_i)^{-1/2} \bar{D}(a_j)^{-1/2} f_{r-1}(a_j)$$

as our message-passing functions and

$$u_r(f_{r-1}(a_i), m_r(a_i)) := \mathrm{ReLU}(W_{r-1}^T (\bar{D}(a_i)^{-1} f_{r-1}(a_i) + m_r(a_i)))$$

as our atom-updating functions. To guarantee that the functions w_r and u_r are indeed well-defined, we assume without loss of generality that the node degree information $\bar{D}(a_i)$ is implicitly included in each initial atom feature vector $f_0(a_i)$ and then simply gets copied from $f_{r-1}(a_i)$ to $f_r(a_i)$ to assure its availability at each iterative feature update. The aggregated message for a_i is now given by

$$\begin{aligned} m_r(a_i) &= \sum \{\bar{D}(a_i)^{-1/2}\bar{D}(a_j)^{-1/2} f_{r-1}(a_j) \mid a_j \in N(a_i)\}_{\text{mul}} \\ &= \sum_{j=1}^{n} \bar{D}(a_i)^{-1/2} S_{i,j} \bar{D}(a_j)^{-1/2} f_{r-1}(a_j). \end{aligned}$$

As explained in 2.2 above, here the expression of the form $\sum\{...\}_{\text{mul}}$ simply represents the sum of all elements in the multiset $\{...\}_{\text{mul}}$, which means that the summation-operator is used as a permutation-invariant multiset-function.

If we denote the i-th row-vector of the matrix $\bar{D}^{-\frac{1}{2}}\bar{S}\bar{D}^{-\frac{1}{2}}$ with $[\bar{D}^{-\frac{1}{2}}\bar{S}\bar{D}^{-\frac{1}{2}}]_{[i,:]}$ then it follows that

$$\begin{aligned} f_r(a_i) &= \text{ReLU}\big(W_{r-1}^T(\bar{D}(a_i)^{-1} f_{r-1}(a_i) + m_r(a_i))\big) \\ &= \text{ReLU}\left(W_{r-1}^T \sum_{j=1}^{n} \bar{D}(a_i)^{-1/2} \bar{S}_{i,j} \bar{D}(a_j)^{-1/2} f_{r-1}(a_j)\right) \\ &= \text{ReLU}\left(W_{r-1}^T \big([\bar{D}^{-\frac{1}{2}}\bar{S}\bar{D}^{-\frac{1}{2}}]_{[i,:]} F_{r-1}\big)^T\right) \\ &= \text{ReLU}\left(W_{r-1}^T F_{r-1}^T [\bar{D}^{-\frac{1}{2}}\bar{S}\bar{D}^{-\frac{1}{2}}]_{[i,:]}^T\right) \end{aligned}$$

We can now finally conclude that

$$f_r(a_i)^T = \text{ReLU}\left([\bar{D}^{-\frac{1}{2}}\bar{S}\bar{D}^{-\frac{1}{2}}]_{[i,:]} F_{r-1} W_{r-1}\right)$$

and therefore

$$F_r = \text{ReLU}(\bar{D}^{-\frac{1}{2}}\bar{S}\bar{D}^{-\frac{1}{2}} F_{r-1} W_{r-1}).$$

□

We constructed our proof of Proposition 2.1 as a slightly adapted and more detailed version of the original proof we found in the article of Gilmer et al. [17]. Note that while GCNs are sensitive to the connectivity structure of their input graph, like many GNNs they do not take into account bond feature vectors. GCNs have been demonstrated to work reasonably well for a variety of graph-based prediction tasks [9, 18, 30, 84, 85, 86].

2.6.3 Graph Isomorphism Networks

In 2018, Xu et al. [29] published an influential article in which they introduced the graph isomorphism network (GIN) model. The GIN was developed to overcome some of the theoretical shortcomings of GCNs and other popular GNN-architectures available at the time. We will discuss the highly relevant motivation of Xu et al. and their theoretical insights in the next section, but will first give a brief description of their proposed model. The message-passing mechanism of GINs expressed via Equations 2.1 is defined in a straightforward manner via

$$w_r(f_{r-1}(a), f_{r-1}(\tilde{a}), g(\{a, \tilde{a}\})) = f_{r-1}(\tilde{a})$$

and

$$u_r(f_{r-1}(a), m_r(a)) = \phi_r((1 + \epsilon_r) f_{r-1}(a) + m_r(a)).$$

Here ϵ_r is a small (optionally trainable) parameter and ϕ_r is a trainable multilayer perceptron with at least one hidden layer. Note that ϕ_r is specifically required to be more powerful than the shallow neural network $x^T \mapsto \mathrm{ReLU}(x^T W)$ used in the definition of GCNs in Equation 2.4, which has no hidden layer and therefore does not fulfill the requirements of the universal approximation theorem for feedforward neural networks [87]. One can write the atom feature updating process for GINs in the compact recursive form

$$f_r(a) = \phi_r\Big((1 + \epsilon_r) f_{r-1}(a) + \sum f_{r-1}(N(a))_{\mathrm{mul}}\Big).$$

Like GCNs, standard GINs also ignore bond feature vectors, although modifications of the GIN-architecture that do consider bond features have been used successfully [21]. GINs are easy to implement, work well in practice and are currently reaching state-of-the-art performance in a variety of applications [9, 29, 84, 88].

2.6.4 Theoretical Expressivity of Graph Neural Networks

The relatively old GCNs are still popular GNN-models. However, they come with a significant caveat. This caveat was pointed out by Xu et al. [29] in their seminal 2018 paper and used as a motivation for the design of the GIN. They proved that GCNs (as well as a plethora of other commonly-used GNN-architectures) sometimes map non-isomorphic graphs to identical vectorial representations and thus suffer from a lack of theoretical expressivity. In other words, they discovered that GCNs and many other popular GNN-models cannot learn to distinguish certain simple graph structures, to the point where in some instances they severely underfit the training

set. The GIN-model was specifically developed to overcome this problematic lack of expressivity. And indeed, Xu et al. [29] managed to prove that GINs are strictly more expressive than GCNs and a number of other popular GNN-models. They specifically demonstrated that GINs can distinguish certain non-isomorphic graphs that GCNs cannot. In addition they showed that GINs are as expressive as standard message-passing GNNs can ever be; GINs are in the subclass of *maximally* expressive message-passing GNNs and are exactly as powerful at distinguishing non-isomorphic graphs as the canonical 1-Weisfeiler-Lehman (1-WL) graph isomorhism test [89].

Description 2.2 (1-Weisfeiler-Lehman Test)**.** Let $\mathcal{G} = (A, B)$ be a finite graph with a node featurisation function $f : A \to \mathbb{R}^k$. We now iteratively construct a sequence of functions $(c_r)_{r \in \mathbb{N}_0}$ on the set of graph nodes A. Each value $c_r(a)$ is imagined to be a momentary colouring of the node $a \in A$ at step r. The colouring functions are constructed via the following iterative scheme:

$$c_0(a) \coloneqq f(a),$$
$$c_r(a) = h(c_{r-1}(a), c_{r-1}(N(a))_{\text{mul}}).$$

Here $a \in A$ is a node of $\mathcal{G}$ and h is an injective hash function that maps each vertex colour along with its multiset of neighbouring colours $c_{r-1}(N(a))_{\text{mul}}$ to a new unique colour. This procedure terminates after a finite number of R steps when a stable colouring c_R is reached that cannot be changed by subsequent iterations. Let

$$c_R(A)_{\text{mul}} \coloneqq \{c_R(a) \mid a \in A\}_{\text{mul}}$$

be called the multiset of final vertex colours. If the multisets of final vertex colours of two input graphs $\mathcal{G}_1, \mathcal{G}_2$ are distinct, then the graphs are guaranteed not to be isomorphic. However, if the multisets of final vertex colours are identical, then the graphs are potentially but not necessarily isomorphic. If two non-isomorphic graphs reach identical multisets of final vertex colours, we say that the 1-WL test cannot distinguish these graphs.

A pair of non-isomorphic graphs that nevertheless cannot be distinguished by the 1-WL test is depicted in Figure 2.4. However, in spite of the existence of such counterexamples, the 1-WL test is still an elegant and classical method to tackle the graph isomorphism problem that can efficiently tell apart a large number of non-isomorphic graph structures. Moreover, the 1-WL test provides a natural benchmark against which the theoretical expressivity of GNNs can be compared.

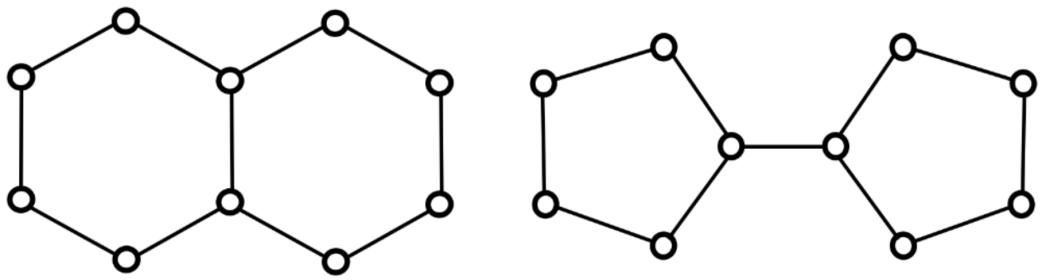

Figure 2.4: Example of two non-isomorphic graphs that cannot be distinguished by the 1-WL test if all nodes are assumed to have identical initial colourings. Image source: [90].

Theorem 2.1 (GNN-Conditions for 1-WL Power)**.** *A message-passing GNN can be shown to be maximally expressive and (equivalently) as powerful as the* 1*-WL test at distinguishing non-isomorphic graphs if the following injectivity-conditions hold:*

- ***Injective aggregation and update****. For each graph layer $r \in \{1, ..., R\}$ there must exist an injective multiset-function $\oplus_r$ and an injective function ψ_r such that the updating-procedure for atom features defined in Equations 2.1 can be written in the form*

  $$f_r(a) = \psi_r(f_{r-1}(a), \oplus_r f_{r-1}(N(a))_{mul}).$$

  This condition guarantees that distinct atom-neighbourhoods are always mapped to distinct atom feature vector updates.

- ***Injective pooling.*** *The multiset function $\tilde{\oplus}_R$ used in the pooling step 2.3 at the R-th (i.e. last) graph layer must be injective and the final pooling function q must be injective in the argument corresponding to the R-th graph layer. This condition guarantees that graphs with distinct final multisets of updated atom feature vectors are always mapped to distinct graph-level vectorial representations.*

Proof. We will show that the GNN can distinguish every pair of graphs that the 1-WL test can. Let $\mathcal{G} = (A, B)$ be a (finite) graph with a node featurisation function $f : A \to \mathbb{R}^k$. We will prove that for each graph layer r there exists an injective map α_r such that

$$f_r(a) = \alpha_r(c_r(a)) \quad \forall a \in A.$$

This means that the colour $c_r(a)$ of a node $a \in A$ at step r (as computed by the WL-algorithm) uniquely determines the node feature vector $f_r(a)$ at step r (as computed

by the GNN). Furthermore, the injectivity condition on α_r guarantees that nodes with distinct colours have distinct feature vectors.

We show the existence of α_r via induction over r. For $r = 0$, it holds for all $a \in A$ that $f_0(a) = \alpha_0(a)$ so we can simply choose the identity map for α_0. If we now assume for the induction step that α_{r-1} exists, then we can write

$$f_r(a) = \psi_r(f_{r-1}(a), \oplus_r f_{r-1}(N(a))_{\text{mul}}) = \psi_r(\alpha_{r-1}(c_{r-1}(a)), \oplus_r \alpha_{r-1}(c_{r-1}(N(a))_{\text{mul}})_{\text{mul}}).$$

Since ψ_r, $\oplus_r$ and α_{r-1} are injective and the composition of injective functions is again injective, it follows that there must be an injective function β_r such that

$$f_r(a) = \beta_r(c_{r-1}(a), c_{r-1}(N(a))_{\text{mul}}).$$

We now set $\alpha_r := \beta_r \circ h^{-1}$ whereby h^{-1} is the inverse of the hash function used in the 1-WL-algorithm. Once again, α_r is injective since β_r and h^{-1} are injective. Moreover, it now holds for all $a \in A$ that

$$\alpha_r(c_r(a)) = \beta_r(h^{-1}(c_r(a))) = \beta_r(c_{r-1}(a), c_{r-1}(N(a))_{\text{mul}}) = f_r(a)$$

which concludes the proof that α_r exists and has the desired properties.

Now let $\mathcal{G} = (A, B)$ and $\tilde{\mathcal{G}} = (\tilde{A}, \tilde{B})$ be two graphs that can be distinguished with the 1-WL test. We want to show that $\mathcal{G}$ and $\tilde{\mathcal{G}}$ are then mapped to different graph-level feature vectors by the message-passing GNN. The fact that $\mathcal{G}$ and $\tilde{\mathcal{G}}$ can be distinguished by the 1-WL test means that at some graph layer R their respective multisets of node colours must be different:

$$c_R(A)_{\text{mul}} \neq \tilde{c}_R(\tilde{A})_{\text{mul}}.$$

Due to the injectivity of α_R this means that the multisets of updated atom feature vectors at the R-th layer must also be different:

$$f_R(A)_{\text{mul}} = \alpha_R(c_R(A)_{\text{mul}})_{\text{mul}} \neq \alpha_R(\tilde{c}_R(A)_{\text{mul}})_{\text{mul}} = f_R(\tilde{A}_{\text{mul}}).$$

Since the multiset-function $\tilde{\oplus}_R$ that is used in the pooling step is assumed to be injective and since the final pooling function q is assumed to be injective in its R-th argument as well, we can finally conclude that

$$\mathbb{R}^l \ni q(\tilde{\oplus}_0 f_0(A)_{\text{mul}}, ..., \tilde{\oplus}_R f_R(A)_{\text{mul}}) \neq q(\tilde{\oplus}_0 f_0(\tilde{A})_{\text{mul}}, ..., \tilde{\oplus}_R f_R(\tilde{A})_{\text{mul}}) \in \mathbb{R}^l$$

which proves that the final vectorial GNN-embeddings for $\mathcal{G}$ and $\tilde{\mathcal{G}}$ must be different.

We have shown that all graphs that can be distinguished by the 1-WL test can also be distinguished by a message-passing GNN under suitable injectivity conditions which means that such GNNs are at least as powerful as the 1-WL test. For the full proof, including the converse that all graphs that can be distinguished by a message-passing GNN can also be distinguished by the 1-WL test, we refer the reader to the original article of Xu et al. [29]. □

For simplicity we have omitted bond feature vectors in Description 2.2 and Theorem 2.1, but it is easy to formulate and prove equivalent versions of both statements that take these into account. Unlike GCNs, GINs fulfill the injectivity conditions from Theorem 2.1 and are thus in the class of maximally expressive message-passing GNNs that are as powerful as the 1-WL test. Xu et al. [29] suggested that the technical reasons why GCNs fail to meet the injectivity criteria of Theorem 2.1 is partly connected with the fact that their atom-updating scheme described in 2.4 only involves a shallow neural network of the form $x^T \mapsto \mathrm{ReLU}(x^T W)$ that lacks hidden layers. We hypothesise that the increased expressivitiy of GINs compared to theoretically weaker models such as GCNs most strongly translates into superior performance when training data is abundant. This idea is greatly supported by recent experiments in the realm of self-supervised learning where GINs consistently beat GCNs by a large margin when fine-tuned on supervised tasks after pre-training on millions of unlabelled molecular graphs [9, 21].

Arguably the most important contribution of Xu et al. [29] was not the invention of the GIN, but the formulation of a mathematical framework to analyse the expressivity of GNNs that was based on well-known concepts from classical graph theory such as the graph isomorphism problem and the 1-WL test. Notably, this breakthrough was achieved in parallel with Morris et al. [91] who published similar insights and proposed k-GNNs as a generalisation of traditional message-passing GNNs. The k-GNN architecture was designed to break through the expressivity barrier posed by the 1-WL test and move up the *WL-hierarchy*.

Description 2.3 (WL-Hierarchy)**.** The WL-hierarchy is constituted by the k-WL tests for $k \in \mathbb{N}$. The k-WL tests represent a family of polynomial-time graph isomorphism tests of strictly increasing expressivity (with the exception of the 2-WL test that is as powerful as the 1-WL test). Each k-WL test represents an extension of the 1-WL test from Description 2.2 that operates on k-tuples of nodes.

One can prove that the k-GNN model is as expressive as the k-WL test; however, the k-GNN-architecture does not fall under the umbrella of local message-passing

GNNs since it involves higher-order operations on sets of graph nodes. This makes k-GNNs prohibitively computationally expensive and normally impossible to use in practice.

A potential way forward was recently proposed by Bouritsas et al. [90] in the form of graph substructure networks (GSNs). GSNs are associated with the emerging field of geometric deep learning [82, 83] whose aim it is to effectively generalise deep learning architectures to non-Euclidean domains such as graph structures. GSNs were designed with the goal of being simultaneously practically applicable *and* more expressive than the 1-WL test. The key idea of GSNs is to compute messages in a manner that depends on the structural relationships between the considered nodes. This can be achieved by adding precomputed atomic structural descriptors for both nodes to the message-passing function $(w_r)_{r=1}^{R}$ in the procedure outlined in Equations 2.1. Each node descriptor counts how often the node appears in a particular topological role in all subgraphs of the input graph that are isomorphic to a particular predefined (small) graph. The node descriptors do not get updated and remain static across all graph layers. The *a priori* choice of basic subgraphs to look out for in the input graphs allows for the establishment of a useful inductive bias tailored to the prediction task at hand: for example, triangles might be relevant in social networks while ring-structures might play an important role in molecular graphs. Since GSNs only represent a slight generalisation of the standard local message-passing framework, their application is computationally feasible; and since they still allow for the inclusion of higher-order structural information in each message-passing step, they can be made strictly more powerful than the 1-WL test (which then for example enables them to distinguish the graphs from Figure 2.4). The expressivity of GSNs cannot be easily formulated though within the larger WL-hierarchy since suitably designed GSNs can distinguish (for example) *some* graph pairs in the 3-WL or 4-WL class but not all. GSNs have already shown encouraging performance on chemical prediction tasks [90].

2.6.5 Critical View

Below we give a list of advantages **(+)** and disadvantages **(−)** of message-passing GNNs as a molecular featurisation method.

(+) **Differentiability.** Message-passing GNNs are able to learn features directly from highly explicit graph-representations of molecules in a differentiable and task-specific manner. In this particular sense we refer to GNNs as trainable.

From a mathematical perspective, the training process for GNNs takes the form of a gradient-based continuous optimisation problem. The differentiability of GNNs could potentially allow them to extract substantially larger amounts of valuable chemical information from molecular graphs than non-differentiable featurisation methods such as ECFPs and PDVs.

(+) **Low dimensionality.** Many message-passing GNNs still produce useful features for downstream prediction tasks even if the fingerprint length is limited to $l \leq 100$. This can be seen in contrast to ECFPs where the minimum fingerprint length to reach an acceptable level of performance is usually considered to be $l \geq 1024$. The relatively low dimensionality of GNN-based feature vectors can decrease the risk of costly downstream-computations and overfitting.

(+) **Ability to capture intricate patterns due to model complexity.** GNN-architectures are often complex and involve large numbers of trainable parameters. This may enable them to detect nuanced and intricate patterns, especially when the training set is large.

(+) **High expressivity of some models.** The recent development of the GSN [90] as discussed in Section 2.6.4 has shown that GNN-models can be made both practical and substantially more powerful than the 1-WL test (and thus also more powerful than ECFPs) at distinguishing graph structures.

(−) **Low expressivity of some models.** As discussed in Section 2.6.4, many common GNN-models such as GCNs are substantially less powerful than the 1-WL test at distinguishing graph structures and in some cases severely underfit the training set.

(−) **Risk of overfitting due to model complexity.** The complexity of GNN-architectures may lead them to require relatively large amounts of data and/or careful regularisation in order to avoid overfitting and generalise effectively.

(−) **Limited depth.** The success of modern deep-learning architectures lies to a large part in their ability to automatically extract abstract high-level features directly from raw input data. This ability is crucially reliant on the sufficient *depth* of the neural model since learned features tend to become more abstract and task-specific as they are moving through consecutive network layers. GNNs cannot yet leverage the power of deep architectures as the predictive utility of learned node embeddings tends to decrease sharply after a few GNN-layers.

The exact reasons and possible solutions for this pathology are still under research [30, 92, 93, 94, 95]. A likely cause might lie in the phenomenon of *oversmoothing* which refers to a tendency of successively updated node features to eventually become indistinguishable.

(−) **Difficulty of interpreting learned features.** The graph-level featurisation generated by the global pooling step of a GNN usually consists of a vector of obscure real numbers that cannot be directly interpreted by human experts in any straightforward manner.

(−) **No chemically reasonable initial embedding of chemical space.** The molecular featurisations extracted by untrained GNNs essentially represent extremely noisy random projections of the information contained in the initial molecular graph features. GNNs therefore need to relearn a reasonable embedding from chemical space from scratch every time they are trained on a new problem (unless they have been combined with a suitable self-supervised or transfer learning approach). In particular, this might force GNNs to continuously relearn basic chemical features that are useful across a wide set of tasks. This can be seen in contrast to classical methods like ECFPs and PDVs which automatically generate embeddings of chemical space that have a basic utility for many applications.

(−) **Difficulty of implementation.** PDVs and ECFPs can be generated easily via ready-to-use algorithms implemented in widely-used cheminformatics libraries such as the `Python`-package `RDKit` [70]. The implementation of a tailored message-passing GNN-model, on the other hand, usually requires familiarity with technically advanced graph-based deep-learning libraries such as `PyTorch Geometric` [96].

(−) **High computational cost.** Like most deep-learning models, GNNs involve large numbers of trainable parameters and costly computations related to the multiplication of large matrices. This can make GNN-models much slower to train than PDV or ECFP-based models, although this difference in speed can be significantly mitigated if a suitable GPU can be leveraged during GNN-training.

(−) **Locality of receptive field.** Just like ECFPs, message-passing GNNs are based on a local neighbourhood-aggregation scheme for the nodes in a graph. By design, they are thus only capable of learning node embeddings that contain

features from local circular subgraphs; this prevents information-flow between distant nodes during the atom feature updating process.

(−) Potential information-loss during graph pooling. The global pooling step in a GNN that combines the multisets of updated node embeddings to a final graph-level feature vector can easily turn into a (non-injective) information-bottleneck if not designed carefully. Imagine for example two multisets of identical node feature vectors $\{f, f\}_{\text{mul}}$ and $\{f, f, f\}_{\text{mul}}$; if the popular averaging-operator is used to pool these two multisets, they lead to the same output representation f even though both multisets are different and should thus be mapped to distinct representations. A potential way to avoid such pitfalls may be to employ differentiable graph pooling techniques that (at least in theory) can provably approximate any sufficiently regular pooling function [31].

2.7 Molecular Featurisations: Critical Overview

We have summarised the critical analyses of PDVs (see Section 2.4.2), ECFPs (see Section 2.5.4) and message-passing GNNs (see Section 2.6.5) in Table 2.4 to provide a final overview before moving on to the systematic computational experiments in Chapter 3.

Molecular Featurisation Methods: Critical Overview		
Method	**Advantages (+)**	**Disadvantages (−)**
PDV	• Low computational cost • Interpretability (in some cases) • Simplicity of implementation • Chemically reasonable initial embedding of chemical space • Low dimensionality • Global receptive field	• Non-differentiability • Necessity for feature selection • Finite number of descriptors
ECFP	• Low computational cost • Interpretability • Simplicity of implementation • Chemically reasonable initial embedding of chemical space • Arbitrary circular subgraphs	• Non-differentiablity • Trade-off between dimensionality and hash collisions • Locality of receptive field
GNN	• Differentiability • Low dimensionality • Ability to capture intricate patterns due to model complexity • High expressivity of some models	• Low expressivity of some models • Risk of overfitting due to model complexity • Limited depth • Difficulty of interpreting learned features • No chemically reasonable initial embedding of chemical space • Difficulty of implementation • High computational cost • Locality of receptive field • Potential information-loss during graph pooling

Table 2.4: Advantages and disadvantages of physicochemical-descriptor vectors (PDVs), extended-connectivity fingerprints (ECFPs) and message-passing graph neural networks (GNNs) for molecular featurisation.

Chapter 3

Exploring Molecular Featurisations for QSAR and Activity-Cliff Prediction: A Computational Study

We have published our findings from this chapter as a peer-reviewed research paper [45] in the Journal of Cheminformatics. Most of the figures and tables, as well as major parts of the text contained in this chapter are thus either identical or similar to the content of our published article. We have also presented results from this chapter as a scientific poster [46] at the 10th International Congress on Industrial and Applied Mathematics (ICIAM 2023, Tokyo).

3.1 Overview

In Chapter 2 we have given a detailed technical description of state-of-the-art featurisation methods in molecular machine learning along with a critical discussion of their theoretical strengths and weaknesses. However, perhaps the most pressing question has not yet been addressed: which featurisation method actually leads to the strongest performance at chemical prediction tasks? There is still disagreement in the computational-chemistry community whether modern trainable message-passing GNNs do in fact outperform classical non-trainable featurisation methods such as ECFPs and PDVs at important molecular machine-learning tasks. A multitude of studies have found that GNNs do indeed clearly outcompete ECFPs and PDVs [17, 20, 22, 32, 33, 34, 35, 36]. However, a considerable number of other studies have found evidence pointing towards the exact opposite [6, 7, 8, 9, 10, 11, 12, 13, 37].

In this chapter, we present a series of carefully designed computational experiments to systematically investigate the predictive powers of PDVs, ECFPs and GINs for two important tasks in computational drug discovery: the well-researched prob-

lem of quantitative structure-activity relationship (QSAR) prediction and the largely unexplored and difficult challenge of activity-cliff (AC) prediction. QSAR-prediction refers to the problem of using experimental data to learn a mapping from a computational representation $\mathcal{R}$ of a chemical compound to its biological activity value against a given pharmacological target such as an enzyme or a receptor. A QSAR model usually takes the form of a machine-learning model that can be decomposed into a molecular featurisation method followed by a regression technique:

$$\mathcal{R} \quad \overset{\longmapsto}{\text{featurisation}} \quad \mathcal{F} = (f_1, ..., f_l) \in \mathbb{R}^l \quad \overset{\longmapsto}{\text{regression}} \quad \text{activity } (\text{pK}_\text{i}, \text{pIC}_{50}, ...) \in \mathbb{R}.$$

ACs are pairs of very similar compounds whose molecular structure only differs by a small change at a specific site but which exhibit a very large difference in their activity against a given pharmacological target. ACs explicitly encode small structural changes that abruptly change a biological effect and are thus rich in pharmacological information. AC-prediction primarily refers to the task of classifying whether a given pair of structurally similar compounds forms an AC or not, but usually also implicitly encompasses the classification of the potency direction (PD) of the pair (i.e. which of both compounds is more active). Every QSAR model can be repurposed as an AC-prediction model by using it to individually predict the activities of two structurally similar compounds (which gives the PD-classification) and then thresholding the absolute difference of the two predicted activities (which gives the AC-classification). Accurate QSAR-prediction and AC-prediction models would represent valuable tools in the computer-aided search for novel pharmacological compounds with desired properties.

In this chapter, we conduct a rigorous computational study to evaluate the QSAR and AC-prediction performance of nine modern QSAR models on three curated pharmacological data sets (dopamine receptor D2, factor Xa and SARS-CoV-2 main protease). Each QSAR model is generated by merging one of three molecular featurisation methods (PDVs, ECFPs, or GINs) with one of three canonically used regression techniques (random forests (RFs), k-nearest neighbours (kNNs), or multilayer perceptrons (MLPs)). Our experimental setup thus allows for a systematic comparison of the three studied featurisations across three regression techniques, three pharmacological targets, and three distinct chemical prediction tasks (QSAR-prediction, AC-classification, and PD-classification). Our experiments are thus organised according to a robust combinatorial methodology of the form

$$|\{\text{featurisers}\}| \times |\{\text{regressors}\}| \times |\{\text{data sets}\}| \times |\{\text{tasks}\}| = 3 \times 3 \times 3 \times 3$$

that to the best of our knowledge has not been described before in the literature. The QSAR-prediction, AC-classification and PD-classification performance of each featuriser-regressor combination on each data set is measured using a strict data splitting and evaluation strategy involving a full algorithmic hyperparameter optimisation. To evaluate AC and PD-classification performance, we develop a novel pair-based data-splitting method that operates on top of data splits for individual molecules in a natural and interpretable manner. Our proposed data split for sets of molecular pairs is conceptually simple, yet allows one to make important distinctions between several types of compound pairs with respect to their molecular overlap with an underlying training set of individual compounds.

It has been hypothesised that ACs form one of the major sources of prediction error in QSAR modelling [39, 41] but so far only a few studies have attempted to generate empirical evidence for this claim [37, 40, 97]. However, these studies follow an indirect approach by measuring QSAR performance on *individual* compounds involved in ACs instead of *pairs* of similar compounds. Our published work [45] closes a gap in the current QSAR and AC literature by providing the first computational study to investigate the capabilities of state-of-the-art QSAR models to classify whether a given pair of similar compounds forms an AC or not. The main aim of the work in this chapter is to answer the following question:

- Which molecular featurisation method performs best for QSAR or AC-prediction across different regression techniques and data sets? In particular, when (if at all) do trainable GINs outperform non-trainable PDVs and ECFPs?

Besides this, we are also interested in the following questions:

- When (if at all) are common QSAR models capable of predicting the existence of ACs?
- When (if at all) are common QSAR models capable of predicting which of two similar compounds is more active?
- Which QSAR model shows the strongest AC-prediction performance, and should thus be used as a baseline against which to compare tailored AC-prediction models?
- What is the quantitative relationship between QSAR and AC-prediction performance for QSAR models?

3.2 Introduction to Activity Cliffs and Activity-Cliff Prediction

As mentioned above, activity cliffs (ACs) are pairs of small molecules that exhibit high structural similarity but at the same time show an unexpectedly large difference in their binding affinity against a given pharmacological target [38, 39, 40, 41, 42, 43, 44]. The existence of ACs directly defies the intuitive idea that chemical compounds with similar structures should have similar activities, often referred to as the *molecular similarity principle*. An example of an AC between two inhibitors of blood coagulation factor Xa [98] is depicted in Figure 3.1; a small chemical modification involving the addition of a hydroxyl group leads to an increase in binding affinity of almost three orders of magnitude.

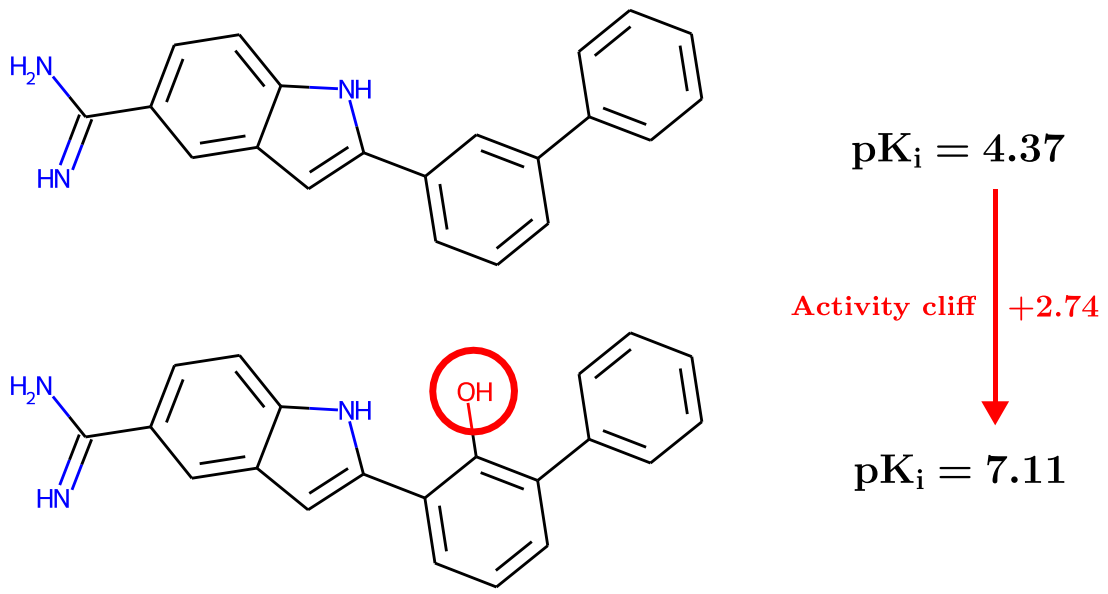

Figure 3.1: Example of an activity cliff (AC) for blood coagulation factor Xa. A small structural change in the upper compound leads to an increase in binding affinity of almost three orders of magnitude. Here binding affinity is quantified via the commonly-used pK_i-value, which represents the negative decadic logarithm of the dissociation constant K_i of the drug-target complex. Both compounds can be found in the same ChEMBL assay with ID 658338.

For medicinal chemists, ACs can be puzzling and confound their understanding of structure-activity relationships (SARs) [42, 99, 100]. ACs reveal small compound-modifications with large biological impact and thus represent rich sources of pharmacological information. Mechanisms by which a small structural change can give rise to an AC include a drastic change in 3D-conformation and/or the switching to a

different binding mode or even binding site. Another mechanism that could potentially induce ACs is the stabilisation of a single molecular conformation as a result of a small structural change; such an effect could possibly increase the equilibrium concentration of the binding conformation, thus leading to an AC without creating any specific change in binding interaction. One can speculate that this might in fact be the mechanism underlying the AC shown in Figure 3.1; it is imaginable that the depicted addition of a hydroxyl group leads to a stabilising intramolecular N-H to O interaction.

ACs form discontinuities in the SAR-landscape and can therefore have a crucial impact on the success of lead-optimisation programmes. While knowledge of ACs can be powerful when trying to escape from flat regions of the SAR-landscape, their presence can be detrimental in later stages of the drug development process, when multiple molecular properties beyond mere activity need to be balanced carefully to arrive at a safe and effective compound [41, 42]. In the field of cheminformatics, ACs are suspected to form one of the major roadblocks for successful QSAR modelling [39, 40, 41, 97]. In practice, abrupt changes in activity are expected to negatively influence the abilities of QSAR methods to learn general SAR-trends. On the other hand, in theory ACs can be seen as an opportunity for such methods to extract precious SAR-knowledge. During the development of QSAR models, ACs are sometimes dismissed as measurement errors [101], but simply removing ACs from a training data set can result in a loss of large amounts of pharmacological information [102].

Golbraikh et al. [97] developed the simple MODI metric which quantifies the smoothness of the SAR-landscape of a given binary molecular classification data set. They subsequently showed that SAR-landscape smoothness is a strong determinant for downstream QSAR-modelling performance across a large number of data sets. In a conceptually related work, Sheridan et al. [40] found that the density of ACs in a given molecular data set is strongly predictive for its overall modelability by classical descriptor and fingerprint-based QSAR methods. Furthermore, they demonstrated that such methods incur a significant drop in performance when the molecular test set is restricted to only include "cliffy" compounds which form a large number of ACs. In a recent and more extensive study, van Tilborg et al. [37] observed a similar drop in performance when testing a large number of classical and graph-based QSAR techniques on sets of compounds involved in ACs. Notably, in both studies this performance drop was also observed for highly nonlinear and adaptive deep learning models. Moreover, van Tilborg reports that descriptor-based QSAR methods do in

fact outperform more complex deep learning models on "cliffy" compounds associated with ACs. This runs counter to earlier hopes expressed in the literature that the approximation powers of highly parametric deep networks might ameliorate the problem of ACs [103].

While these works provide valuable insights into the detrimental effects of SAR discontinuity on QSAR models, they consider ACs mainly indirectly by focussing on *individual* compounds involved in ACs. Arguably, a distinct and more natural approach would be to investigate ACs directly at the level of compound *pairs*. This approach has been followed in the AC-prediction field which is concerned with the development of techniques to classify the existence and direction of potential ACs. An effective AC-prediction method would be of great value for drug development with important applications in rational compound optimisation and automatic SAR-knowledge acquisition.

The AC-prediction literature is still very thin compared to the QSAR-prediction literature. An attempt to conduct an exhaustive literature review on AC-prediction techniques revealed a total of 15 methods [47, 51, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116], all of which have been published since 2012. Current AC-prediction methods are often based on creative ways to extract features from pairs of molecular compounds in a manner suitable for standard machine learning pipelines. For example, Horvath et al. [111] used condensed graphs of reactions [117, 118], a representation technique originally introduced for modelling of chemical reactions, to encode pairs of similar compounds and subsequently predict ACs. Another method was recently described by Iqbal et al. [51] who investigated the abilities of convolutional neural networks operating on 2D images of compound pairs to distinguish between ACs and non-ACs. Interestingly, none of the AC-prediction methods we identified employ feature extraction techniques built on GNNs with the exception of Park et al. [115] who recently applied graph convolutional methods to compound-pairs to predict ACs.

In spite of the existence of various technically complex AC-prediction models there are significant gaps left in the current AC-prediction literature. Note that any given QSAR model can immediately be repurposed as an AC-prediction model by using it to individually predict the activities of two structurally similar compounds and then thresholding the difference of both predicted activities. Nevertheless, at the moment there is no study that uses this straightforward technique to rigorously investigate the potential of modern QSAR models to classify whether a given pair of compounds forms an AC or not. Importantly, this also entails that the most salient AC-prediction

models [51, 104, 105, 106, 111] have not been compared to a simple QSAR-modelling baseline. It is thus an open question to what extent (if at all) these tailored AC-prediction techniques outcompete repurposed state-of-the-art QSAR methods at the detection of ACs. This question is especially relevant in light of the fact that several published AC-prediction models [51, 104, 106] are evaluated via compound-pair-based data splits which incur a significant overlap between the training set and test set at the level of individual molecules. This type of data split should strongly favour standard QSAR models for AC-prediction, yet a comparison to such baseline methods is lacking. We address these problems by providing the first computational study that explores the capabilities of modern QSAR models to predict ACs. The results of our study establish a natural baseline against which more advanced AC-prediction models can be compared. Moreover, we disentangle the prevalent data-splitting issues in AC-prediction settings by introducing a novel splitting technique. This method, which we recommend as the standard for future publications, allows one to make important distinctions between several types of compound-pair test-sets with respect to their molecular overlaps with the underlying training set.

3.3 Experimental Methodology

3.3.1 Molecular Data Sets

We built three binding affinity data sets of small-molecule inhibitors of dopamine receptor D2, factor Xa, and SARS-CoV-2 main protease. Dopamine receptor D2 is the main site of action for classic antipsychotic drugs which act as antagonists of the D2 receptor [119]. Factor Xa is an enzyme in the coagulation cascade and a canonical target for blood-thinning drugs [98]. SARS-CoV-2 main protease is one of the key enzymes in the viral replication cycle of the SARS coronavirus 2, that recently caused the unprecedented COVID-19 pandemic; it is one of the most promising targets for antiviral drugs against this coronavirus [120]. The protein structures of dopamine receptor D2, factor Xa and SARS-CoV-2 main protease are visualised in Figures 3.2 to 3.4, respectively.

For dopamine receptor D2 and factor Xa, data was extracted from the ChEMBL database [122] in the form of SMILES strings with associated K_i [nM] values. All activity values extracted from ChEMBL were associated with the equality standard relation "="; qualified activity values associated with other relations such as "<" or ">" were not used. In the case of SARS-CoV-2 main protease, data was obtained from the COVID moonshot project [123] in the form of SMILES strings with associated

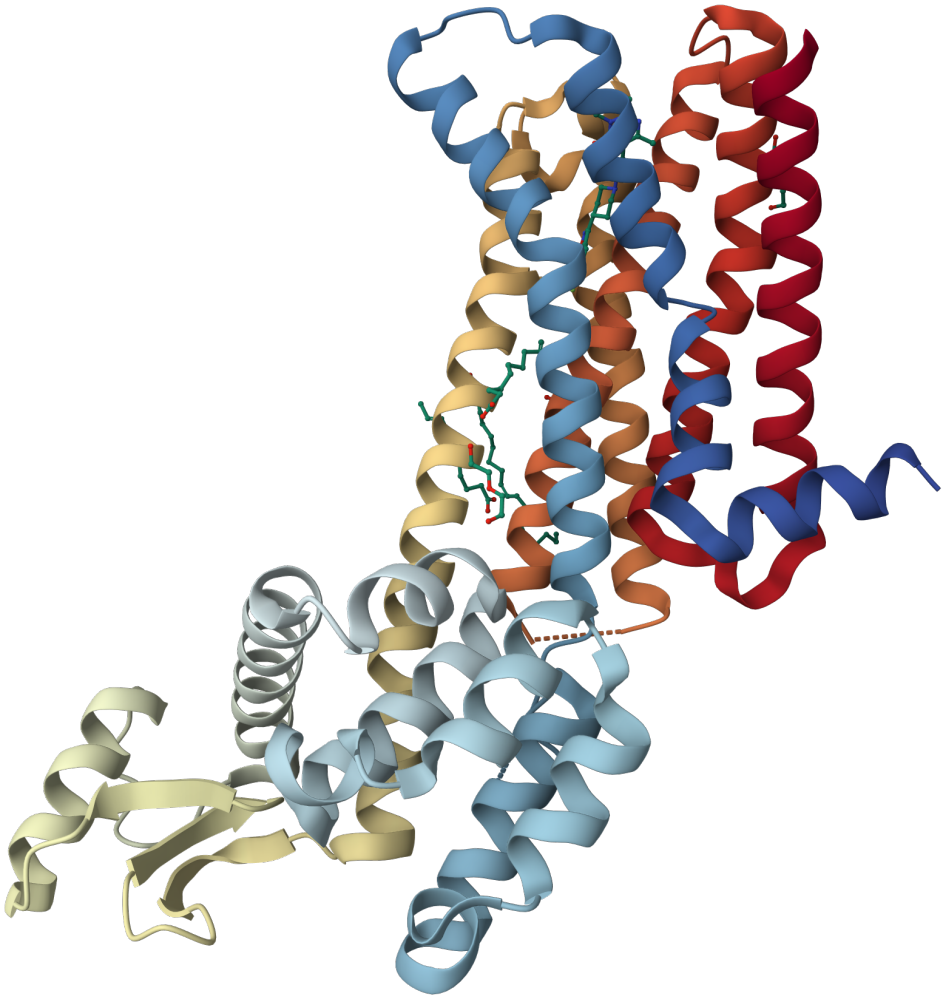

Figure 3.2: Protein structure of **dopamine receptor D2**. Extracted from the Research Collaboratory for Structural Bioinformatics Protein Data Bank (RCSB PDB) [121]. PDB ID: 6CM4.

Figure 3.3: Protein structure of **factor Xa**. Extracted from the Research Collaboratory for Structural Bioinformatics Protein Data Bank (RCSB PDB) [121]. PDB ID: 2JKH.

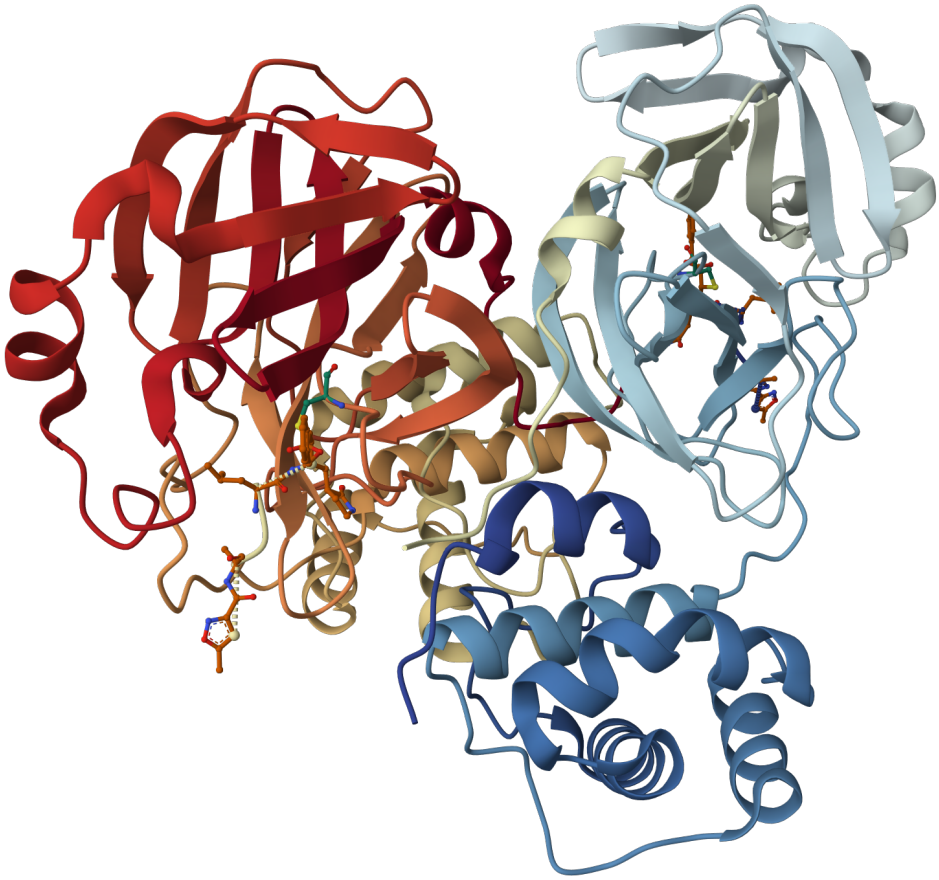

Figure 3.4: Protein structure of **SARS-CoV-2 main protease**. Extracted from the Research Collaboratory for Structural Bioinformatics Protein Data Bank (RCSB PDB) [121]. PDB ID: 6LU7.

IC_{50} [µM] values. SMILES strings were standardised and desalted via the ChEMBL structure pipeline [124]. This step also removed solvents and isotopic information. Following this, SMILES strings that produced error messages when turned into an `RDKit` mol object were deleted. Finally, a scan for duplicate molecules was performed: If the activities in a set of duplicate molecules were within the same order of magnitude then the set was unified via geometric averaging. Otherwise, the measurements were considered unreliable and the corresponding set of duplicate molecules was removed. This procedure reduced the data set for dopamine receptor D2 / factor Xa / SARS-CoV-2 main protease from 8883 / 4116 / 1926 compounds to 6333 / 3605 / 1924 unique compounds whereby 174 / 21 / 0 sets of duplicate SMILES were removed and the rest was unified.

3.3.2 Definition of Binary Activity-Cliff Classification-Tasks

The exact definition of an AC hinges on two concepts: structural similarity and large activity difference. An elegant technique to measure structural similarity in the context of AC analysis is given by the matched molecular pair (MMP) formalism [125, 126]. An MMP is a pair of compounds that share a common structural core but differ by a small structural change at a specific site. Figure 3.1 depicts an example of an MMP whose variable parts are formed by a hydrogen atom and a hydroxyl group. To detect MMPs algorithmically, we used the `mmpdb` `Python`-package provided by Dalke et al. [127]. We restricted ourselves to the commonly-used definition of MMPs [104, 105, 111] which employs relatively generous size constraints: the MMP core was required to contain at least twice as many heavy atoms as either of the two variable parts; each variable part was required to contain no more than 13 heavy atoms; the maximal size difference between both variable parts was set to eight heavy atoms; and bond cutting was restricted to single exocyclic bonds. Note that the decomposition of an MMP into a core part and two variable parts is not necessarily unique as the size of the chosen core part can vary. To guarantee a well-defined mapping from each MMP to a unique structural core, we canonically chose the core that contained the largest number of heavy atoms whenever there was ambiguity.

Based on the ratio of the activity values of both MMP compounds, each MMP was assigned to one of three classes: “AC”, “non-AC” or “half-AC”. In accordance with the literature [99, 104, 109, 111, 128] we assigned an MMP to the “AC”-class if both activity values differed by at least a factor of 100. If both activity values differed by no more than a factor of 10, then the MMP was assigned to the “non-AC”-class. In the residual case the MMP was assigned to the “half-AC”-class. To

arrive at a well-separated binary classification task, we labelled all ACs as positives and all non-ACs as negatives. The half-ACs were removed and not considered further in our experiments. It is also relevant to know the direction of a potential activity cliff, i.e. which of the compounds in the pair is more active. We thus assigned a binary label to each MMP indicating its potency direction (PD). PD-classification is a balanced binary classification task. Table 3.1 gives an overview of all our curated data sets.

Data Set	**Dopamine Receptor D2**	**Factor Xa**	**SARS-CoV-2 Main Protease**
Compounds	6333	3605	1924
MMPs	35484	21292	12594
ACs	461	1896	521
Half-ACs	3804	4693	1762
Non-ACs	31219	14703	10311
ACs : Non-ACs	$\approx 1:68$	$\approx 1:8$	$\approx 1:20$
Measurement	K_i [nM]	K_i [nM]	IC_{50} [µM]

Table 3.1: Sizes of our curated data sets and their respective numbers of matched molecular pairs (MMPs), activity cliffs (ACs), half-activity-cliffs (half-ACs) and non-activity-cliffs (non-ACs).

It is worth emphasizing again that the definition of an AC hinges on the employed measure of structural similarity. While the MMP-formalism currently represents the most widespread similarity criterion in the field of AC research, other techniques have regularly been employed in the past. In particular, the Tanimoto similarity of binary structural fingerprints (such as ECFPs or MACCS fingerprints) has originally frequently been used to define ACs [43, 129]. One obvious advantage of this approach is its computational simplicity. Another advantage may be that ACs can be defined directly within the input feature space of a potential machine learning model. As a result, ACs based on this definition might intuitively map very well to compound pairs whose activity difference is indeed challenging to predict for fingerprint-based QSAR models. However, using Tanimoto similarity in feature space to define ACs also has a variety of serious drawbacks that have been effectively summarised by Stumpfe et al. [43]: Tanimoto similarity varies continuously in the interval $[0, 1]$ and thus requires the choice of a subjective threshold value for the classification of compound pairs as similar. Furthermore, different fingerprints generally lead to

different Tanimoto similarities; this makes the classification of compound pairs as similar dependent on the employed fingerprint. Finally, Tanimoto similarity values can sometimes be difficult to interpret from a chemical perspective and might not always accurately reflect the intuitions of a chemical expert. In contrast, note that MMPs do not require the choice of a similarity threshold, are not dependent on a particular fingerprint featurisation, and have a clear chemical interpretation in terms of structural cores and variable parts. These reasons might explain the shift away from Tanimoto similarity towards the MMP-formalism that could be observed in recent years in the field of AC research.

3.3.3 Developed Pair-Based Data Splitting Technique

ACs consist of molecular pairs rather than single molecules; it is thus not obvious how best to split up a chemical data set into non-overlapping training and test sets for the fair evaluation of an AC-prediction method. There seems to be no consensus about which data splitting strategy should be canonically used. Several authors [51, 104, 106] have employed a random split at the level of compound pairs. While this technique is conceptually straightforward, it must be expected to incur a significant overlap between training and test set at the level of individual molecules. For example, randomly splitting up a set of three MMPs $\{(\mathcal{R}_1, \mathcal{R}_2), (\mathcal{R}_2, \mathcal{R}_3), (\mathcal{R}_1, \mathcal{R}_3)\}$ into a training and a test set may lead to $(\mathcal{R}_1, \mathcal{R}_2)$ and $(\mathcal{R}_1, \mathcal{R}_3)$ getting assigned to the training set and $(\mathcal{R}_2, \mathcal{R}_3)$ getting assigned to the test set. This corresponds to a full inclusion of the test set in the training set at the level of individual molecules. A molecular overlap of this kind is problematic for at least three reasons: Firstly, it likely leads to overly optimistic performance estimates of AC-prediction methods since they will have already encountered some of the test compounds during training. Secondly, it does not model the natural situation encountered by medicinal chemists who it is assumed do not know the activity value of at least one compound in a test-set MMP. Thirdly, the mentioned molecular overlap should lead to strong AC-prediction results for standard QSAR models, but to the best of our knowledge, no such control experiments have been conducted in the literature.

Horvath et al. [111] and Tamura et al. [105] have made efforts to address the shortcomings of a compound-pair-based random split. They proposed advanced data splitting algorithms designed to mitigate the described molecular-overlap problem by either managing distinct types of test sets according to compound membership in the training set or by designing splitting techniques based on the structural cores of MMPs. However, their data splitting schemes exhibit a relatively high degree of

technical complexity which can make their implementation and interpretation non-straightforward.

For our study, we propose a novel data splitting method which may represent a favourable trade-off between rigour, interpretability and simplicity. Our technique shares some of its concepts with the methods proposed by Horvath et al. [111] and Tamura et al. [105] but might be simpler to implement and interpret. We first split the data into a training and test set at the level of individual molecules and then use this basic split to distinguish several types of test sets at the level of compound pairs. Let

$$\mathfrak{D} = \{\mathcal{R}_1, \mathcal{R}_2, ...\}$$

be the given data set of individual compounds. One can for instance think of $\mathfrak{D}$ as a set of SMILES strings or molecular graphs. Furthermore, let

$$\mathfrak{M} \subseteq \{(\mathcal{R}, \tilde{\mathcal{R}}) \mid \mathcal{R}, \tilde{\mathcal{R}} \in \mathfrak{D}\}$$

be the set of MMPs in $\mathfrak{D}$ that are eligible for the AC-classification task. This means that $\mathfrak{M}$ represents the set of MMPs in $\mathfrak{D}$ that have either been labelled as ACs or as non-ACs. Then each MMP $(\mathcal{R}, \tilde{\mathcal{R}}) \in \mathfrak{M}$ consists of two structurally similar compounds $\mathcal{R}$ and $\tilde{\mathcal{R}}$ that share a common structural core which we denote as $\text{core}(\mathcal{R}, \tilde{\mathcal{R}})$. To avoid redundancy, we associate each MMP with an (arbitrary) ordering of its two involved compounds and only contain one of both orderings in $\mathfrak{M}$, i.e. if $(\mathcal{R}, \tilde{\mathcal{R}}) \in \mathfrak{M}$ then $(\tilde{\mathcal{R}}, \mathcal{R}) \notin \mathfrak{M}$.

We now use a uniform random split to partition $\mathfrak{D}$ into a training set $\mathfrak{D}_{\text{train}}$ and a test set $\mathfrak{D}_{\text{test}}$ such that $\mathfrak{D}_{\text{train}} \cap \mathfrak{D}_{\text{test}} = \emptyset$ and $\mathfrak{D}_{\text{train}} \cup \mathfrak{D}_{\text{test}} = \mathfrak{D}$. On the basis of this split, we define the following MMP-sets:

$$\begin{aligned}
\mathfrak{M}_{\text{train}} &= \{(\mathcal{R}, \tilde{\mathcal{R}}) \in \mathfrak{M} \mid \mathcal{R}, \tilde{\mathcal{R}} \in \mathfrak{D}_{\text{train}}\}, \\
\mathfrak{M}_{\text{test}} &= \{(\mathcal{R}, \tilde{\mathcal{R}}) \in \mathfrak{M} \mid \mathcal{R}, \tilde{\mathcal{R}} \in \mathfrak{D}_{\text{test}}\}, \\
\mathfrak{M}_{\text{inter}} &= \mathfrak{M} \setminus (\mathfrak{M}_{\text{train}} \cup \mathfrak{M}_{\text{test}}), \\
\mathfrak{M}_{\text{cores}} &= \{(\mathcal{R}, \tilde{\mathcal{R}}) \in \mathfrak{M}_{\text{test}} \mid \text{core}(\mathcal{R}, \tilde{\mathcal{R}}) \notin \mathfrak{C}_{\text{train}}\}.
\end{aligned}$$

Here

$$\mathfrak{C}_{\text{train}} = \{\text{core}(\mathcal{R}, \tilde{\mathcal{R}}) \mid (\mathcal{R}, \tilde{\mathcal{R}}) \in \mathfrak{M}_{\text{train}} \cup \mathfrak{M}_{\text{inter}}\}$$

describes the set of structural MMP-cores that appear in $\mathfrak{D}_{\text{train}}$.

Note that $\mathfrak{M}_{\text{train}} \cup \mathfrak{M}_{\text{inter}} \cup \mathfrak{M}_{\text{test}} = \mathfrak{M}$. The pair $(\mathfrak{D}_{\text{train}}, \mathfrak{M}_{\text{train}})$ describes the training space at the level of individual molecules and MMPs, and can be used to train a QSAR or AC-prediction method. MMPs in $\mathfrak{M}_{\text{test}}$, $\mathfrak{M}_{\text{inter}}$ and $\mathfrak{M}_{\text{cores}}$ can then

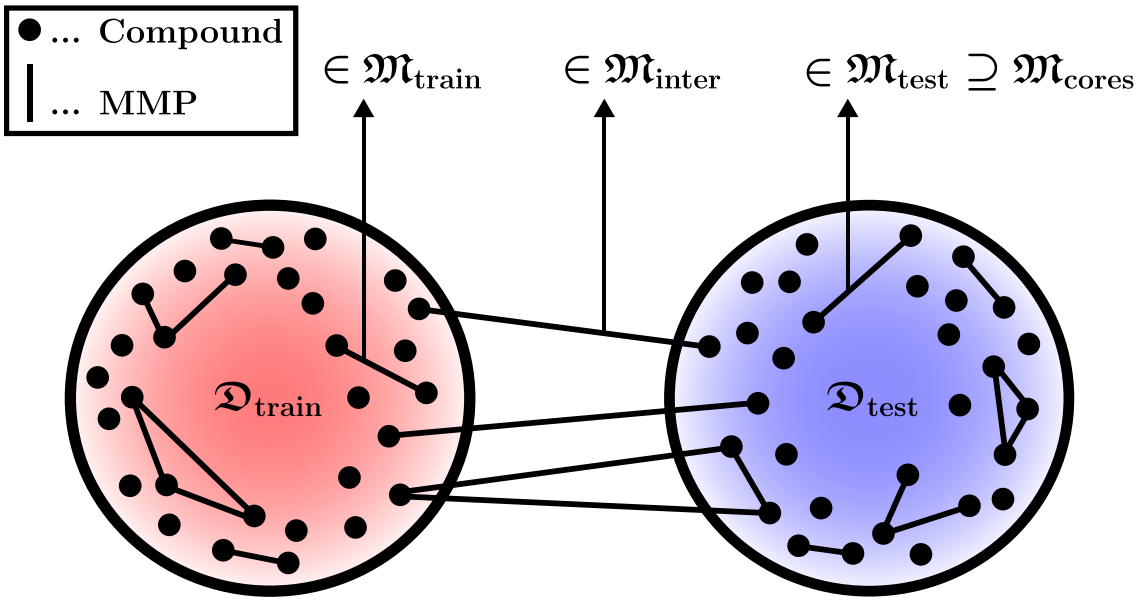

Figure 3.5: Illustration of our data splitting strategy for activity-cliff (AC) and potency-direction (PD) classification. We distinguish between three sets of matched molecular pairs (MMPs), $\mathfrak{M}_{\text{train}}, \mathfrak{M}_{\text{inter}}$ and $\mathfrak{M}_{\text{test}}$, depending on whether both MMP-compounds are in $\mathfrak{D}_{\text{train}}$, one MMP-compound is in $\mathfrak{D}_{\text{train}}$ and the other one is in $\mathfrak{D}_{\text{test}}$, or both MMP-compounds are in $\mathfrak{D}_{\text{test}}$. We additionally consider a fourth MMP-set, $\mathfrak{M}_{\text{cores}}$, consisting of the MMPs in $\mathfrak{M}_{\text{test}}$ whose structural cores do not appear in $\mathfrak{M}_{\text{train}} \cup \mathfrak{M}_{\text{inter}}$.

be classified via a trained AC-predictor. $\mathfrak{M}_{\text{test}}$ models an AC-prediction setting where the activities of both MMP-compounds are unknown. $\mathfrak{M}_{\text{cores}}$ represents the subset of MMPs in $\mathfrak{M}_{\text{test}}$ whose structural cores do not appear in $\mathfrak{M}_{\text{train}} \cup \mathfrak{M}_{\text{inter}}$; $\mathfrak{M}_{\text{cores}}$ thus models the difficult task of predicting ACs within MMPs that do not contain near analogs to MMP-compounds in the training set. Finally, $\mathfrak{M}_{\text{inter}}$ represents an AC-prediction scenario where the activity of one MMP-compound is given *a priori*; this can be interpreted as a compound-optimisation task where one strives to predict small AC-inducing modifications of a query compound with known activity. Arguably the scenario modelled by $\mathfrak{M}_{\text{inter}}$ is the one that is most representative of real-world applications. An illustration of our data splitting strategy is given in 3.5.

We implemented our data splitting strategy within a k-fold cross validation scheme repeated with m random seeds. This generated data splits of the form

$$\mathfrak{S}^{i,j} = (\mathfrak{D}^{i,j}_{\text{train}}, \mathfrak{D}^{i,j}_{\text{test}}, \mathfrak{M}^{i,j}_{\text{train}}, \mathfrak{M}^{i,j}_{\text{test}}, \mathfrak{M}^{i,j}_{\text{inter}}, \mathfrak{M}^{i,j}_{\text{cores}})$$

for $i \in \{1, ..., m\}$ and $j \in \{1, ..., k\}$ whereby the pair $(\mathfrak{D}^{i,j}_{\text{train}}, \mathfrak{D}^{i,j}_{\text{test}})$ represents the j-th split with the i-th random seed of the underlying data set $\mathfrak{D}$ in a k-fold cross validation scheme repeated with m random seeds. The overall performance of each model for all prediction tasks was recorded as the average over the mk training and test runs based

on all data splits $\mathfrak{S}^{1,1}, ..., \mathfrak{S}^{m,k}$. We chose the configuration $(m, k) = (3, 2)$ which gave a good trade-off between computational cost and accuracy and reasonable numbers of MMPs in the compound-pair-sets. In particular, random cross-validation with $k = 2$ gave expected relative sizes of:

$$|\mathfrak{M}_{\text{train}}| : |\mathfrak{M}_{\text{inter}}| : |\mathfrak{M}_{\text{test}}| = 1 : 2 : 1.$$

On average, 12.7%, 11.91%, and 6.84% of MMPs in $\mathfrak{M}_{\text{test}}$ were also in $\mathfrak{M}_{\text{cores}}$ for dopamine receptor D2, factor Xa, and SARS-CoV-2 main protease, respectively.

3.3.4 Prediction Strategies

In a data split of the form

$$\mathfrak{S} = (\mathfrak{D}_{\text{train}}, \mathfrak{D}_{\text{test}}, \mathfrak{M}_{\text{train}}, \mathfrak{M}_{\text{test}}, \mathfrak{M}_{\text{inter}}, \mathfrak{M}_{\text{cores}})$$

each MMP

$$(\mathcal{R}, \tilde{\mathcal{R}}) \in \mathfrak{M}_{\text{train}} \cup \mathfrak{M}_{\text{inter}} \cup \mathfrak{M}_{\text{test}} = \mathfrak{M}$$

comes with a binary label, $\text{AC}(\mathcal{R}, \tilde{\mathcal{R}}) \in \{\text{Non-AC}, \text{AC}\}$, indicating whether $(\mathcal{R}, \tilde{\mathcal{R}})$ is an AC or not and another binary label, $\text{PD}(\mathcal{R}, \tilde{\mathcal{R}}) \in \{\text{Right}, \text{Left}\}$, indicating which of both compounds is more active. Furthermore, each individual compound

$$\mathcal{R} \in \mathfrak{D}_{\text{train}} \cup \mathfrak{D}_{\text{test}} = \mathfrak{D}$$

can be associated with a numerical activity label $\text{act}(\mathcal{R}) \in \mathbb{R}$ which we define as the negative decadic logarithm of the experimentally measured activity of $\mathcal{R}$. We stuck with the original units used in the ChEMBL database and the COVID moonshot project before applying the logarithm ([nM] for K_i and [µM] for IC_{50}); each activity label $\text{act}(\mathcal{R})$ thus represents a standard pK_i or pIC_{50} value with a slight additive shift towards 0 caused by the use of [nM] or [µM]-units instead of the canonical [M]-units; this shift towards 0 might slightly benefit prediction techniques initialised around the origin.

We are interested in QSAR-prediction functions

$$Q : \mathfrak{D} \to \mathbb{R}$$

that map a given molecular representation $\mathcal{R} \in \mathfrak{D}$ to an estimate of its binding affinity:

$$Q(\mathcal{R}) \approx \text{act}(\mathcal{R}).$$

In our study, the molecular representation $\mathcal{R}$ is either a SMILES string or a molecular graph. The mapping Q is found via an algorithmic training process on the labelled data set

$$\{(\mathcal{R}, \mathrm{act}(\mathcal{R})) \mid \mathcal{R} \in \mathfrak{D}_{\mathrm{train}}\}$$

and can then either be used to predict the activity labels of compounds in $\mathfrak{D}_{\mathrm{test}}$, or it can be repurposed to classify whether an MMP forms an activity cliff (AC-classification) and what the potency direction of an MMP is (PD-classification).

If $(\mathcal{R}, \tilde{\mathcal{R}}) \in \mathfrak{M}_{\mathrm{inter}}$, then one can assume that the activity label of one of the compounds, say $\mathrm{act}(\mathcal{R})$, is known. Q is then used to generate an AC-classification for $(\mathcal{R}, \tilde{\mathcal{R}})$ via

$$(\mathcal{R}, \tilde{\mathcal{R}}) \mapsto \begin{cases} \text{Non-AC} & \text{if } |\mathrm{act}(\mathcal{R}) - Q(\tilde{\mathcal{R}})| \leq d_{\mathrm{crit}}, \\ \text{AC} & \text{else.} \end{cases}$$

Here $d_{\mathrm{crit}} \in \mathbb{R}_{>0}$ is a critical threshold above which an MMP is classified as an AC. Throughout this work we use $d_{\mathrm{crit}} = 1.5$ (in $\mathrm{pK_i}$ or $\mathrm{pIC_{50}}$ units) since this value represents the middle point between the intervals $[0, 1]$ and $[2, \infty)$ which correspond to absolute logarithmic activity differences associated with non-ACs and ACs respectively. If $(\mathcal{R}, \tilde{\mathcal{R}}) \in \mathfrak{M}_{\mathrm{test}} \cup \mathfrak{M}_{\mathrm{cores}}$ then $\mathcal{R}, \tilde{\mathcal{R}} \in \mathfrak{D}_{\mathrm{test}}$ and therefore the activities of both compounds are unknown. We hence perform the AC-classification for $(\mathcal{R}, \tilde{\mathcal{R}})$ via

$$(\mathcal{R}, \tilde{\mathcal{R}}) \mapsto \begin{cases} \text{Non-AC} & \text{if } |Q(\mathcal{R}) - Q(\tilde{\mathcal{R}})| \leq d_{\mathrm{crit}}, \\ \text{AC} & \text{else.} \end{cases}$$

The PD-classification for an MMP $(\mathcal{R}, \tilde{\mathcal{R}}) \in \mathfrak{M}_{\mathrm{inter}}$ with Q is performed in a straightforward manner by simply comparing the binding affinity prediction of the test compound $\tilde{\mathcal{R}}$ with the experimentally measured binding affinity of the training compound $\mathcal{R}$:

$$(\mathcal{R}, \tilde{\mathcal{R}}) \mapsto \begin{cases} \text{Right} & \text{if } \mathrm{act}(\mathcal{R}) \leq Q(\tilde{\mathcal{R}}), \\ \text{Left} & \text{else.} \end{cases}$$

Similarly, the PD-classification for an MMP $(\mathcal{R}, \tilde{\mathcal{R}}) \in \mathfrak{M}_{\mathrm{test}} \cup \mathfrak{M}_{\mathrm{cores}}$ with Q is performed via:

$$(\mathcal{R}, \tilde{\mathcal{R}}) \mapsto \begin{cases} \text{Right} & \text{if } Q(\mathcal{R}) \leq Q(\tilde{\mathcal{R}}), \\ \text{Left} & \text{else.} \end{cases}$$

3.3.5 Performance Metrics

The performance of Q when used as a standard QSAR method for the prediction of the activity labels of individual molecules in $\mathfrak{D}_{\text{test}}$ was measured via the mean absolute error (MAE):

$$\frac{1}{|\mathfrak{D}_{\text{test}}|} \sum_{\mathcal{R} \in \mathfrak{D}_{\text{test}}} |Q(\mathcal{R}) - \text{act}(\mathcal{R})|.$$

For the balanced PD-classification problem we could rely on simple accuracy as a suitable performance metric:

$$\frac{\text{number of correct predictions}}{\text{number of predictions}}.$$

However, when using Q for the naturally highly imbalanced AC-classification task, a more nuanced set of performance metrics had to be chosen. Denote with TP, TN, FP, and FN the numbers respectively representing true positives, true negatives, false positives and false negatives for the AC-classification task. We then used the Matthews correlation coefficient (MCC) as a suitable overall performance metric:

$$\frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}.$$

In addition, we tracked AC-sensitivity

$$\frac{TP}{TP + FN}$$

and AC-precision

$$\frac{TP}{TP + FP}.$$

The MCC represents a summary statistic for the confusion matrix of a binary classification problem that is reasonably robust against imbalanced class labels. Sensitivity (also known as recall) can be interpreted as an approximation for the probability of the classifier to classify an actually positive instance as a positive one. Finally, precision (also known as positive predictive value) approximates the probability that a positively classified instance is indeed positive.

For the relatively small SARS-CoV-2 main protease data set we sometimes encountered the edge case where $\text{TP} + \text{FP} = 0$, i.e. where there were no positive predictions. In this situation we set $\text{MCC} = 0$ and ignored the ill-defined precision measurements when averaging the performance metrics.

Molecular Featurisation Methods and Regression Techniques

We constructed nine QSAR models (i.e. nine versions of Q) via a robust combinatorial methodology that systematically combines three molecular featurisation methods with three regression techniques. This setup allows one to systematically compare the performance of molecular featurisations across regression techniques, data sets and predictions tasks. For molecular featurisation, we used PDVs (2.4) and ECFPs (2.5), both generated from SMILES strings (2.3), as well as GINs (2.6.3) on top of molecular graphs (2.2). Both the ECFPs and the PDVs were computed via `RDKit` [70]. The ECFPs used a radius of two, a length of 2048 bits, and standard atom features with active tetrahedral R-S chirality flags. The PDVs had a dimensionality of 200, were constructed using the list of descriptors specified in Table 2.2, and were normalised to lie in the hypercube $[0, 1]^{200}$ via their componentwise cumulative distribution functions derived from the training set as explained in Section 2.4. The PDV descriptor-list encompassed properties related to druglikeness, logP, molecular refractivity, electrotopological state, molecular graph-structure, fragment-profile, molecular charge, and molecular surface. The GIN was implemented via `PyTorch Geometric` [96]. The atom features of the underlying molecular graph objects can be found in Table 2.1. For global graph pooling we chose the componentwise maximum over all atom feature vectors in the final graph-layer.

Each molecular featurisation was used as an input featurisation for three regression techniques: random forests (RFs), k-nearest neigbours (kNNs) and multilayer perceptrons (MLPs). The RF and kNN-models were implemented via scikit-learn [130] and the MLP-models via `PyTorch` [131]. The MLPs used ReLU activations and batch normalisation at each hidden layer. The GIN was combined with the regression techniques as follows: For MLP regression, the GIN was trained with the MLP as a projection head after the pooling step in the usual end-to-end manner. For RF or kNN-regression, the GIN was first trained with a single linear layer added after the global pooling step that directly mapped the graph-level representation to an activity prediction. After this training phase the weights of the GIN were frozen and it was used as a static feature extractor. The RF or kNN-regressor was then trained on top of the features extracted by the frozen GIN. Figure 3.6 depicts an overview of all investigated QSAR models.

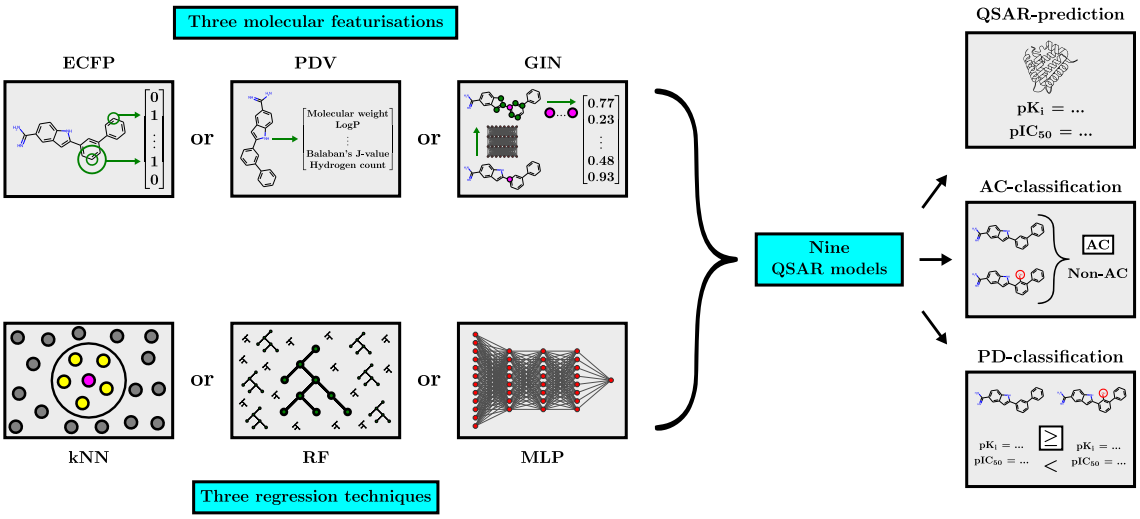

Figure 3.6: Schematic showing the combinatorial experimental methodology used for the study. Each molecular featurisation method is systematically combined with each regression technique, giving a total of nine quantitative structure-activity relationship (QSAR) models. Each QSAR model is trained and evaluated for QSAR-prediction, activity-cliff (AC) classification and potency-direction (PD) classification via 2-fold cross validation repeated with 3 random seeds. For each of the $2 * 3 = 6$ trials, an extensive inner hyperparameter-optimisation loop on the training set is performed for each QSAR model.

3.3.6 Model Training and Hyperparameter Optimisation

The three investigated regression techniques as well as the GIN feature-extractor come with a multitude of training and model hyperparameters which need to be tuned properly as to allow each QSAR model to unfold its maximum performance. Since our presented study is comparative in nature, it was essential to include a systematic hyperparameter-optimisation procedure within the training routine of each model. While such a hyperparameter optimisation greatly increased the computational cost and the difficulty of implementation for our numerical experiments, it formed an indispensable part of a fair and objective model comparison.

As described previously, each QSAR model was evaluated within a k-fold cross validation scheme repeated with m random seeds. This implies that an independent version of each QSAR model was trained on each molecular training set $\mathfrak{D}^{i,j}_{\text{train}}$ for $i \in \{1, ..., m\}$ and $j \in \{1, ..., k\}$ and the results were then averaged for each model over all mk trials. At each of the mk training rounds, each model was first trained via an inner hyperparameter-optimisation loop on $\mathfrak{D}^{i,j}_{\text{train}}$. The determined set of hyperparameters was then used to train a model with optimised architecture on $\mathfrak{D}^{i,j}_{\text{train}}$ and this optimised model was used for evaluation. In the implementation of the inner hyperparameter-optimisation loop we distinguished between the four classical

machine learning models (ECFPs or PDVs combined with either RFs or kNNs) and the five models that contained a deep learning component (a GIN or an MLP) in order to save computational resources.

In the classical case we used a five-fold inner cross validation split on $\mathfrak{D}^{i,j}_{\text{train}}$; 10 models with distinct hyperparameter settings were then trained within this cross validation scheme. The 10 hyperparameter configurations were sampled uniformly at random from a predefined grid using the RandomizedSearchCV routine implemented in scikit-learn [130]. The hyperparameters that minimised the MAE over the inner cross validation loop were subsequently chosen to train the final model. For RF regression, we chose a forest size of 500 trees and optimised the maximum tree depth, the minimum number of samples required to split an internal node, the minimum number of samples required to be at a leaf node, the number of features to consider for the best split, and whether bootstrap samples should be used or not when building trees. For the kNN algorithm, we optimised the number of considered neighbours, the power parameter for the underlying Minkowski distance measure, and whether to give uniform or inverse-distance weights to neighbours.

In the deep learning case, we employed a 4:1 split of $\mathfrak{D}^{i,j}_{\text{train}}$ into an inner training set and an inner validation set. We then trained 20 models with distinct hyperparameter configurations on the inner training set. The 20 hyperparameter sets were sampled from a predefined grid using the tree-structured Parzen estimator (TPE) algorithm implemented in the Optuna hyperparameter-optimisation package [132]. The hyperparameters that minimised the MAE on the inner validation set were subsequently chosen to train the final model. For the MLP architecture, we optimised the number of hidden layers and the number of neurons per hidden layer. Additionally, we chose ReLU as the hidden activation function and used batch normalisation [133] throughout the neural network.

For the GINs (2.6.3) we optimised the total number of graph layers R and the dimensional length of the updated atom feature vectors $f_r(a)$ at the r-th graph layer. For each layerwise multilayer perceptron ϕ_r that formed part of the GIN we used two internal hidden layers whereby the number of neurons in each hidden layer was equivalent to the dimensionality of the atom feature vectors in the r-th layer. For the two hidden layers of each ϕ_r we employed batch normalisation and again chose ReLU as the hidden activation function. Each ϵ_r was set to 0. To reduce the molecular graph to a single feature vector after the message-passing phase, we employed max-pooling at the R-th graph layer which computes the componentwise maximum over all final atom feature vectors. Since we optimised the atom feature dimensionality

in the final graph layer we also implicitly optimised the dimensionality l of the final graph-level fingerprint.

All deep learning models were trained for 500 epochs on a single NVIDIA GeForce RTX 3060 GPU using the mean squared error loss function and AdamW optimisation [134]. During training we employed weight decay, learning rate decay and dropout [135] at all hidden layers for regularisation. Batch size, learning rate, learning rate decay rate, weight decay rate, and dropout rate were treated as hyperparameters and subsequently optimised. Note that the training length (the number of gradient updates) was implicitly optimised via tuning the batch size for the fixed number of 500 training epochs.

3.4 Results and Discussion

The QSAR-prediction, AC-classification and PD-classification results for all three investigated data sets are depicted in Figures 3.7 to 3.12 below.

3.4.1 QSAR-Prediction Performance

When considering the results depicted in Figures 3.7 to 3.12 with respect to QSAR-prediction performance, one can see that ECFPs tend to lead to better performance (i.e. a lower QSAR-MAE) compared to GINs, which in turn tend to lead to better performance compared to PDVs. In particular, the combination ECFP-MLP consistently produced the lowest QSAR-MAE across all three targets. These observations reinforce a growing corpus of literature that suggests that trainable GNNs have not yet reached a level of technical maturity by which they consistently and definitively outperform the much simpler task-agnostic ECFPs at important molecular property prediction problems [6, 7, 8, 10, 11, 12, 13].

3.4.2 AC-Classification Performance

The AC-MCC plots in Figures 3.7 to 3.9 reveal surprisingly strong overall AC-classification results on $\mathfrak{M}_{\text{inter}}$. This type of MMP-set models a compound-optimisation scenario where a researcher strives to identify small structural modifications with a large impact on the activity of query compounds with known activities. For this task, a substantial number of our QSAR models exhibit an AC-MCC value greater than 0.5 across targets, which appears impressive considering the simplicity of the approach. Exchanging $\mathfrak{M}_{\text{inter}}$ with either $\mathfrak{M}_{\text{test}}$ or $\mathfrak{M}_{\text{cores}}$ leads to a substantial drop

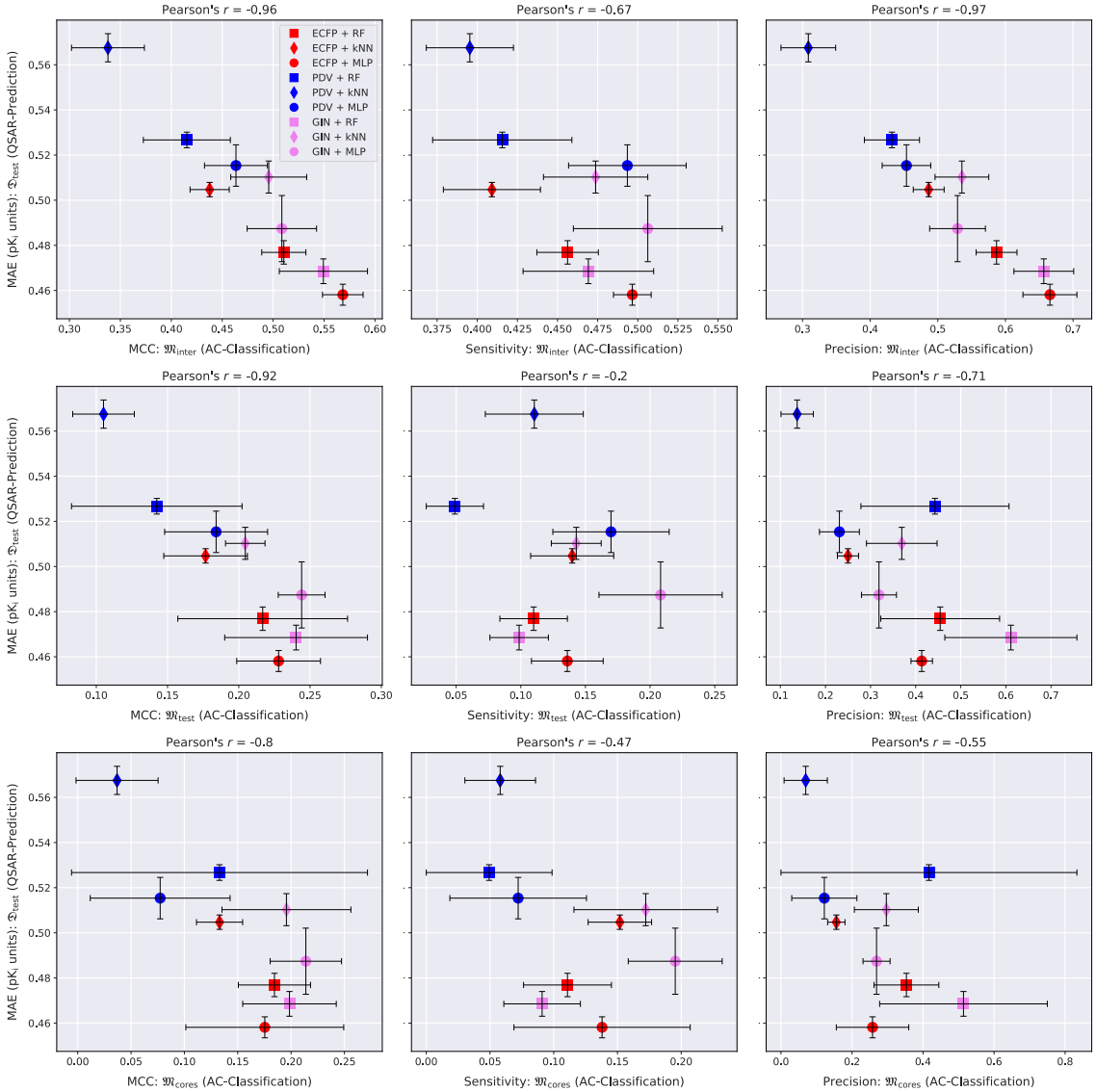

Figure 3.7: QSAR-prediction and activity-cliff (AC) classification results for **dopamine receptor D2**. For each plot, the x-axis corresponds to a combination of MMP-set and AC-classification performance metric and the y-axis shows the QSAR-prediction performance on the molecular test set $\mathfrak{D}_{\text{test}}$. The total length of each error bar equals twice the standard deviation of the performance metric measured over all $mk = 3 * 2 = 6$ hyperparameter-optimised models. For each plot, the lower right corner corresponds to strong performance at both prediction tasks.

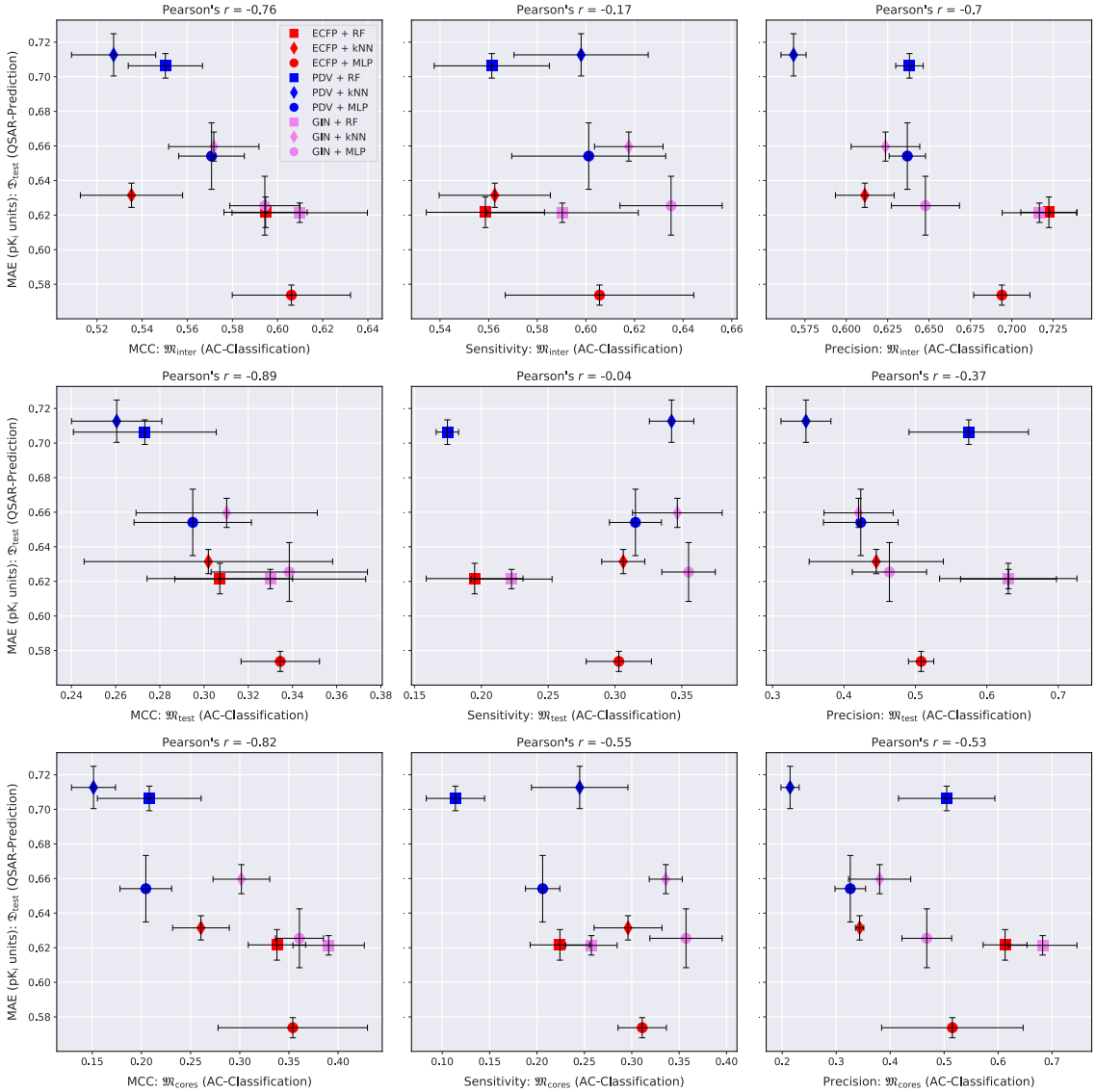

Figure 3.8: QSAR-prediction and activity-cliff (AC) classification results for **factor Xa**. For each plot, the x-axis corresponds to a combination of MMP-set and AC-classification performance metric and the y-axis shows the QSAR-prediction performance on the molecular test set $\mathfrak{D}_{\text{test}}$. The total length of each error bar equals twice the standard deviation of the performance metric measured over all $mk = 3 * 2 = 6$ hyperparameter-optimised models. For each plot, the lower right corner corresponds to strong performance at both prediction tasks.

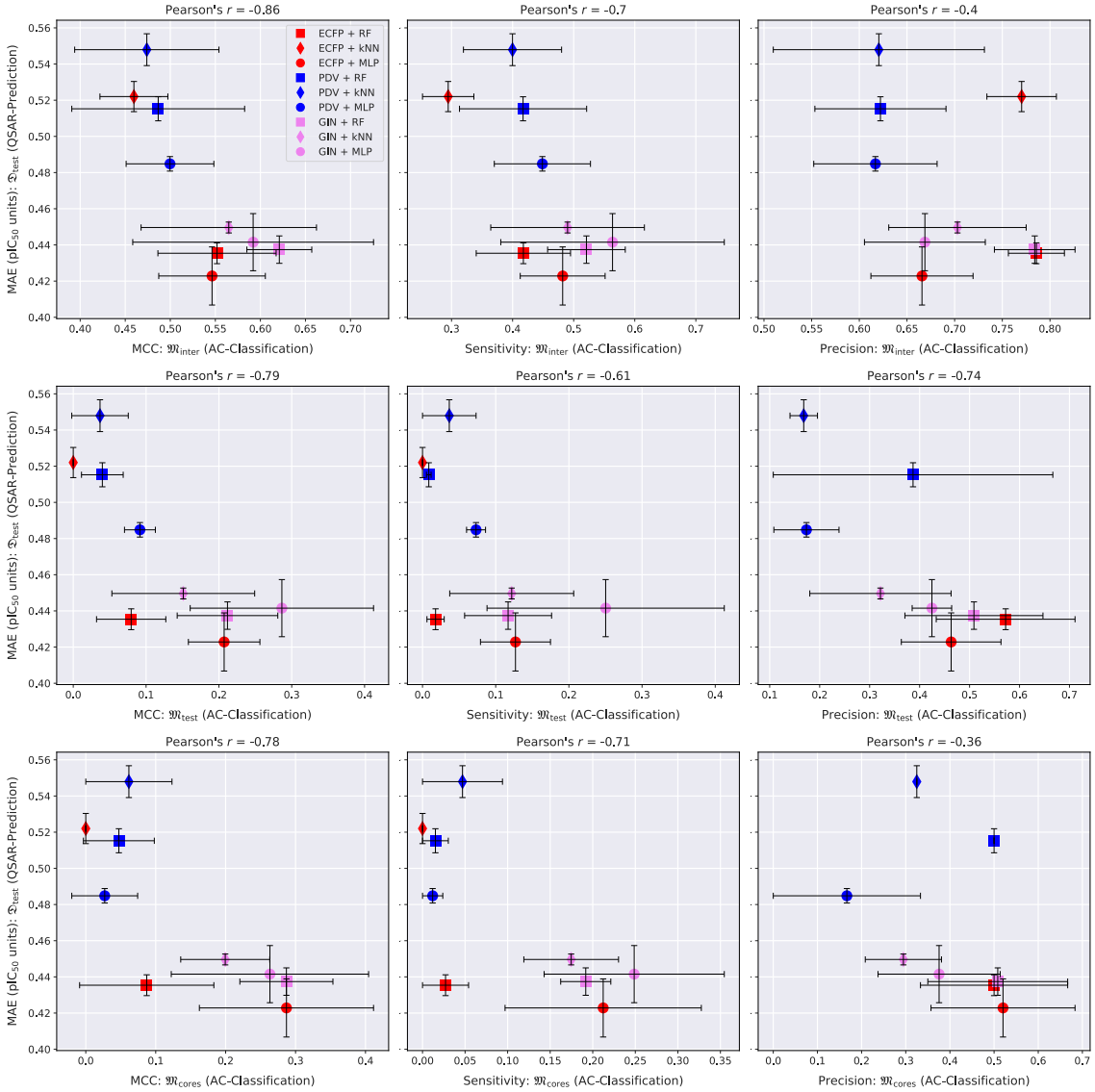

Figure 3.9: QSAR-prediction and activity-cliff (AC) classification results for **SARS CoV-2 main protease**. For each plot, the x-axis corresponds to a combination of MMP-set and AC-classification performance metric and the y-axis shows the QSAR-prediction performance on the molecular test set $\mathfrak{D}_{\text{test}}$. The total length of each error bar equals twice the standard deviation of the performance metric measured over all $mk = 3 * 2 = 6$ hyperparameter-optimised models. The precision of the AC-classification task is not shown for the ECFP + kNN technique on $\mathfrak{M}_{\text{test}}$ and $\mathfrak{M}_{\text{cores}}$ since this method produced only negative AC-classifications for all trials on this data set. For each plot, the lower right corner corresponds to strong performance at both prediction tasks.

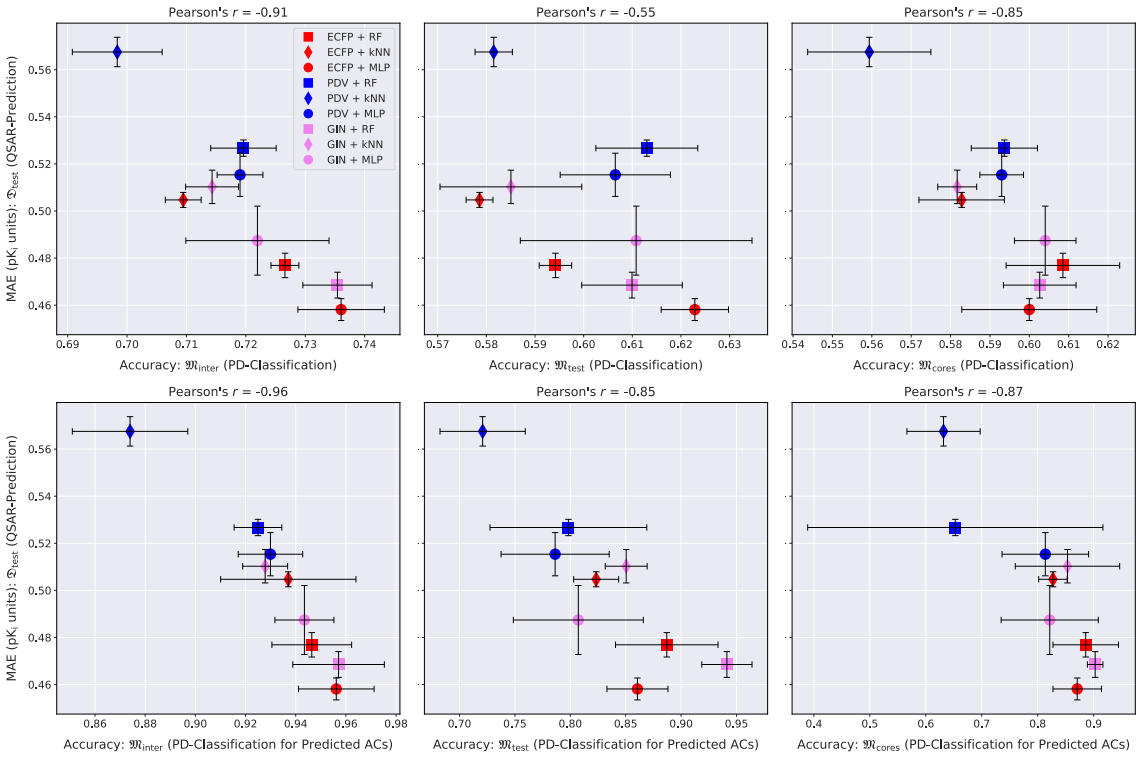

Figure 3.10: QSAR-prediction and potency-direction (PD) classification results for **dopamine receptor D2**. Each column corresponds to an upper plot and a lower plot for one of the MMP-sets $\mathfrak{M}_{\text{inter}}$, $\mathfrak{M}_{\text{test}}$ or $\mathfrak{M}_{\text{cores}}$. The x-axis of each upper plot indicates the PD-classification accuracy on the full MMP-set; the x-axis of each lower plot indicates the PD-classification accuracy on a restricted MMP-set only consisting of MMP predicted to be ACs by the respective method. The y-axis of each plot shows the QSAR-prediction performance on the molecular test set $\mathfrak{D}_{\text{test}}$. The total length of each error bar equals twice the standard deviation of the performance metrics measured over all $mk = 3 * 2 = 6$ hyperparameter-optimised models. For each plot, the lower right corner corresponds to strong performance at both prediction tasks.

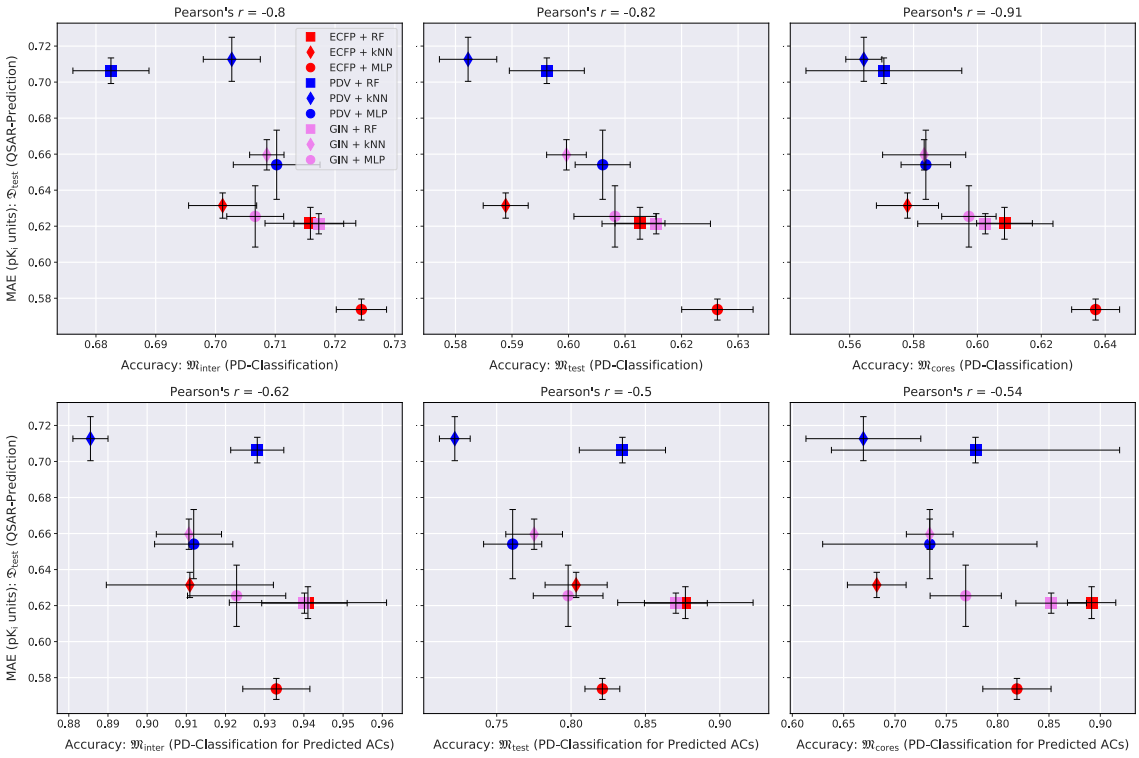

Figure 3.11: QSAR-prediction and potency-direction (PD) classification results for **factor Xa**. Each column corresponds to an upper plot and a lower plot for one of the MMP-sets $\mathfrak{M}_{\text{inter}}$, $\mathfrak{M}_{\text{test}}$ or $\mathfrak{M}_{\text{cores}}$. The x-axis of each upper plot indicates the PD-classification accuracy on the full MMP-set; the x-axis of each lower plot indicates the PD-classification accuracy on a restricted MMP-set only consisting of MMP predicted to be ACs by the respective method. The y-axis of each plot shows the QSAR-prediction performance on the molecular test set $\mathfrak{D}_{\text{test}}$. The total length of each error bar equals twice the standard deviation of the performance metrics measured over all $mk = 3 * 2 = 6$ hyperparameter-optimised models. For each plot, the lower right corner corresponds to strong performance at both prediction tasks.

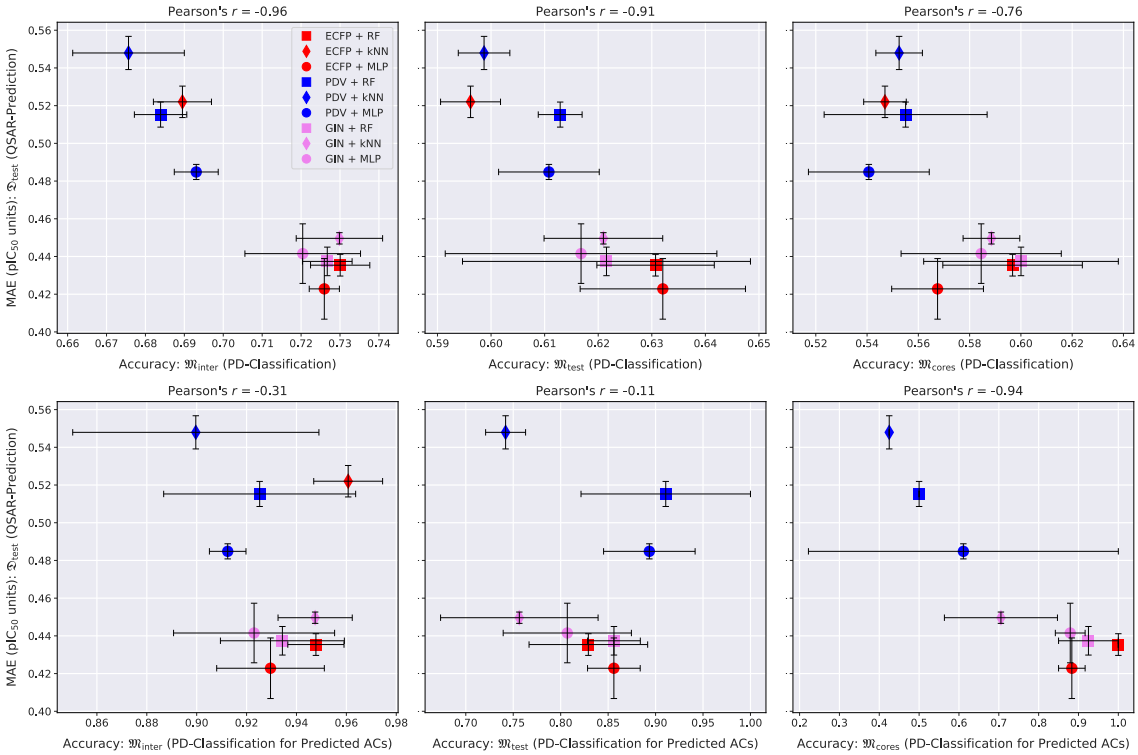

Figure 3.12: QSAR-prediction and potency-direction (PD) classification results for **SARS-CoV-2 main protease**. Each column corresponds to an upper plot and a lower plot for one of the MMP-sets $\mathfrak{M}_{\text{inter}}$, $\mathfrak{M}_{\text{test}}$ or $\mathfrak{M}_{\text{cores}}$. The x-axis of each upper plot indicates the PD-classification accuracy on the full MMP-set; the x-axis of each lower plot indicates the PD-classification accuracy on a restricted MMP-set only consisting of MMP predicted to be ACs by the respective method. The y-axis of each plot shows the QSAR-prediction performance on the molecular test set $\mathfrak{D}_{\text{test}}$. The total length of each error bar equals twice the standard deviation of the performance metrics measured over all $mk = 3 * 2 = 6$ hyperparameter-optimised models. The accuracy of the PD-classification task for predicted ACs is not shown for the ECFP + kNN technique on $\mathfrak{M}_{\text{test}}$ and $\mathfrak{M}_{\text{cores}}$ since this method produced only negative AC-classifications for all trials on this data set. For each plot, the lower right corner corresponds to strong performance at both prediction tasks.

in the AC-MCC to approximately 0.3 that appears to be mediated by a large drop in AC-sensitivity.

In most cases, GINs perform better than the other molecular featurisation methods with respect to the AC-MCC. Notably, the combination GIN-kNN consistently performs considerably better for AC-classification than the combinations ECFP-kNN and PDV-kNN. This supports the idea that GINs might have a heightened ability to resolve ACs by learning an embedding of chemical space in which the distance between two compounds is reflective of activity difference rather than structural difference. The combinations GIN-MLP, GIN-RF and ECFP-MLP exhibit particularly high AC-MCC values relative to the other methods. We recommend using at least one of these three models as a baseline against which to compare tailored AC-classification models; the practical utility of any AC-classification technique that cannot outperform these three common QSAR methods is questionable.

Across all three targets, AC-sensitivity is moderately high on $\mathfrak{M}_{\text{inter}}$ but universally low on $\mathfrak{M}_{\text{test}}$ and $\mathfrak{M}_{\text{cores}}$. This is consistent with the hypothesis that ACs form one of the major sources of prediction error for QSAR models. The weak AC-sensitivity on $\mathfrak{M}_{\text{test}}$ and $\mathfrak{M}_{\text{cores}}$ indicates that modern QSAR methods are largely blind to ACs formed by two MMP-compounds outside the training set and thus lack essential chemical knowledge. GINs clearly outperform the other two more classical molecular featurisations across regression techniques with respect to AC-sensitivity. In particular, the GIN-MLP combination leads to the highest AC-sensitivity in all examined cases and thus discovers the most ACs. The highly parametric nature of GINs that makes them prone to overfitting could at the same time enable them to better model jagged regions of the SAR-landscape that contain ACs than classical task-agnostic representations.

There is a wide gap between distinct prediction techniques with respect to AC-precision: some models achieve a considerable level of AC-precision such that over 50% of positively predicted MMPs in $\mathfrak{M}_{\text{test}}$ and $\mathfrak{M}_{\text{cores}}$ are indeed actual ACs. Other QSAR models, however, seem to fail almost entirely with respect to this metric on $\mathfrak{M}_{\text{test}}$ and $\mathfrak{M}_{\text{cores}}$ and only deliver modest performance on $\mathfrak{M}_{\text{inter}}$. RFs tend to exhibit the strongest AC-precision and the weakest AC-sensitivity. This might be as a result of their ensemble nature which should intuitively lead to conservative but trustworthy predictions of extreme effects such as ACs.

3.4.3 PD-Classification Performance

The abilities of the evaluated QSAR models to identify which compound in an MMP is more active is universally weak, with PD-accuracies clustering around 0.7 on $\mathfrak{M}_{\text{inter}}$ and around 0.6 on $\mathfrak{M}_{\text{test}}$ and $\mathfrak{M}_{\text{cores}}$, as can be seen in the top rows of Figures 3.10 to 3.12. Predicting the potency direction for two compounds with similar structures and thus usually similar levels of activity must be considered a challenging task. The combination ECFP-MLP reaches the strongest PD-accuracy in the majority of cases and we recommend starting with this model as a baseline for more advanced PD-classification methods.

One can argue that the activity direction of two similar compounds is of little interest if the true activity difference is small, as is often the case. We therefore also restricted PD-classification to predicted ACs. The three plots in the bottom rows of Figures 3.10 to 3.12 depict the PD-accuracy of each QSAR model on the subset of MMPs that were also predicted to be ACs by the same model. In this practically more relevant scenario, PD-classification accuracy tends to exceed 0.9 on $\mathfrak{M}_{\text{inter}}$ and 0.8 on $\mathfrak{M}_{\text{test}}$ and $\mathfrak{M}_{\text{cores}}$. The QSAR models investigated here are thus able to identify the correct activity direction of MMPs if they also predict them to be ACs. The relatively rare instances in which the PD of a predicted AC is misclassified, however, reflect severe QSAR-prediction errors.

3.4.4 Linear Relationship between QSAR-MAE and AC-MCC

Our experiments reveal a consistent linear functional relationship between the QSAR-MAE and the AC-MCC as can be seen in the left columns of Figures 3.7 to 3.9. A potential mechanism driving this effect could be as follows: As the overall QSAR-MAE of a model improves, its accuracy at predicting activity differences between similar molecules could be expected to improve as well. Previously misclassified MMPs whose predicted absolute activity differences were already close to the critical value $d_{\text{crit}} = 1.5$ might then gradually move to the correct side of the decision boundary and increase the AC-MCC. These results suggest that for real-world QSAR models the AC-MCC and the QSAR-MAE are strongly predictive of each other, i.e. there appears to be a strong positive association between the general ability of a QSAR model to predict activities of individual compounds and its capability to correctly distinguish between ACs and non-ACs. While this observation only rests on nine models, it is highly consistent across MMP-sets and pharmacological targets. Since

AC-precision also appears to reliably increase as the QSAR-MAE decreases, one can speculate that as the QSAR-prediction performance of a model gets better, it gradually removes "false spikes" from its generated SAR-landscape that would otherwise result in the prediction of false AC-positives.

3.5 Conclusions

To the best of our knowledge this is the first study to investigate the capabilities of QSAR models to classify the existence and direction of ACs within pairs of similar compounds. It is also the first work to explore the quantitative relationship between QSAR-prediction at the level of individual molecules and AC-prediction at the level of compound-pairs. As part of our methodology we have additionally introduced a simple, interpretable, and rigorous data-splitting technique for pair-based prediction problems.

When the activities of both MMP-compounds are unknown (i.e. absent from the training set) then common QSAR models exhibit low AC-sensitivity which limits their utility for AC-classification. This strongly supports the hypothesis that QSAR methods do indeed regularly fail to predict ACs which might thus form a major source of prediction errors in QSAR modelling [39, 40, 41, 97]. However, in the practically significantly more relevant scenario where the activity of one MMP-compound is known (i.e. present in the training set) AC-sensitivity increases substantially; for query compounds with known activities, QSAR methods can therefore be used as simple AC-classification, compound-optimisation and SAR-knowledge-acquisition tools. Furthermore, based on the observed PD-classification results, we can expect the predicted direction of predicted ACs to have a high degree of accuracy.

With respect to molecular featurisation, we have found PDVs to be consistently inferior to ECFPs and GINs at both QSAR-prediction and AC-classification. It might be the case that simply too much of the explicit structural information that is relevant for both tasks is lost during the task-agnostic PDV-transformation. Moreover, we have found robust evidence that precomputed ECFPs do not only outcompete PDVs but also differentiable GINs at general QSAR-prediction. This adds to a growing awareness that standard message-passing GNNs might need to be improved further to definitively beat classical molecular featurisations based on structural fingerprints such as ECFPs [6, 7, 8, 10, 11, 12, 13]. One potential angle to achieve this could be self-supervised GNN pre-training, which has recently shown promising results in the molecular domain [9, 21]. However, while GINs appear to be inferior to ECFPs

at QSAR-prediction, they tend to be advantageous for AC-classification; their highly parametric nature might simultaneously lead to increased overfitting but to a better modelling of the more jagged regions of the SAR-landscape. We thus recommend using GINs as an AC-classification baseline since such an agreed-upon benchmark is currently lacking.

Finally, the low AC-sensitivity of the tested QSAR models when the activities of both MMP-compounds are unknown suggests that such methods are still lacking essential SAR knowledge. On the flip side, one can speculate that it might be possible to considerably boost the performance of common QSAR models in the future by focussing on the development of techniques to specifically increase their AC-sensitivity.

Chapter 4

A Twin Neural Network Model for Activity-Cliff Prediction

We have presented an early version of our twin neural network model for activity-cliff prediction described in this chapter at the 4th RSC-BMCS / RSC-CICAG Artificial Intelligence in Chemistry Symposium (2021, virtual) where we were subsequently awarded the prize for the best scientific poster [47]. The material in this chapter is built on the ideas outlined in our poster.

4.1 Overview

In Chapter 3 we have seen that the utility of standard QSAR models for the detection of ACs is limited if the activites of both compounds are unknown and that ACs do in fact form a major source of prediction error in such cases. However, a method to predict ACs accurately *in silico* would still be of great value for computational drug discovery due to its potential utility for tasks such as compound optimisation and SAR-knowledge acquisition. A natural research question to ask is thus how to design an AC-prediction model that provably outperforms the QSAR-modelling baselines established in the previous chapter.

As mentioned in Section 3.2, the AC-prediction literature is still thin compared to the QSAR-prediction literature. A generous and thorough literature search for computational AC-prediction models revealed a total of 15 methods [51, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116], all of which were published since 2012. The core difference between these tailored AC-prediction models is their respective method to extract features from pairs of molecular compounds in a manner suitable for standard machine learning pipelines. Pair-based feature extraction techniques that have been employed for AC-prediction include condensed graphs of reactions [111], convolutional neural networks operating on 2D images of compound pairs [51], and

newly designed kernel functions for support-vector-machine classification of compound pairs [104]. One of the major flaws of the published work on AC-prediction is that none of the most salient AC-prediction models [51, 104, 105, 106, 111] have been compared to a simple QSAR model. It is thus unclear whether these technically complex methods do in fact outcompete the simple QSAR-based AC-classification baselines established in Chapter 3. This casts doubt on their practical utility. Running control experiments that compare the AC-prediction model to a straightforward QSAR-modelling baseline is especially relevant in light of the fact that a variety of published AC-prediction methods [51, 104, 106] are tested on compound-pair-based data splits which incur a large overlap between training and test set at the level of individual compounds. This less-than-rigorous data splitting technique should naturally favour QSAR models for AC-prediction.

In this chapter, we introduce a novel deep learning model for the classification of the existence and direction ACs in chemical space. Our key idea is to employ a twin architecture [136, 137, 138, 139] that is specifically designed to process dual inputs in a natural way. Twin networks are artificial neural network architectures that contain two (or more) indistinguishable copies of the same subnetwork. All subnetworks are required to share the same architecture and trainable parameters, and these parameters are updated jointly for all subnetworks during training. Twin architectures have been shown to be well-suited to learn similarity and distance metrics for pairs of complex data structures [136]; such metric-learning approaches can be used for data-scarce prediction tasks in a process called single-shot learning. Successful areas of application for twin neural networks can for example be found in facial recognition [139], signature verification [137] and single-shot image recognition [138]. Comparatively little work has been done, however, to study twin neural networks in the context of computational drug discovery. A small number of studies have investigated the potential of twin architectures for drug-drug interaction prediction [140, 141, 142], one-shot drug-discovery [143, 144, 145], protein-protein interaction prediction [146], bioactivity prediction [147], drug-response similarity prediction [148], compound-structure determination [149], protein-representation learning [150], and the identification of drug-target interactions [151]. To the best of our knowledge, our work represents the first application of twin neural networks to the problem of activity-cliff prediction.

Our proposed twin network is jointly trained on two distinct prediction tasks: a ternary AC-classification task to predict whether an input MMP represents an AC, a half-AC or a non-AC, and a binary PD-classification task to predict which compound

in the MMP is the more potent one. Important symmetry properties can be hard-coded into the neural architecture of the twin network as a useful inductive bias for pair-based prediction problems, and we give straightforward mathematical proofs that illustrate these properties. The developed twin network can be seamlessly combined with either modern GNNs or classical molecular featurisations such as ECFPs or PDVs, or indeed with any representation of individual molecules. This removes the need to develop complex feature-engineering procedures for compound-pairs, which appears to have been the main technical hurdle in the development of previous AC-prediction models [51, 104, 105, 111].

We experimentally evaluate an ECFP-based and a GIN-based version of the proposed twin model using the SARS-CoV-2 main protease data set described in Section 3.3.1 and our novel rigorous data splitting technique for pair-based data developed in Section 3.3.3. We additionally experiment with a transfer learning approach to enrich the input features of each of the two model versions in an attempt to further boost performance. We also run strict control experiments to compare the twin network models to two QSAR models that have shown strong performance in Chapter 3 when repurposed for AC-classification, namely ECFP-MLP and GIN-MLP. However, to guarantee comparability with the twin network, this time we will also repurpose the QSAR models for *ternary* rather than binary AC-classification. As stated before, such indispensable control experiments involving QSAR models are lacking in other studies [51, 104, 105, 111].

We start off this chapter by giving a mathematical description and visual illustration of the proposed twin neural network model for AC and PD-classification, along with the transfer learning technique to improve the information-content of its input features. We then present our computational experiments and discuss our empirical observations. Finally, we summarise our findings and draw conclusions for the AC-prediction field.

4.2 Twin Neural Network: Mathematical Description

4.2.1 Neural Architecture and Symmetry Properties

Let $(\mathcal{R}, \tilde{\mathcal{R}})$ be an ordered pair of two molecular representations forming an MMP. An example for an MMP is depicted in the previous chapter in Figure 3.1. Just

like in Chapter 3, we assume that both compounds are associated with activity labels $\mathrm{act}(\mathcal{R}), \mathrm{act}(\tilde{\mathcal{R}}) \in \mathbb{R}$ which quantify their biological activity with respect to the same predefined pharmacological target. More specifically, we consider the expression $\mathrm{act}(\mathcal{R})$ to be the negative decadic logarithm of the experimentally measured activity value of $\mathcal{R}$. This is equivalent to its $\mathrm{pK_i}$ or $\mathrm{pIC_{50}}$-value (up to a minor additive shift if one chooses to use units other than the canonical [M]-units). From $\mathrm{act}(\mathcal{R})$ and $\mathrm{act}(\tilde{\mathcal{R}})$ we can derive a ternary label

$$\mathrm{AC}(\mathcal{R}, \tilde{\mathcal{R}}) \in \{(1,0,0), (0,1,0), (0,0,1)\} =: \{\text{AC}, \ \text{half-AC}, \ \text{non-AC}\}$$

indicating whether $(\mathcal{R}, \tilde{\mathcal{R}})$ is an AC, half-AC or non-AC, as well as a binary label

$$\mathrm{PD}(\mathcal{R}, \tilde{\mathcal{R}}) \in \{0, 1\} =: \{\mathrm{Right}, \mathrm{Left}\}$$

indicating which of both compounds is more active. Note that

$$\mathrm{AC}(\mathcal{R}, \tilde{\mathcal{R}}) = \mathrm{AC}(\tilde{\mathcal{R}}, \mathcal{R})$$

and

$$\mathrm{PD}(\mathcal{R}, \tilde{\mathcal{R}}) = 1 - \mathrm{PD}(\tilde{\mathcal{R}}, \mathcal{R}).$$

Our goal is to predict both $\mathrm{AC}(\mathcal{R}, \tilde{\mathcal{R}})$ and $\mathrm{PD}(\mathcal{R}, \tilde{\mathcal{R}})$ from the input MMP $(\mathcal{R}, \tilde{\mathcal{R}})$ using a twin neural network model. Note that unlike in Chapter 3 we now also explicitly consider "half-ACs" in our AC-classification task. As stated in Section 3.3.2, half-ACs are defined as MMPs which exhibit a potency difference between one and two orders of magnitude. Until now, half-ACs have been essentially ignored in the AC-prediction literature [104, 105, 111]; however, extending our classification task to also include half-ACs increases the amount of available training data and leads to a more complete and practically relevant (albeit more challenging) *ternary* formulation of the AC-classification problem.

A visual introduction to our developed twin neural network architecture is depicted in Figure 4.1. This illustration might serve as a useful point of reference to facilitate the theoretical discussions in the rest of this section. From a mathematical point of view, our twin network model can be expressed as the composition of two distinct mappings. The first mapping corresponds to a featurisation step that creates a vectorial embedding for each MMP-compound:

$$(\mathcal{R}, \tilde{\mathcal{R}}) \quad \mapsto \quad \begin{bmatrix} T_\theta(\mathcal{R}) \\ T_\theta(\tilde{\mathcal{R}}) \end{bmatrix} \in \mathbb{R}^l \times \mathbb{R}^l. \tag{4.1}$$

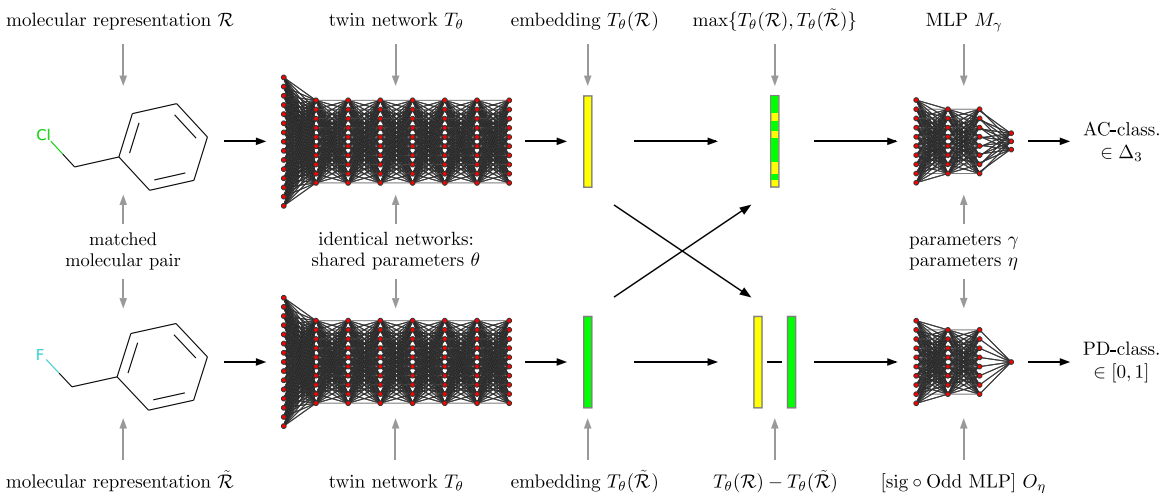

Figure 4.1: Twin neural network model for activity-cliff (AC) and potency-direction (PD) classification. Changing the order of the input compounds leaves the predicted AC-classification label invariant but reverses the predicted PD-classification label.

The second mapping corresponds to a classification step that uses both vectorial embeddings to create probabilistic estimates for $\mathrm{AC}(\mathcal{R}, \tilde{\mathcal{R}})$ and $\mathrm{PD}(\mathcal{R}, \tilde{\mathcal{R}})$:

$$\begin{bmatrix} T_\theta(\mathcal{R}) \\ T_\theta(\tilde{\mathcal{R}}) \end{bmatrix} \quad \mapsto \quad \begin{bmatrix} M_\gamma(\max\{T_\theta(\mathcal{R}), T_\theta(\tilde{\mathcal{R}})\}) \\ O_\eta(T_\theta(\mathcal{R}) - T_\theta(\tilde{\mathcal{R}})) \end{bmatrix} \in \Delta_3^+ \times (0, 1). \tag{4.2}$$

The upper component on the right-hand side of Expression 4.2 corresponds to the predicted AC-classification label,

$$\mathrm{AC}(\mathcal{R}, \tilde{\mathcal{R}}) \approx M_\gamma(\max\{T_\theta(\mathcal{R}), T_\theta(\tilde{\mathcal{R}})\}) =: \hat{\mathrm{AC}}_{\theta,\gamma}(\mathcal{R}, \tilde{\mathcal{R}}) \in \Delta_3^+,$$

and the lower component corresponds to the predicted PD-classification label,

$$\mathrm{PD}(\mathcal{R}, \tilde{\mathcal{R}}) \approx O_\eta(T_\theta(\mathcal{R}) - T_\theta(\tilde{\mathcal{R}})) =: \hat{\mathrm{PD}}_{\theta,\eta}(\mathcal{R}, \tilde{\mathcal{R}}) \in (0, 1).$$

- Here T_θ is a trainable deep-learning based molecular featurisation method and θ is its associated vector of trainable parameters. T_θ is used to map a given molecular representation $\mathcal{R}$ to a feature vector $T_\theta(\mathcal{R}) \in \mathbb{R}^l$ in a differentiable manner. For example, if $\mathcal{R}$ is a molecular graph, then T_θ could take the form of a GIN, and if $\mathcal{R}$ is an ECFP, then T_θ could take the form of a simple MLP. Note that T_θ is applied twice in our model, once to each input compound. T_θ can thus be seen as representing both branches of a twin neural network, as visualised in Figure 4.1 below.

- The vector $\max\{v_1, v_2\} \in \mathbb{R}^l$ describes their componentwise maximum of two arbitrary vectors $v_1, v_2 \in \mathbb{R}^l$. Note that $\max\{\cdot,\cdot\}$ is thus a permutation-invariant set function, i.e. $\max\{v_1, v_2\} = \max\{v_2, v_1\}$. In Remark 4.1 below, we will give some reasons why we specifically chose the max-operator out of all possible permutation-invariant set functions for our model.

- The symbol $\Delta_3^+ := \{(p_1, p_2, p_3) \in \mathbb{R}^3 \mid p_1, p_2, p_3 > 0 \ \wedge \ p_1 + p_2 + p_3 = 1\}$ denotes the set of all positive three-dimensional probability vectors. Δ_3^+ forms a flat triangular surface in $\mathbb{R}^3$ and is sometimes also referred to as the unit 2-simplex.

- $M_\gamma : \mathbb{R}^l \to \Delta_3^+$ is an MLP and γ is its associated vector of trainable parameters. M_γ is of the form
$$M_\gamma = \text{softmax} \circ \bar{M}_\gamma$$
whereby $\bar{M}_\gamma$ is an MLP with three output neurons in its final layer and
$$\text{softmax}(x_1, x_2, x_3) := \left(\frac{e^{x_1}}{e^{x_1} + e^{x_2} + e^{x_3}}, \frac{e^{x_2}}{e^{x_1} + e^{x_2} + e^{x_3}}, \frac{e^{x_3}}{e^{x_1} + e^{x_2} + e^{x_3}}\right) \in \Delta_3^+.$$
M_γ thus outputs a three-dimensional vector whose components represent the predicted probabilites that the MMP $(\mathcal{R}, \tilde{\mathcal{R}})$ is an AC, half-AC or non-AC respectively.

- $O_\eta : \mathbb{R}^l \to (0, 1)$ is an MLP and η is its associated vector of trainable parameters. O_η outputs the predicted probability that $\text{act}(\mathcal{R}) \geq \text{act}(\tilde{\mathcal{R}})$. We restrict O_η to be of the functional form
$$O_\eta(v) = \text{sig} \circ \bar{O}_\eta$$
with
$$\bar{O}_\eta := W_k \circ \arctan \circ W_{k-1} \circ \ \ldots \circ \arctan \circ W_1 \tag{4.3}$$
Here $W_1, ..., W_k$ are trainable weight matrices with no added bias vectors, with W_1 containing l columns and W_k containing 1 row. The expression $\arctan(\cdot)$ is the componentwise applied arctangent map used as a nonlinear activation function. Finally,
$$\text{sig}(x) := \frac{e^x}{e^x + 1} \in [0, 1]$$
is the sigmoidal activation function applied to the single output neuron in the final MLP-layer of $\bar{O}_\eta$. We show below that the above choices make $\bar{O}_\eta$ an *odd function*, i.e. a function with the property that $\bar{O}_\eta(-v) = -\bar{O}_\eta(v)$. We therefore call $\bar{O}_\eta(v)$ an *odd* MLP.

Note that the twin model consists of three neural network components: T_θ, M_γ and O_η. The feature-extracting network component T_θ is separately applied to each of the two input compounds, leading to a twin architecture which can be thought of as containing two copies of T_θ. Both network copies are required to share the same weight parameter vector θ which means that an update of θ during training changes both copies of T_θ in the same manner. We refer to neural networks of this type that contain two or more indistinguishable copies of the same subnetwork as *twin neural networks*. Twin neural networks are natural tools in situations where two or more distinct inputs must be processed by a machine learning system simultaneously [136, 137, 138, 139] since they allow for important pair-based symmetry properties to be hard-coded into the model architecture as an inductive bias. We first have a look at a symmetry property relevant for AC-classification.

Proposition 4.1 (Order-Invariance of AC-classification)**.** *Changing the order of the input compounds from* $(\mathcal{R}, \tilde{\mathcal{R}})$ *to* $(\tilde{\mathcal{R}}, \mathcal{R})$ *in the twin network model from Figure 4.1 leaves the predicted AC-classification label unchanged.*

Proof. The mathematical definition of the twin network model in Expression 4.2 specifies that the predicted AC-classification label for a compound pair $(\mathcal{R}, \tilde{\mathcal{R}})$ is given by

$$M_\gamma(\max\{T_\theta(\mathcal{R}), T_\theta(\tilde{\mathcal{R}})\}) \in \Delta_3^+.$$

Since $\max\{\cdot,\cdot\}$ is a permutation-invariant (i.e. order-independent) set function, one can immediately see that

$$M_\gamma(\max\{T_\theta(\mathcal{R}), T_\theta(\tilde{\mathcal{R}})\}) = M_\gamma(\max\{T_\theta(\tilde{\mathcal{R}}), T_\theta(\mathcal{R})\})$$

which proves the claim. □

Remark 4.1 (Choice of Max-Operator)**.** The proof of Proposition 4.1 is based on the fact that $\max\{\cdot,\cdot\}$ is a permutation-invariant set function. Such functions are also commonly used for global graph pooling in modern GNN architectures where an unordered set of node features must eventually be reduced to a single feature vector. Exchanging $\max\{\cdot,\cdot\}$ with another permutation-invariant set function such as a summation, averaging, sorting, absolute-difference or minimum operator would preserve the order-invariance of the AC-classification. When designing our model we thus experimented with a variety of such operators and ultimately converged on the $\max\{\cdot,\cdot\}$-function as it showed the strongest performance in a set of preliminary

experiments. While this is an empirical finding, an intuitive (albeit speculative) explanation for this might be that ACs are inherently rare outliers; featurisation methods for MMPs that put an emphasis on extreme values such as the $\max\{\cdot,\cdot\}$-function might therefore be more efficient than simple averaging or summation strategies at catching extreme features relevant for outlier-detection.

We now turn our attention to a symmetry property built into our model for PD-classification.

Definition 4.1 (Odd Function)**.** *A real function $f : \mathbb{R}^k \to \mathbb{R}^n$ is called odd if*

$$\forall x \in \mathbb{R}^k : \quad f(-x) = -f(x).$$

Examples of odd functions include the trigonometric function $\sin(x)$ and the family of monomial functions x^{2k+1} for $k \in \mathbb{N}_0$.

Lemma 4.1 (Composition of Odd Functions is Odd)**.** *Let $f : \mathbb{R}^k \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^s$ be two odd functions. Then the composition $h := g \circ f : \mathbb{R}^k \to \mathbb{R}^s$ is also an odd function.*

Proof. Let $x \in \mathbb{R}^k$. Then the oddity of f and g implies that

$$h(-x) = (g \circ f)(-x) = g(f(-x)) = g(-f(x)) = -g(f(x)) = -(g \circ f)(x) = -h(x)$$

which shows that h is odd as well. □

Lemma 4.2 (Oddity of Odd MLP.)**.** *The multilayer perceptron $\bar{O}_\eta : \mathbb{R}^l \to \mathbb{R}^{\bar{l}}$ defined in Equation 4.3 via*

$$\bar{O}_\eta := W_k \circ \arctan \circ W_{k-1} \circ \ \ldots \circ \arctan \circ W_1$$

is an odd function.

Proof. The neural network $\bar{O}_\eta$ is a composition of matrix-multiplication functions

$$W_i : \mathbb{R}^{l_{i-1}} \to \mathbb{R}^{l_i}$$

without added bias vectors and the componentwise-applied trigonometric nonlinear activation function

$$\arctan : \mathbb{R} \to \mathbb{R}.$$

Since Lemma 4.1 establishes that the composition of odd functions is again odd, it is sufficient to show that both W_i and arctan are odd functions.

Since matrix-multiplication is a linear function, it holds in particular that

$$\forall v \in \mathbb{R}^{l_{i-1}} \quad \forall \lambda \in \mathbb{R}: \quad W_i(\lambda v) = \lambda W_i(v).$$

Setting $\lambda := -1$ in the above equation immediately shows that W_i is odd. The oddity of W_i is thus a trivial consequence of its linearity.

To show that arctan is odd, we first observe that its inverse function

$$\tan : (-\pi/2, \pi/2) \to \mathbb{R}$$

is odd since

$$\forall y \in (-\pi/2, \pi/2): \quad \tan(-y) = \frac{\sin(-y)}{\cos(-y)} = \frac{-\sin(y)}{\cos(y)} = -\tan(y).$$

This proof relies on the well-known facts that $\sin(-y) = -\sin(y)$ and $\cos(-y) = \cos(y)$. Now let $x \in \mathbb{R}$ be an arbitrary real number. Since tan is a bijection, there exists exactly one real number $y_x \in (-\pi/2, \pi/2)$ such that $x = \tan(y_x)$ and thus $\arctan(x) = y_x$. We can now exploit the oddity of tan to show that

$$\arctan(-x) = \arctan(-\tan(y_x)) = \arctan(\tan(-y_x)) = -y_x = -\arctan(x)$$

which proves the oddity of arctan.

□

Proposition 4.2 (Order-Equivariance of PD-classification)**.** *Changing the order of the input compounds from $(\mathcal{R}, \tilde{\mathcal{R}})$ to $(\tilde{\mathcal{R}}, \mathcal{R})$ in the twin network model from Figure 4.1 flips the predicted PD-classification label, i.e. transforms it according to the map*

$$(0,1) \ni p \mapsto 1 - p \in (0,1).$$

Proof. The mathematical definition of the twin network model in Expression 4.2 specifies that the predicted PD-classification label for a compound pair $(\mathcal{R}, \tilde{\mathcal{R}})$ is given by

$$O_\eta(T_\theta(\mathcal{R}) - T_\theta(\tilde{\mathcal{R}})) = \mathrm{sig}(\bar{O}_\eta(T_\theta(\mathcal{R}) - T_\theta(\tilde{\mathcal{R}}))) \in (0,1).$$

We start by observing that the Sigmoid activation function fulfills the following functional equation:

$$\mathrm{sig}(x) = \frac{e^x}{e^x + 1} = 1 - \left(1 - \frac{e^x}{e^x + 1}\right) = 1 - \left(\frac{1}{e^x + 1}\right) = 1 - \left(\frac{e^{-x}}{1 + e^{-x}}\right) = 1 - \mathrm{sig}(-x).$$

Based on this relationship and the oddity of $\bar{O}_\eta$ which we showed in Lemma 4.2, we can write

$$
\begin{aligned}
\mathrm{sig}(\bar{O}_\eta(T_\theta(\mathcal{R}) - T_\theta(\tilde{\mathcal{R}}))) &= 1 - \mathrm{sig}(-\bar{O}_\eta(T_\theta(\mathcal{R}) - T_\theta(\tilde{\mathcal{R}}))) \\
&= 1 - \mathrm{sig}(\bar{O}_\eta(-(T_\theta(\mathcal{R}) - T_\theta(\tilde{\mathcal{R}})))) \\
&= 1 - \mathrm{sig}(\bar{O}_\eta(T_\theta(\tilde{\mathcal{R}}) - T_\theta(\mathcal{R})))
\end{aligned}
$$

which proves the claim. □

Propositions 4.1 and 4.2 assure that the twin neural network respects the natural symmetry properties of AC and PD-classification: if we change the order of the input compounds, then the predicted AC-classification label does not change while the predicted PD-classification label flips in the expected manner. Since these properties are hard-coded into the neural architecture of our model, they do not need to be inferred statistically during training.

4.2.2 Loss Function and Model Training

Let

$$\Delta_n \coloneqq \{(p_1, ..., p_n) \in \mathbb{R}^n \mid p_1, ..., p_n \geq 0 \ \wedge \ p_1 + ... + p_n = 1\}$$

denote the set of n-dimensional probability vectors and let

$$\Delta_n^+ \coloneqq \{(p_1, ..., p_n) \in \mathbb{R}^n \mid p_1, ..., p_n > 0 \ \wedge \ p_1 + ... + p_n = 1\}.$$

denote the set of *positive* n-dimensional probability vectors. Then the cross-entropy between two probability vectors is defined as

$$H : \Delta_n \times \Delta_n^+ \to \mathbb{R}, \quad H(p, q) = -\sum_{k=1}^{n} p_i \log(q_i).$$

For a fixed $p_0 \in \Delta_n$, the map

$$q \mapsto H(p_0, q)$$

is minimised if $q \to p_0$; this is one of the reasons why H is canonically used as a loss function for machine-learning-based classification problems. In the binary case with $n = 2$, the definition of H can be simplified to

$$H_{\mathrm{bin}} : [0, 1] \times (0, 1) \to \mathbb{R}, \quad H_{\mathrm{bin}}(p, q) = -p \log(q) - (1 - p) \log(1 - q).$$

Our twin neural network from Figure 4.1 is trained via the following loss function:

$$
\begin{gathered}
\mathcal{L}_{(\mathcal{R},\tilde{\mathcal{R}})}(\theta, \gamma, \eta) \coloneqq \\
w_{\mathrm{AC}}(\mathrm{AC}(\mathcal{R}, \tilde{\mathcal{R}}))H(\mathrm{AC}(\mathcal{R}, \tilde{\mathcal{R}}), \hat{\mathrm{AC}}_{\theta,\gamma}(\mathcal{R}, \tilde{\mathcal{R}})) + \\
w_{\mathrm{PD}} H_{\mathrm{bin}}(\mathrm{PD}(\mathcal{R}, \tilde{\mathcal{R}}), \hat{\mathrm{PD}}_{\theta,\eta}(\mathcal{R}, \tilde{\mathcal{R}})).
\end{gathered}
$$

Here we used a variety of abbreviations:

- $\mathrm{AC}(\mathcal{R}, \tilde{\mathcal{R}}) \in \{(1,0,0),(0,1,0),(0,0,1)\}$ is the AC-classification label.
- $\mathrm{PD}(\mathcal{R}, \tilde{\mathcal{R}}) \in \{0,1\}$ is the PD-classification label.
- $\hat{\mathrm{AC}}_{\theta,\gamma}(\mathcal{R}, \tilde{\mathcal{R}}) \coloneqq M_\gamma(\max\{T_\theta(\mathcal{R}), T_\theta(\tilde{\mathcal{R}})\}) \in \Delta_3^+$ is the predicted AC-classification label.
- $\hat{\mathrm{PD}}_{\theta,\eta}(\mathcal{R}, \tilde{\mathcal{R}}) \coloneqq O_\eta(T_\theta(\mathcal{R}) - T_\theta(\tilde{\mathcal{R}})) \in (0,1)$ is the predicted PD-classification label.

The function

$$w_{\mathrm{AC}} : \{(1,0,0),(0,1,0),(0,0,1)\} \to [0,\infty)$$

is used to place distinct weights on ACs, half-ACs and non-ACs during training. It plays an important role in counteracting the class imbalance in the naturally highly imbalanced task of AC-classification and is an essential part of our loss function. We choose the values of w_{AC} in proportion to the relative frequencies of ACs, half-ACs and non-ACs in the training set to in total give equal weight to each class. This means that if

$$n_{\mathrm{AC}}, n_{\text{half-AC}}, n_{\text{non-AC}} \in \mathbb{N}$$

are the respective numbers of MMPs in the training set that are ACs, half-ACs and non-ACs, then we choose

$$w_{\mathrm{AC}}(1,0,0) = \frac{n_{\text{non-AC}}}{n_{\mathrm{AC}}}, \quad w_{\mathrm{AC}}(0,1,0) = \frac{n_{\text{non-AC}}}{n_{\text{half-AC}}}, \quad w_{\mathrm{AC}}(0,0,1) = 1.$$

The constant $w_{\mathrm{PD}} \in [0,\infty)$ is chosen in relation to the function values of w_{AC} to guarantee that the AC-classification and the (well-balanced) PD-classification task receive on average an equal amount of weight during training. We thus choose w_{PD} to be the expected weight of a training-MMP with respect to AC-classification:

$$w_{\mathrm{PD}} \coloneqq \frac{n_{\mathrm{AC}} w_{\mathrm{AC}}(1,0,0) + n_{\text{half-AC}} w_{\mathrm{AC}}(0,1,0) + n_{\text{non-AC}} w_{\mathrm{AC}}(0,0,1)}{n_{\mathrm{MMP}}}.$$

Here $n_{\mathrm{MMP}} = n_{\mathrm{AC}} + n_{\text{half-AC}} + n_{\text{non-AC}} \in \mathbb{N}$ denotes the total number of MMPs in the training set. Our choice of w_{PD} guarantees that a randomly chosen training-MMP will on average (i.e. in expectation) receive equal weights for AC and PD-classification.

Finally, we show how the symmetry properties of the twin neural network translate into a symmetry property of the loss function $\mathcal{L}_{(\mathcal{R},\tilde{\mathcal{R}})}$ that is highly useful during model training.

Proposition 4.3 (Order-Invariance of Loss Function)**.** *Let* $(\mathcal{R}, \tilde{\mathcal{R}})$ *be an MMP and* $(\tilde{\mathcal{R}}, \mathcal{R})$ *be the same MMP in reversed order. Then it holds for all neural network training parameter configurations* θ, γ, η *that*

$$\mathcal{L}_{(\mathcal{R},\tilde{\mathcal{R}})}(\theta, \gamma, \eta) = \mathcal{L}_{(\tilde{\mathcal{R}},\mathcal{R})}(\theta, \gamma, \eta).$$

Proof. Using the symmetry properties of the twin network with respect to AC and PD-classification shown in Propositions 4.1 and 4.2 along with the identity

$$\begin{aligned} H_{\text{bin}}(p, q) &= -p\log(q) - (1-p)\log(1-q) \\ &= -(1-p)\log(1-q) - (1-(1-p))\log(1-(1-q)) = H_{\text{bin}}(1-p, 1-q), \end{aligned}$$

we can calculate

$$\begin{aligned} &\mathcal{L}_{(\mathcal{R},\tilde{\mathcal{R}})}(\theta, \gamma, \eta) = \\ &\\ &w_{\text{AC}}(\text{AC}(\mathcal{R}, \tilde{\mathcal{R}}))H(\text{AC}(\mathcal{R}, \tilde{\mathcal{R}}), \hat{\text{AC}}_{\theta,\gamma}(\mathcal{R}, \tilde{\mathcal{R}})) + \\ &w_{\text{PD}}H_{\text{bin}}(\text{PD}(\mathcal{R}, \tilde{\mathcal{R}}), \hat{\text{PD}}_{\theta,\eta}(\mathcal{R}, \tilde{\mathcal{R}})) = \\ &\\ &w_{\text{AC}}(\text{AC}(\tilde{\mathcal{R}}, \mathcal{R}))H(\text{AC}(\tilde{\mathcal{R}}, \mathcal{R}), \hat{\text{AC}}_{\theta,\gamma}(\tilde{\mathcal{R}}, \mathcal{R})) + \\ &w_{\text{PD}}H_{\text{bin}}(1 - \text{PD}(\tilde{\mathcal{R}}, \mathcal{R}), 1 - \hat{\text{PD}}_{\theta,\eta}(\tilde{\mathcal{R}}, \mathcal{R})) = \\ &\\ &w_{\text{AC}}(\text{AC}(\tilde{\mathcal{R}}, \mathcal{R}))H(\text{AC}(\tilde{\mathcal{R}}, \mathcal{R}), \hat{\text{AC}}_{\theta,\gamma}(\tilde{\mathcal{R}}, \mathcal{R})) + \\ &w_{\text{PD}}H_{\text{bin}}(\text{PD}(\tilde{\mathcal{R}}, \mathcal{R}), \hat{\text{PD}}_{\theta,\eta}(\tilde{\mathcal{R}}, \mathcal{R})) = \\ &\\ &\mathcal{L}_{(\tilde{\mathcal{R}},\mathcal{R})}(\theta, \gamma, \eta) \end{aligned}$$

which completes the proof. □

Proposition 4.3 guarantees that $\mathcal{L}_{(\mathcal{R},\tilde{\mathcal{R}})}$ is symmetric with respect to the order of the compounds in the input MMP $(\mathcal{R}, \tilde{\mathcal{R}})$. Since the gradients of $\mathcal{L}_{(\mathcal{R},\tilde{\mathcal{R}})}$ with respect to θ, γ and η automatically inherit this symmetry, they too remain unchanged if we flip the order of the input compounds. This property has an important practical consequence: it allows one to only train on one randomly chosen ordering of an input MMP instead of both possible orderings without loss of information.

4.2.3 Molecular Featurisations: Four Model Versions

The exact architecture of T_θ in the twin network depicted in Figure 4.1 hinges upon the molecular representation technique used for the input compounds $(\mathcal{R}, \tilde{\mathcal{R}})$ and the subsequent molecular featurisation method applied to the input representations. We imagine that initially we are given a training space of the form

$$(\mathfrak{D}_{\text{train}}, \mathfrak{M}_{\text{train}})$$

where

$$\mathfrak{D}_{\text{train}} = \{\mathcal{R}_1, \mathcal{R}_2, ...\}$$

represents a data set of individual molecules represented via SMILES strings

$$\{\mathcal{S}_1, \mathcal{S}_2, ...\}$$

and

$$\mathfrak{M}_{\text{train}}^{\text{double}} = \{(\mathcal{R}, \tilde{\mathcal{R}}) \mid (\mathcal{R}, \tilde{\mathcal{R}}) \text{ is MMP and } \mathcal{R}, \tilde{\mathcal{R}} \in \mathfrak{D}_{\text{train}}\}$$

is the set of ordered MMPs that are fully contained in $\mathfrak{D}_{\text{train}}$. Note that by construction $\mathfrak{M}_{\text{train}}^{\text{double}}$ contains both orientations of each MMP. However, Proposition 4.3 guarantees that for our training purposes it is sufficient to only contain one arbitrarily chosen ordering of each MMP. We thus define $\mathfrak{M}_{\text{train}}$ as a proper subset of $\mathfrak{M}_{\text{train}}^{\text{double}}$ of exactly half the size that only contains one arbitrarily chosen ordering of each MMP, i.e. we demand that if $(\mathcal{R}, \tilde{\mathcal{R}}) \in \mathfrak{M}_{\text{train}}$ then $(\tilde{\mathcal{R}}, \mathcal{R}) \notin \mathfrak{M}_{\text{train}}$.

The twin network is always trained only on $\mathfrak{M}_{\text{train}}$, ignoring compunds in $\mathfrak{D}_{\text{train}}$ that are not involved in MMPs, but we developed a transfer learning approach that enables the twin network to nevertheless implicitly exploit extra information encapsulated in $\mathfrak{D}_{\text{train}}$. More specifically, we experimented with four different MMP-representations which subsequently led to four distinct version of our twin model.

- **MMP-representation 1: ECFPs.** Each individual SMILES string in an MMP $(\mathcal{R}, \tilde{\mathcal{R}}) \in \mathfrak{M}_{\text{train}}$ is transformed into a 1024-bit ECFP4 2.5 with active tetrahedral R-S chirality flags using `RDKit` [70]. The featuriser T_θ takes the form of a deep MLP with input dimension 1024.

- **MMP-representation 2: GINs.** Each individual SMILES string in an MMP $(\mathcal{R}, \tilde{\mathcal{R}}) \in \mathfrak{M}_{\text{train}}$ is transformed into a molecular graph using the atom and bond features specified in Table 2.1. The featuriser T_θ takes the form of a GIN-MLP model 2.6.3 with GIN-radius $R = 2$ and GIN-fingerprint length $l = 128$. The GIN-part uses global max pooling in its final graph layer to produce neural fingerprints that feed into the MLP-part.

- **MMP-representation 3 (supervised): ECFP-NFPs.** Each SMILES string in $\mathfrak{D}_{\text{train}}$ is transformed into a 1024-bit ECFP4 2.5 with active tetrahedral R-S chirality flags using `RDKit` [70]. Then an ECFP-MLP model Q with hidden width 1024 is trained on $\mathfrak{D}_{\text{train}}$ as a supervised QSAR model to predict the activities of individual compounds. After training, the final layer of Q that maps vectors from a 1024-dimensional learned feature space onto scalar activity predictions is removed to obtain a feature extractor Q_{feat}. We refer to the 1024-dimensional feature vectors generated by Q_{feat} for individual compounds as ECFP-neural-fingerprints (ECFP-NFPs). Q_{feat} is finally used to map MMPs $(\mathcal{R}, \tilde{\mathcal{R}}) \in \mathfrak{M}_{\text{train}}$ to pairs of ECFP-NFPs on which the twin network is subsequently trained. The featuriser T_θ then takes the form of a deep MLP with input dimension 1024.

- **MMP-representation 4 (supervised): GIN-NFPs.** Each SMILES string in $\mathfrak{D}_{\text{train}}$ is transformed into a molecular graph using the atom and bond features specified in Table 2.1. Then a GIN-MLP model Q with GIN-radius $R = 2$, GIN-fingerprint length $l = 128$, and hidden MLP-width 256 is trained on $\mathfrak{D}_{\text{train}}$ as a supervised QSAR model to predict the activities of individual compounds. After training, the final layer of Q that maps vectors from a 256-dimensional learned feature space onto scalar activity predictions is removed to obtain a feature extractor Q_{feat}. We refer to the 256-dimensional feature vectors generated by Q_{feat} for individual compounds as GIN-neural-fingerprints (GIN-NFPs). Q_{feat} is finally used to map MMPs $(\mathcal{R}, \tilde{\mathcal{R}}) \in \mathfrak{M}_{\text{train}}$ to pairs of GIN-NFPs on which the twin network is subsequently trained. The featuriser T_θ then takes the form of a deep MLP with input dimension 256.

Consider the set of individual training compounds involved in MMPs:

$$\mathcal{D}_{\text{train}}^{\text{MMP}} := \{\mathcal{R} \in \mathfrak{D}_{\text{train}} \mid \exists\, \tilde{\mathcal{R}} \in \mathfrak{D}_{\text{train}} : (\tilde{\mathcal{R}}, \mathcal{R}) \in \mathfrak{M}_{\text{train}} \text{ or } (\mathcal{R}, \tilde{\mathcal{R}}) \in \mathfrak{M}_{\text{train}}\}.$$

All four introduced MMP-representations use the training signal encapsulated in in $\mathcal{D}_{\text{train}}^{\text{MMP}}$. However, the transfer-learning-based representations ECFP-NFP and GIN-NFP go further: they also allow us to implicitly leverage the information contained in the set of isolated compounds $\mathfrak{D}_{\text{train}} \setminus \mathcal{D}_{\text{train}}^{\text{MMP}}$ since the feature extractor Q_{feat} must have encountered these compounds during its own preliminary training process on the full training space $\mathfrak{D}_{\text{train}}$. Knowledge about the isolated compounds in $\mathfrak{D}_{\text{train}} \setminus \mathcal{D}_{\text{train}}^{\text{MMP}}$ and their experimentally measured activity labels is thus implicitly encoded in the trained parameters of Q_{feat} and transferred to the features it extracts.

4.3 Computational Experiments

In this section, we present a series of computational experiments to investigate the AC and PD-classification capabilities of the four versions of our twin neural network model discussed in Section 4.2.3. We further compare the twin models to the two strongest QSAR-modelling baselines for AC-classification found in our previous study in Chapter 3: the combinations ECFP-MLP and GIN-MLP.

4.3.1 Experimental Methodology

4.3.1.1 Molecular Data Set

For our experiments we employed the SARS-CoV-2 main protease data set introduced in Section 3.3.1 since it is composed of a single high-quality assay and has a high density of MMPs. Note that this is the exact same data set that we used for our computational study on QSAR models for AC-prediction in Chapter 3. The protein structure of SARS-CoV-2 main protease is visualised in Figure 3.4.

The data was obtained and cleaned in the manner described in Section 3.3.1 and takes the form of SMILES strings with associated IC_{50} [µM] values. An overview of the numbers of compounds, MMPs, ACs, half-ACs and non-ACs in the data set is given in Table 3.1. SARS-CoV-2 main protease is one of the key enzymes in the viral replication cycle of the SARS coronavirus 2 which recently led to the global COVID-19 pandemic. It is a promising target for antiviral drugs against this coronavirus [120].

4.3.1.2 Data Splitting Technique and Prediction Tasks

For data splitting into training and test sets, we employed the novel technique for pair-based data that we developed for our previous computational study in Section 3.3.3. It is visualised in Figure 3.5 and delivers data splits of the form

$$\mathfrak{S}^{i,j} = (\mathfrak{D}^{i,j}_{\text{train}}, \mathfrak{D}^{i,j}_{\text{test}}, \mathfrak{M}^{i,j}_{\text{train}}, \mathfrak{M}^{i,j}_{\text{test}}, \mathfrak{M}^{i,j}_{\text{inter}}, \mathfrak{M}^{\text{i,j}}_{\text{cores}})$$

for $i \in \{1, ..., m\}$ and $j \in \{1, ..., k\}$. Here the pair $(\mathfrak{D}^{i,j}_{\text{train}}, \mathfrak{D}^{i,j}_{\text{test}})$ represents the j-th random split with the i-th random seed of the underlying SARS-CoV-2 main protease data set $\mathfrak{D}$ in a k-fold cross validation scheme repeated with m random seeds. The MMP-sets $\mathfrak{M}^{i,j}_{\text{train}}, \mathfrak{M}^{i,j}_{\text{test}}, \mathfrak{M}^{i,j}_{\text{inter}}, \mathfrak{M}^{\text{i,j}}_{\text{cores}}$ differ via the relationship of their associated MMPs with the individual compounds in $\mathfrak{D}^{i,j}_{\text{train}}$ and $\mathfrak{D}^{i,j}_{\text{test}}$. These sets are rigorously defined in Section 3.3.3 and visualised in Figure 3.5. We will shortly repeat their definitions here in intuitive terms:

- $\mathfrak{M}^{i,j}_{\text{train}}$ contains MMPs that are fully included in $\mathfrak{D}^{i,j}_{\text{train}}$.
- $\mathfrak{M}^{i,j}_{\text{inter}}$ contains MMPs with exactly one compound in $\mathfrak{D}^{i,j}_{\text{train}}$ and the other compound in $\mathfrak{D}^{i,j}_{\text{test}}$. It simulates a compound-optimisation scenario where one is searching for small modifications of known compounds that would give rise to ACs or half-ACs.
- $\mathfrak{M}^{i,j}_{\text{test}}$ contains MMPs that are fully included in $\mathfrak{D}^{i,j}_{\text{test}}$. It models a setting where one tries to discover novel ACs and half-ACs in the same area of chemical space that $\mathfrak{D}^{i,j}_{\text{train}}$ was sampled from.
- Finally, $\mathfrak{M}^{i,j}_{\text{cores}}$ is a subset of $\mathfrak{M}^{i,j}_{\text{test}}$ consisting of MMPs in $\mathfrak{D}^{i,j}_{\text{test}}$ that do not share structural cores with MMPs in $\mathfrak{M}^{i,j}_{\text{inter}}$ or $\mathfrak{M}^{i,j}_{\text{train}}$. It corresponds to the difficult task of predicting ACs and half-ACs within structurally novel MMPs that do not contain near analogs to MMP-compounds involved in the training set.

As mentioned above, Proposition 4.3 assures us that it is possible without loss of generality to always only consider one randomly chosen compound-ordering for each MMP in all MMP-sets. The overall AC and PD-classification performance of each model is recorded via the average over mk training and test runs for all data splits $\mathfrak{S}^{1,1}, ..., \mathfrak{S}^{m,k}$. For our experiments we set $(m, k) = (50, 2)$. The number of $50*2 = 100$ repetitions is substantial and considerably large for deep-learning experiments due to their associated computational cost. This choice led to a runtime in the order of approximately 10-20 hours per model; however, it significantly reduced the effects of stochastic fluctations on our results and led to increased experimental quality and reliability.

4.3.1.3 Prediction Tasks and Prediction Strategies

As specified in Section 4.2, each MMP

$$(\mathcal{R}, \tilde{\mathcal{R}}) \in \mathfrak{M}^{i,j}_{\text{train}} \cup \mathfrak{M}^{i,j}_{\text{test}} \cup \mathfrak{M}^{i,j}_{\text{inter}} \cup \mathfrak{M}^{i,j}_{\text{cores}}$$

comes with a ternary label

$$\text{AC}(\mathcal{R}, \tilde{\mathcal{R}}) \in \{(1,0,0), (0,1,0), (0,0,1)\} =: \{\text{AC},\ \text{half-AC},\ \text{non-AC}\}$$

indicating whether $(\mathcal{R}, \tilde{\mathcal{R}})$ is an AC, half-AC or non-AC; and a binary label

$$\text{PD}(\mathcal{R}, \tilde{\mathcal{R}}) \in \{0, 1\} =: \{\text{Right}, \text{Left}\}$$

indicating which of both compounds is more active. The goal of each model is to predict both $\mathrm{AC}(\mathcal{R}, \tilde{\mathcal{R}})$ and $\mathrm{PD}(\mathcal{R}, \tilde{\mathcal{R}})$ from the input MMP-representation $(\mathcal{R}, \tilde{\mathcal{R}})$. The four twin neural network models are designed to output explicit probabilistic estimates of $\mathrm{AC}(\mathcal{R}, \tilde{\mathcal{R}})$ in Δ_3 and of $\mathrm{PD}(\mathcal{R}, \tilde{\mathcal{R}})$ in $(0, 1)$, and can therefore be directly used for AC and PD-classification. The two QSAR-modelling baselines ECFP-MLP and GIN-MLP, however, can only directly output estimates of the activity labels of individual molecules. Thus, when dealing with a QSAR model Q, we thresholded the predicted MMP-activity-difference,

$$Q(\mathcal{R}) - Q(\tilde{\mathcal{R}}) \approx \mathrm{act}(\mathcal{R}) - \mathrm{act}(\tilde{\mathcal{R}}),$$

to construct discrete predictions for $\mathrm{AC}(\mathcal{R}, \tilde{\mathcal{R}})$ and $\mathrm{PD}(\mathcal{R}, \tilde{\mathcal{R}})$. This was done in the same straightforward manner as described in Section 3.3.4 for our previous computational study, with the only exception that we extended our AC-classification stategy to now also include half-ACs. If $(\mathcal{R}, \tilde{\mathcal{R}}) \in \mathfrak{M}^{i,j}_{\mathrm{train}} \cup \mathfrak{M}^{i,j}_{\mathrm{test}} \cup \mathfrak{M}^{i,j}_{\mathrm{cores}}$ then we performed AC-classification via

$$(\mathcal{R}, \tilde{\mathcal{R}}) \mapsto \begin{cases} (0,0,1) & \text{if } |Q(\mathcal{R}) - Q(\tilde{\mathcal{R}})| \leq 1, \\ (0,1,0) & \text{if } |Q(\mathcal{R}) - Q(\tilde{\mathcal{R}})| \in (1,2), \\ (1,0,0) & \text{if } |Q(\mathcal{R}) - Q(\tilde{\mathcal{R}})| \geq 2 \end{cases}$$

and PD-classification via

$$(\mathcal{R}, \tilde{\mathcal{R}}) \mapsto \begin{cases} 1 & \text{if } Q(\mathcal{R}) \geq Q(\tilde{\mathcal{R}}), \\ 0 & \text{else.} \end{cases}$$

Similarly, if $(\mathcal{R}, \tilde{\mathcal{R}}) \in \mathfrak{M}^{i,j}_{\mathrm{inter}}$ and we were *a priori* given (say) the experimental activity $\mathrm{act}(\tilde{\mathcal{R}})$, then we performed AC-classification via

$$(\mathcal{R}, \tilde{\mathcal{R}}) \mapsto \begin{cases} (0,0,1) & \text{if } |Q(\mathcal{R}) - \mathrm{act}(\tilde{\mathcal{R}})| \leq 1, \\ (0,1,0) & \text{if } |Q(\mathcal{R}) - \mathrm{act}(\tilde{\mathcal{R}})| \in (1,2), \\ (1,0,0) & \text{if } |Q(\mathcal{R}) - \mathrm{act}(\tilde{\mathcal{R}})| \geq 2 \end{cases}$$

and PD-classification via

$$(\mathcal{R}, \tilde{\mathcal{R}}) \mapsto \begin{cases} 1 & \text{if } Q(\mathcal{R}) \geq \mathrm{act}(\tilde{\mathcal{R}}), \\ 0 & \text{else.} \end{cases} .$$

4.3.1.4 Performance Measures

For the balanced PD-classification problem we employ the standard accuracy as a suitable performance measure:

$$\frac{\text{number of correct predictions}}{\text{number of predictions}} \in [0, 1].$$

For each MMP-set and each model, we measured PD-classification accuracy (1) on the whole MMP-set, (2) on the subset of MMPs predicted by the model to be half-ACs, and (3) on the subset of MMPs predicted by the model to be ACs.

The ternary AC-classification task is naturally highly imbalanced and it is therefore necessary to choose a more nuanced set of performance measures in order to paint an adequate and detailed picture of model performance. For each class

$$C \in \{\text{AC},\ \text{half-AC},\ \text{non-AC}\},$$

let n_C be the number of MMPs in class C, p_C be the number of MMPs predicted to be in class C, and p_C^{true} be the number of MMPs correctly predicted to be in class C. In our experiments, we recorded the *sensitivity*

$$\frac{p_C^{\text{true}}}{n_C} \in [0,1]$$

and the *precision*

$$\frac{p_C^{\text{true}}}{p_C} \in [0,1]$$

of each model for each class C. There is an implicit trade-off between sensitivity and precision; improvement in one of both metrics usually leads to deterioration of the other. A model that shows both higher precision and sensitivity than another model for a class C can be seen as a better classifier for this particular class.

Finally, as a simple overall performance measure for ternary AC-classification we employed the multi-class version of the Matthews correlation coefficient (MCC). Let $n_{\text{MMP}} := n_{\text{AC}} + n_{\text{half-AC}} + n_{\text{non-AC}}$ be the total number of MMPs and $p^{\text{true}} = p_{\text{AC}}^{\text{true}} + p_{\text{half-AC}}^{\text{true}} + p_{\text{non-AC}}^{\text{true}}$ be the total number of correct predictions. Then the multi-class MCC is given by

$$\frac{n_{\text{MMP}} p^{\text{true}} - \sum\limits_{C \in \{\text{AC, half-AC, non-AC}\}} n_C p_C}{\sqrt{\left(n_{\text{MMP}}^2 - \sum\limits_{C \in \{\text{AC, half-AC, non-AC}\}} p_C^2\right)\left(n_{\text{MMP}}^2 - \sum\limits_{C \in \{\text{AC, half-AC, non-AC}\}} n_C^2\right)}} \in [-1,1].$$

While the MCC can give a quick and rough assessement of AC-classification performance, it should be used with caution: A lot of subtle differences are lost entirely when boiling down the performance of a highly imbalanced multi-class prediction model into a single scalar performance metric such as the MCC. For an AC-classifier, one must therefore also take a close look at its individual sensitivity and precision values for each class in order to get accurate insights into its true performance and utility.

4.3.1.5 Evaluated Models

We included six distinct models in our computational experiments:

- **ECFP + MLP Baseline:** A standard ECFP-MLP model trained on individual molecules for QSAR-prediction, as was used in the computational study from Chapter 3.
- **GIN + MLP Baseline:** A standard GIN-MLP model trained on individual molecules for QSAR-prediction, as was used in the computational study from Chapter 3.
- **ECFPs + Twin Network:** A twin neural network as visualised in Figure 4.1 that uses ECFPs for the featurisation of individual MMP-compounds.
- **GINs + Twin Network:** A twin neural network as visualised in Figure 4.1 that uses GINs for the featurisation of individual MMP-compounds.
- **ECFP-NFPs + Twin Network:** A twin neural network as visualised in Figure 4.1 that uses pre-trained ECFP-NFPs for the featurisation of individual MMP-compounds.
- **GIN-NFPs + Twin Network:** A twin neural network as visualised in Figure 4.1 that uses pre-trained GIN-NFPs for the featurisation of individual MMP-compounds.

Each of the four twin neural network models above corresponds to one of the four MMP-featurisations described in Section 4.2.3. ECFPs, MLPs and GINs were implemented in `RDKit` [70], `PyTorch` [131], and `PyTorch Geometric` [96], respectively.

4.3.1.6 Model Training and Hyperparameter Settings

As mentioned, each model was evaluated within a k-fold cross validation scheme repeated with m random seeds for $(m, k) = (50, 2)$. This means that an independent version of each model was trained on each training space $(\mathfrak{D}^{i,j}_{\text{train}}, \mathfrak{M}^{i,j}_{\text{train}})$ for $i \in \{1, ..., 50\}$ and $j \in \{1, 2\}$ and the results were then averaged over all $mk = 50 * 2 = 100$ trials. All models were trained on a single NVIDIA GeForce RTX 3060 GPU using AdamW optimisation [134]. QSAR models were trained via the mean squared error loss function and twin neural networks via the custom cross-entropy-based loss function $\mathcal{L}_{(\mathcal{R},\tilde{\mathcal{R}})}$ introduced in Section 4.2.2.

Table 4.1: Training and model hyperparameters of twin neural networks and baseline QSAR methods.

ECFP + MLP **Architecture:** arch(MLP) $= (1024, 1, 1024, 5)$ **Training:** batch size $= 64$, learning rate $= 10^{-3}$, learning rate decay $= \max\{0.98^{\text{epoch}}, 10^{-1}\}$, weight decay $= 0.1$, dropout rate $= 0.25$, epochs $= 500$
ECFPs + Twin Network **Architecture:** arch$(T_\theta) = (1024, 1024, 1024, 3)$, arch$(M_\gamma) = (1024, 3, 1024, 1)$, arch$(O_\eta) = (1024, 1, 1024, 1)$ **Training:** batch size $= 2048$, learning rate $= 10^{-4}$, learning rate decay $= \max\{0.98^{\text{epoch}}, 10^{-2}\}$, weight decay $= 0.1$, dropout rate $= 0.25$, epochs $= 500$
ECFP-NFPs + Twin Network **Architecture:** arch$(T_\theta) = (1024, 1024, 1024, 3)$, arch$(M_\gamma) = (1024, 3, 1024, 1)$, arch$(O_\eta) = (1024, 1, 1024, 1)$ **Training:** batch size $= 2048$, learning rate $= 10^{-4}$, learning rate decay $= \max\{0.98^{\text{epoch}}, 10^{-2}\}$, weight decay $= 0.1$, dropout rate $= 0.25$, epochs $= 500$
GIN + MLP **Architecture:** arch(GIN) $= \{R = 2,\ l = 128,\ \text{arch}(\phi_1) = (78, 128, 128, 2),\ \text{arch}(\phi_2) = (128, 128, 128, 2)\}$, arch(MLP) $= (128, 1, 256, 3)$ **Training:** batch size $= 256$, learning rate $= 10^{-2}$, learning rate decay $= \max\{0.98^{\text{epoch}}, 10^{-1}\}$, weight decay $= 0.1$, dropout rate $= 0.25$, epochs $= 1000$
GINs + Twin Network **Architecture:** arch$(T_\theta^{\text{GIN-part}}) = \{R = 2,\ l = 128,\ \text{arch}(\phi_1) = (78, 128, 128, 2),\ \text{arch}(\phi_2) = (128, 128, 128, 2)\}$, arch$(T_\theta^{\text{MLP-part}}) = (128, 256, 256, 3)$, arch$(M_\gamma) = (256, 3, 256, 1)$, arch$(O_\eta) = (256, 1, 256, 1)$ **Training:** batch size $= 2048$, learning rate $= 10^{-3}$, learning rate decay $=$ none, weight decay $= 0.1$, dropout rate $= 0.25$, epochs $= 1500$
GIN-NFPs + Twin Network **Architecture:** arch$(T_\theta) = (256, 256, 256, 3)$, arch$(M_\gamma) = (256, 3, 256, 1)$, arch$(O_\eta) = (256, 1, 256, 1)$ **Training:** batch size $= 2048$, learning rate $= 10^{-4}$, learning rate decay $=$ none, weight decay $= 0.1$, dropout rate $= 0.25$, epochs $= 500$

A detailed specification of the hyperparameter choices for the evaluated models and their training loops can be found in Table 4.1. The architecture of MLPs is specified via quadruples of integers in $\mathbb{N}^4$ which specify input dimension, output dimension, hidden dimension and number of hidden layers; for example, the quadruple $(100, 1, 200, 5)$ describes an MLP with 100 neurons in its input layer, followed by 5 hidden layers with 200 neurons each, followed by an output layer with 1 neuron. The architecture of GINs is specified by their radius R, their fingerprint length l and the architectures of the MLPs ϕ_r at each graph layer r (see Section 2.6.3).

All neural networks consistently used $\mathrm{ReLU}(x) := \max\{0, x\}$ as their hidden activation function with the exception of O_η from Figure 4.1 which was equipped with arctan-activations to preserve the desired symmetry-properties of the twin model (see Proposition 4.2). Furthermore, all neural networks employed trainable additive bias vectors after each weight-matrix multiplication. Batch normalisation [133] was used in all models. The training and model hyperparameter settings of a QSAR method Q for NFP-generation (see Section 4.2.3) were chosen to be almost identical to the hyperparameter settings of its corresponding baseline QSAR method specified in Table 4.1, with one difference being that batch-normalisation was dropped between the last hidden layer and the scalar output layer when using Q to learn NFPs via QSAR-prediction. These architectural choices led to a dimensionality of 1024 for ECFP-NFPs and of 256 for GIN-NFPs.

Due to time constraints and the complexity of the twin neural network architecture, a full hyperparameter optimisation of all models was not feasible in this project. However, our hyperparameter choices for the baseline models ECFP-MLP and GIN-MLP generated strong QSAR-prediction results that were on par with the results achieved by the corresponding fully hyperparameter-optimised models from Chapter 3. The ECFP-MLP model in this section reached a mean QSAR-MAE on $\mathfrak{D}_{\text{test}}$ of 0.426 (in pIC_{50} units) and the GIN-MLP model reached a mean QSAR-MAE of 0.442. These results are as strong as the ones depicted in Figure 3.9 which are also based on a 2-fold cross validation scheme on the same SARS-CoV-2 data set, but included extensive hyperparameter-optimisation routines for the ECFP-MLP and the GIN-MLP model. We can thus conclude with a high degree of confidence that whenever a twin network beats a baseline QSAR model in the experiments in this chapter, then the same twin network would beat the same QSAR model in a related experiment involving full hyperparamter-optimisation of all models. This is because the results in Figure 3.9 imply that hyperparameter optimisation would not improve the performance of the QSAR-modelling baselines in the experiments in this section; it

could only possibly improve the performance of a twin network. Even without hyperparameter optimisation, our experiments are therefore still suitable to rigorously answer the question whether a twin neural network can beat a (hyperparameter-optimised) ECFP-MLP or GIN-MLP-baseline at AC or PD-classification.

4.3.2 Results and Discussion

The results of our computational experiments are depicted in Figures 4.2 and 4.3. Note that the empirical observations in this chapter are based exclusively on the SARS-CoV-2 main protease binding affinity data set. This data set was suitable for our experiments since it is composed of a single high-quality assay and has a high density of MMPs. However, we acknowledge the necessity to repeat our analysis with other data sets to obtain further evidence for the generalisability of our results.

Remark 4.2 (Error Bars)**.** When considering the plots in Figures 4.2 and 4.3, it is easy to overinterpret overlapping error bars when comparing models. The purpose of an error bar in Figure 4.2 or Figure 4.3 is to communicate the standard deviation associated with the performance of a particular model across multiple random data splits to the reader. In this study, the length of each error bar was set to two standard deviations. While this choice is not uncommon, it is essentially arbitrary. For example, setting the error bar length to one standard deviation instead and pointing this out to the reader would remove many overlaps between error bars while conveying the same information about the stochastic fluctuations in model performance across random data splits.

4.3.2.1 AC-Classification Performance

Looking at overall AC-classification performance as quantified by the MCC in the last row of Figure 4.2 shows that twin networks that use standard ECFPs or GINs as featurisers appear to exhibit slightly weaker overall MCC-performance than their related baseline QSAR models, with the exception of ECFP-based twin networks on $\mathfrak{M}_{\text{test}}$ which clearly outperform ECFP-MLPs in this scenario. The reason might be that these twin networks are exclusively trained on MMPs and ignore compounds not associated with any MMP; unlike QSAR models, such twin networks can thus not leverage the additional SAR-information in isolated compounds in $\mathfrak{D}_{\text{test}}$. This substantially decreases the size of their training set relative to QSAR models that can train on all available compounds. A simple way to remove this data advantage for QSAR models would be to also training them exclusively on compounds involved

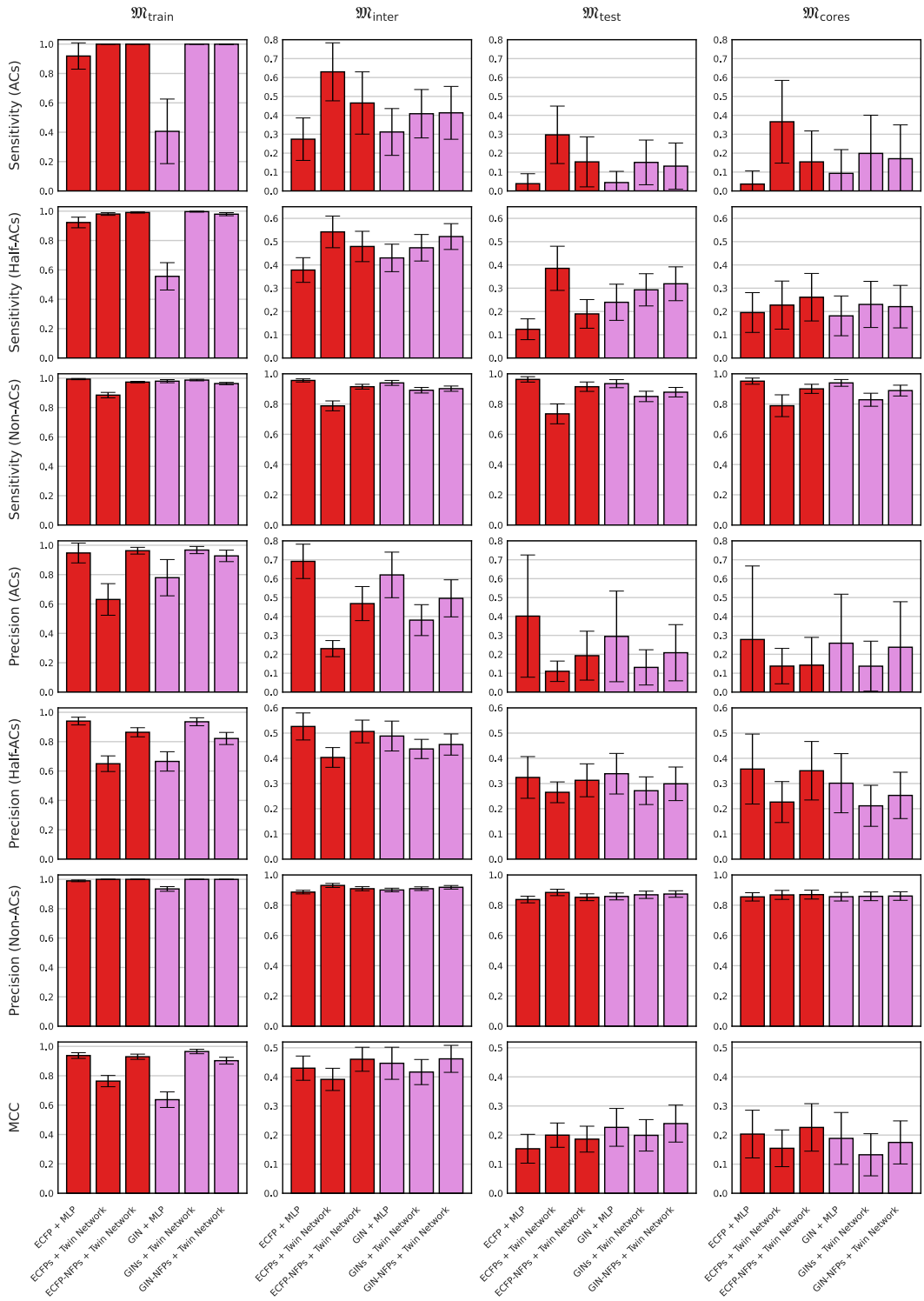

Figure 4.2: Activity-cliff (AC) classification results on the SARS-CoV-2 main protease data set for two baseline QSAR models and four newly developed twin neural network models with distinct input featurisations. The red and violet bars correspond to ECFP-based and GIN-based models, respectively. The total length of each error bar is set to be equal to twice the standard deviation of the performance measured over all $mk = 50 * 2 = 100$ models.

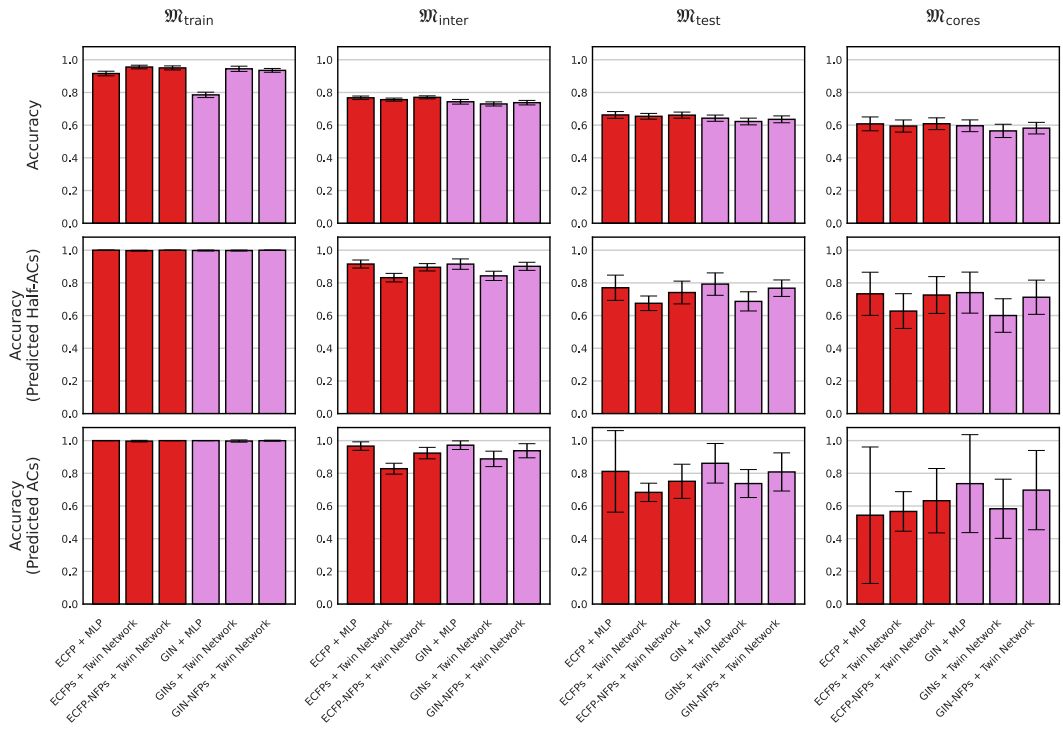

Figure 4.3: Potency-direction (PD) classification results on the SARS-CoV-2 main protease data set for two baseline QSAR models and four newly developed twin neural network models with distinct input featurisations. The red and violet bars correspond to ECFP-based and GIN-based models, respectively. The total length of each error bar is set to be equal to twice the standard deviation of the performance measured over all $mk = 50 * 2 = 100$ models.

in MMPs; however, such experiments would not truly simulate a realistic scenario where one uses all available data to produce the strongest possible model.

The overall picture changes when the input featurisations for the twin networks are enriched via transfer learning. The strongest results on $\mathfrak{M}_{\text{inter}}$ are achieved by ECFP-NFP-based and GIN-NFP-based twin networks. Similarly, the strongest results on $\mathfrak{M}_{\text{test}}$ are achieved by GIN-NFP-based twin networks and the strongest results on $\mathfrak{M}_{\text{cores}}$ are achieved by ECFP-NFP-based twin networks. Our computational study in Chapter 3 and the results in Figure 3.9 indicate that ECFP-MLPs and GIN-MLPs are already amongst the strongest QSAR models for AC-classification. Our observations nevertheless suggest that twin networks when combined with a suitable transfer-learning approach for MMP-featurisation (i.e. NFPs) outcompete even such strong baseline QSAR models for the detection of AC across multiple distinct prediction scenarios.

It is notable though that if the only examined metric is the overall MCC, then the advantage of the NFP-based twin networks over the QSAR-modelling baselines can in some cases appear modest. However, taking a closer look at the AC-sensitivity

and AC-precision of the QSAR-modelling baselines reveals a more nuanced picture. ECFP-MLPs and GIN-MLPs exhibit a reasonably high MCC across MMP-sets which should in theory reflect good overall performance for AC-classification. However, the AC-sensitivity of both QSAR models is exceedingly low, reaching around 0.04 on $\mathfrak{M}_{\text{test}}$; in other words, around 96% of ACs are not correctly classified by these two techniques. This weakness is to some extent compensated by comparatively high AC-precision values: for example, if an ECFP-MLP classifies an MMP in $\mathfrak{M}_{\text{test}}$ as an AC then the probability that this MMP truly is an AC is after all approximately 40%. However, the low AC-sensitivity of the tested QSAR models still casts doubt on their practical utility as AC-classifiers, in spite of their moderately high MCCs. Tuning the thresholds for AC-classification in the QSAR-modelling-based prediction strategies outlined in Section 4.3.1.3 could mitigate this problem by increasing AC-sensitivity at the cost of AC-precision, thus potentially leading to a more well-balanced and useful classifier. However, the originally set classification thresholds in Section 4.3.1.3 precisely reflect the conceptual definitions of non-ACs, half-ACs and ACs used in the literature and throughout this project. Changing these thresholds would thus invite paradoxical situations where for example MMPs with a predicted absolute activity difference of less than two orders of magnitude could be classified as ACs which by definition are MMPs that exhibit an absolute activity difference of more than two orders of magnitude. Therefore, tuning the classification thresholds for QSAR models in an attempt to balance out their AC-sensitivity and AC-precision, while potentially feasible in practice, would arguably be inconsistent and undesirable from a conceptual point of view.

The problem of imbalanced AC-sensitivity and AC-precision can be circumvented in an elegant manner by the twin neural network methodology. The function w_{AC} that forms part of the twin network loss function $\mathcal{L}_{(\mathcal{R},\tilde{\mathcal{R}})}$ introduced in Section 4.2.2 allows one to directly control the class weights given to non-ACs, half-ACs and ACs respectively during twin network training. Assigning a higher relative weight to one class leads to a higher sensitivity (and usually lower precision) for this class in the final trained model. In the twin network model it is thus possible to tune the trade-off between sensitivity and precision for each class in a straightforward and conceptually consistent manner by simply modifying the weight function w_{AC}. In our experiments, we automatically chose class weights that reflected the relative frequencies of non-ACs, half-ACs and ACs in the SARS-CoV-2 data set (see Table 3.1). This generally leads to a much less skewed trade-off between AC-sensitivity and AC-precision than can be observed for the two baseline QSAR models. For instance, ECFP-NFP-based

twin networks show the strongest overall MCC-performance of 0.461 out of all models on $\mathfrak{M}_{\text{inter}}$ while exhibiting an AC-sensitivity of 0.465 and an AC-precision of 0.468. In comparison, ECFP-MLPs on $\mathfrak{M}_{\text{inter}}$ reach a lower MCC of 0.430, a much lower AC-sensitivity of 0.274 and a much higher AC-precision of 0.692. In summary, the twin networks tend to achieve much higher AC-sensitivity and somewhat higher half-AC-sensitivity compared to ECFP-MLPs or GIN-MLPs, at the cost of lower precision in both classes, leading to more balanced classifiers. This important trend cannot be seen from the MCC-performance alone; it makes twin neural network models substantially more well-rounded AC-classifiers than ECFP-MLPs or GIN-MLPs which skew heavily in the direction of high AC-precision and very low AC-sensitivity.

Perhaps unsurprisingly, almost all of the evaluated models exhibit very high AC-classification performance on $\mathfrak{M}_{\text{train}}$ as they were trained on this set of MMPs. There seems to be one salient exception to this though: GIN-MLPs exhibit comparatively very low overall AC-classification performance on the set of training-MMPs, reaching an MCC of only 0.637 and an AC-sensitivity of only 0.406. This effect might simply be caused by insufficient overall QSAR-prediction performance and/or a lack of training time on the QSAR task. However, there are two reasons that speak against this explanation: Firstly, each GIN-MLP was trained for no less than 1000 epochs on $\mathfrak{D}_{\text{train}}$ which was sufficiently long for the training loss to converge to a stable plateau for the last several hundred epochs in all instances. Secondly, GIN-MLPs reach a similar QSAR-MAE for the prediction of individual molecular activities on $\mathfrak{D}_{\text{test}}$ as ECFP-MLPs (0.442 for GIN-MLPs vs. 0.426 for ECFP-MLPs), yet ECFP-MLPs show much higher AC-classification performance on $\mathfrak{M}_{\text{train}}$. In a future research project, it might be interesting to experiment with techniques to increase the AC-classification performance of GIN-MLP models on $\mathfrak{M}_{\text{train}}$ and study the effects of this on the performance on $\mathfrak{M}_{\text{inter}}$, $\mathfrak{M}_{\text{test}}$ and $\mathfrak{M}_{\text{cores}}$. A straightforward avenue to explore would be to increase the size of GIN-MLPs and train them for an unusually long time, potentially on the order of 10^4 to 10^5 epochs; while this would almost certainly lead to overfitting on the QSAR task and thus lower QSAR-prediction performance, one can hypothesise that it might nevertheless increase AC-sensitivity on $\mathfrak{M}_{\text{train}}$.

When analysing the AC-classification results with respect to molecular featurisation, we see that GIN-based methods consistently match or outperform ECFP-based methods according to MCCs on $\mathfrak{M}_{\text{inter}}$ and $\mathfrak{M}_{\text{test}}$. These findings agree with the trends observed in Figure 3.9 from our previous computational study and extend them to the realm of twin neural networks. At first glance, our observations thus once again suggest that graph-based GIN-features tend to be equal or superior to ECFPs for the

detection of ACs, and this seems to be true both when the underlying predictor is an MLP or when it is a twin neural network. There are three caveats to this conclusion though.

Firstly, looking closely at sensitivity and precision values reveals that the MCC might be a misleading performance measure in some cases. For example, GIN-based twin networks reach an MCC of 0.199 on $\mathfrak{M}_{\text{test}}$ while ECFP-based twin networks reach an essentially equivalent MCC of 0.200 on the same MMP-set. This suggests that both methods are equally good at ternary AC-classification. And indeed, the precision values of both methods on $\mathfrak{M}_{\text{test}}$ closely resemble each other for all three classes. However, the sensitivities of GIN-based twin networks for non-ACs/half-ACs/ACs are 0.850 / 0.293 / 0.151 while the corresponding sensitivity of ECFP-based twin networks are 0.735 / 0.385 / 0.297. Thus, since ACs and half-ACs are naturally of greater interest than non-ACs in almost all cases, ECFP-based twin networks seem to be strongly preferable to GIN-based twin networks, even though their respective MCCs suggest that both methods are equivalent. One way to express this advantage is that ECFP-based twin networks can generate a longer list of potential AC and half-AC-candidates from a given test data set than GIN-based twin networks while operating at an equal level of precision.

The second caveat to the idea that GIN-features are superior to ECFPs for AC-classification comes from the fact that this trend, while true on $\mathfrak{M}_{\text{inter}}$ and $\mathfrak{M}_{\text{test}}$, appears to reverse on $\mathfrak{M}_{\text{cores}}$. The MCC-performance of all GIN-based methods drops in a predictable manner when moving from $\mathfrak{M}_{\text{test}}$ to the more difficult test set $\mathfrak{M}_{\text{cores}}$. However, surprisingly, the same is not true for ECFP-based methods: here the MCC-performance of ECFP-MLPs and ECFP-NFP-based twin networks does in fact increase. ECFPs therefore appear to cope better than GINs with the distributional shift between $\mathfrak{M}_{\text{train}}$ and $\mathfrak{M}_{\text{cores}}$; at first sight, this suggest that ECFP-based methods might be able to extract more generalisable chemical knowledge that is not merely based on memorisation. Once again the picture becomes more refined though when looking at sensitivity and precision metrics on $\mathfrak{M}_{\text{cores}}$. The investigated models can be grouped into two sets according to their respective AC-precision values: ECFP-MLPs, GIN-MLPs and GIN-NFP-based twin networks share a similar level of AC-precision; and the same is true for ECFP-based twin networks, ECFP-NFP-based twin networks and GIN-based twin networks. The highest AC-sensitivity by far in the first group is exhibited by GIN-NFP-based twin networks and the same is true in the second group for ECFP-based twin networks. Thus, if the goal is to discover as many AC-candidates as possible in a new data set while maintaining a certain level

of precision, then the best model is either a GIN-NFP-based or an ECFP-based twin network, even though neither of these two methods exhibits a comparatively high MCC.

4.3.2.2 PD-classification Performance

The overall accuracy-results for the balanced binary PD-classification task are depicted in the first row of Figure 4.3. We can see that the differences in performance between distinct models on $\mathfrak{M}_{\text{inter}}$, $\mathfrak{M}_{\text{test}}$ and $\mathfrak{M}_{\text{cores}}$ are minor. All models correctly classify the potency direction of approximately 75% of MMPs in $\mathfrak{M}_{\text{inter}}$, 65% of MMPs in $\mathfrak{M}_{\text{test}}$ and 60% of MMPs in $\mathfrak{M}_{\text{cores}}$. Interestingly, GIN-MLPs perform noticeably worse than the other methods on $\mathfrak{M}_{\text{train}}$ but this does not seem to harm their relative predictive abilities on $\mathfrak{M}_{\text{inter}}$, $\mathfrak{M}_{\text{test}}$ or $\mathfrak{M}_{\text{cores}}$. This resembles the observation already discussed in Section 4.3.2.1 that GIN-MLPs show much lower AC-sensitivity on $\mathfrak{M}_{\text{train}}$ than the other models while still exhibiting competitive AC-classification performance on the other MMP-sets. Our previous discussion on this effect thus also applies to the analogous situation for PD-classification.

The accuracy-results for all models on the full MMP-sets $\mathfrak{M}_{\text{inter}}$, $\mathfrak{M}_{\text{test}}$ and $\mathfrak{M}_{\text{cores}}$ might appear modest at first; note however that MMPs are by definition pairs of *structurally similar* molecules and that this similarity regularly goes hand-in-hand with similar activities for a given pharmacological target. ACs and half-ACs are relatively rare exceptions to this rule which is known by medicinal chemists as the *similarity principle*. Classifying which of two compounds is more active thus becomes a challenging prediction task in a setting where both compounds have almost identical chemical structures and therefore also frequently exhibit almost identical activities. In particular, the similarity principle along with the inherent noisiness of experimentally measured binding affinity values suggests that the discrete PD-classification learning signal derived from $\mathfrak{M}_{\text{train}}$ for the four twin networks likely contains a considerable amount of noise in the form of false binary labels. In light of these obstacles, the capabilities of the investigated methods to detect the potency direction of MMPs is nontrivial. The fact that all six distinct techniques reach almost equivalent levels of accuracy on $\mathfrak{M}_{\text{inter}}$, $\mathfrak{M}_{\text{test}}$ and $\mathfrak{M}_{\text{cores}}$ might potentially be caused by a performance ceiling that is rooted in the underlying data set itself and that is approached by all models.

The second and third row of Figure 4.3 contain PD-classification accuracies in a scenario where the sets of test-MMPs are restricted to only include MMPs that were classified as half-ACs/ACs by a respective model. Arguably, these results are of

higher practical relevance than the overall PD-classification results on the full MMP-sets, since the predicted potency direction of an MMP is often of stronger interest if the predicted activity difference is substantial. This is for instance true in the case of compound optimisation. We overall see stronger PD-classification results on the restricted MMP-sets of predicted half-ACs/ACs than on the full MMPs-sets; perhaps unsurprisingly, this suggests that it is easier to predict the potency direction of MMPs whose underlying activity difference is large. When MMPs are restricted to predicted half-ACs/ACs, we observe essentially perfect results for all models on $\mathfrak{M}_{\text{train}}$. We further see a tendency for the two baseline QSAR models to slightly outperform the four twin neural networks on the restricted versions of $\mathfrak{M}_{\text{inter}}$, $\mathfrak{M}_{\text{test}}$ and $\mathfrak{M}_{\text{cores}}$. For example, on $\mathfrak{M}_{\text{inter}}$ ECFP-MLPs predict close to 100% of potency directions of predicted ACs correctly, while ECFP-based twin networks accurately predict only slightly more than 80%.

Caution needs to be exercised though when directly comparing accuracy-results like these, since the exact sets of predicted half-ACs/ACs differ from model to model. Moreover, the lower the half-AC/AC-precision of a model, the more non-ACs infiltrate its set of predicted half-ACs/ACs and the harder PD-classification subsequently becomes on this set. And indeed, comparing Figures 4.2 and 4.3 reveals that models that exhibit a comparatively high half-AC/AC-precision (such as the two baseline QSAR models) also tend to exhibit a comparatively high PD-classification accuracy on predicted half-ACs/ACs. As mentioned before, a likely underlying reason for this is that the potency direction of half-ACs and ACs might be easier to predict than the potency direction of non-ACs.

Some models also tend to have a substantially larger set of prediced half-ACs/ACs than others. The number of (not necessarily correctly) predicted members of a class C for a given model M_1 can be estimated from the true number of occurrences of C in the test set and the sensitivity and precision values of the model for C:

$$p_C(M_1) = n_C \frac{\text{sens}_C(M_1)}{\text{prec}_C(M_1)}.$$

Here we use the notation introduced in Section 4.3.1.4. If M_2 is a second model, then it follows that

$$\frac{p_C(M_1)}{p_C(M_2)} = \frac{\text{sens}_C(M_1)\text{prec}_C(M_2)}{\text{prec}_C(M_1)\text{sens}_C(M_2)}.$$

With this formula, we can for instance conclude that the (average) ratio of the numbers of predicted ACs for $M_1 :=$ ECFP-based twin networks and $M_2 :=$ GIN-based

twin networks on $\mathfrak{M}_{\text{test}}$ is given by

$$\frac{p_{\text{AC}}(M_1)}{p_{\text{AC}}(M_2)} = \frac{\text{sens}_{\text{AC}}(M_1)\text{prec}_{\text{AC}}(M_2)}{\text{prec}_{\text{AC}}(M_1)\text{sens}_{\text{AC}}(M_2)} = \frac{0.297 * 0.131}{0.110 * 0.151} \approx 2.342.$$

This implies that the list of predicted AC-candidates generated by ECFP-based twin networks is more than twice as long as the equivalent list generated by GIN-based twin networks, even though both models operate at a similar level of AC-precision (0.110 vs. 0.131). However, the PD-classification accuracies of ECFP-based and GIN-based twin networks show a modest but noticeable difference on $\mathfrak{M}_{\text{test}}$ (0.683 vs. 0.737). ECFP-based twin networks can thus detect a much larger volume of potential AC-candidates in $\mathfrak{M}_{\text{test}}$ than GIN-based twin networks at an almost equivalent level of precision, but this advantage comes at the apparent cost of a slightly reduced PD-classification accuracy on predicted ACs.

4.4 Conclusions

In this chapter, we have introduced a novel twin neural network architecture for activity-cliff (AC) and potency-direction (PD) classification. The general design of this twin network enables it to seamlessly integrate with essentially any featurisation method for individual molecules. We first mathematically investigated the built-in symmetry properties of the twin architecture which were designed to serve as a useful inductive bias for our application. We then conducted a series of computational experiments on a data set of SARS-CoV-2 main protease inhibitors to compare the AC and PD-classification performance of two QSAR models and four versions of our twin model, based on either ECFPs or GINs, with or without pre-trained neural fingerprints (NFPs) generated via supervised transfer learning. To the best of our knowledge, this work represents the first application of twin neural networks to the problems of AC and PD-classification and the first AC-prediction study that includes appropriate QSAR-modelling baselines for comparison.

At PD-classification, the baseline QSAR models tend to be slightly more accurate than the twin networks if the MMP-sets are restricted to predicted half-ACs/ACs. However, on the full MMP-sets the differences in overall PD-classification accuracy amongst the six investigated models are negligible. This stands in contrast to the heterogeneous AC-classification results, where we have seen that the developed twin architecture is able to outperform both baseline QSAR models in a variety of important ways. NFP-based twin networks tend to reach the strongest overall MCC-performance across distinct prediction scenarios. This demonstrates the positive ef-

fects of the transfer learning technique used to generate NFPs which enables the exploitation of additional chemical knowledge contained in compounds not involved in MMPs. Our observations suggest that NFP-based twin networks are a well-balanced choice for AC-classification that can outperform even strong baseline QSAR models. Moreover, twin networks generally appear to strike a substantially better balance between sensitivity and precision than the overly conservative QSAR methods. This may make them more attractive than standard QSAR models for applications such as compound optimisation and automatic SAR-knowledge acquisition. The improved AC-classification balance for the twin network is rooted in the fact that class weights can be assigned to non-ACs, half-ACs and ACs in its loss function.

Our observations have further demonstrate that a detailed look at sensitivity and precision metrics in addition to overall performance measures such as the MCC can lead to important insights in certain use cases. In particular, if the goal is to generate a long list of potential half-AC/AC-candidates at a certain level of precision, then sometimes one model can be preferable over another one even if both have essentially the same MCC. For instance, ECFP-based twin networks and GIN-based twin networks have very similar MCCs and AC-precisions on $\mathfrak{M}_{\text{test}}$ in our experiments; however, the AC-sensitivity of ECFP-based twin networks on this MMP-set is much higher and they can thus generate a much longer list of AC-candidates at the same level of precision.

A promising pathway for future research might be to explore ways to further optimise the input-MMP featurisation for AC-classification, potentially by including three-dimensional information that can smooth out two-dimensional MMP-cliffs, or by pre-training a neural feature extractor via a contrastive loss that explicitly incentivises the resolution of ACs in the embedding space.

Chapter 5

Beyond Hashing: Substructure-Pooling Techniques to Robustly Improve Extended-Connectivity Fingerprints

5.1 Introduction

The use of extended-connectivity fingerprints (ECFPs) is omnipresent in current cheminformatics. As described in Section 2.5, the ECFP-algorithm transforms a molecule (usually given in the form of a SMILES string) into a high-dimensional binary vector whose components indicate the presence or absence of particular circular chemical substructures within the input compound.[1] A modern and widely recognised technical description of ECFPs was given in 2010 by Rogers and Hahn [16], although the key ideas underlying ECFP-generation were already introduced by Morgan [73] in 1965. To name only a few applications, ECFPs have been used successfully for ligand-based virtual screening [78], the prediction of the aqueous solubility of molecular compounds [20], the computational detection of cytotoxic substructures of molecules [79], the prediction of the inhibitory activity of molecules against *E. coli* enzymes [80], the identification of unknown binding targets of chemical compounds via similarity searching [81], and the prediction of quantum-chemical properties of small molecules [17].

Perhaps the most typical use case of ECFPs is as a featurisation method for supervised molecular machine learning, i.e. as a method to transform molecules into binary feature vectors for a given downstream molecular property prediction task. For this purpose, ECFPs are popular tools due to their conceptual simplicity, high

[1] *ECFPs with counts* also exist, which do not simply take the form of binary vectors but integer vectors that indicate the exact number of occurrences of each substructure. In this work, however, unless specifically stated otherwise, we always focus on binary ECFPs without counts due to their more widespread use and conceptual simplicity.

interpretability, and low computational cost. Moreover, a growing corpus of literature suggests that ECFPs still regularly match or even surpass the predictive performance of trainable feature-extraction methods based on state-of-the-art message-passing graph neural networks (GNNs) [6, 7, 10, 11, 13, 45]. These findings agree with our own results presented in Chapter 3 where we demonstrated via a series of rigorous computational experiments that ECFPs consistently beat modern GINs [29] at binding affinity prediction for a given protein (i.e. QSAR-prediction).

From a bird's eye view, the ECFP-algorithm outlined in Section 2.5.3 can be decomposed into two steps: a first step

$$\mathcal{S} \mapsto \mathcal{I} \subseteq \{1, ..., 2^{32}\}$$

in which the SMILES string $\mathcal{S}$ of a compound is mapped to a set of integer identifiers $\mathcal{I}$ which correspond to circular chemical subgraphs of varying radii $r \in \{0, ..., R\}$ with atomic centers in $\mathcal{S}$ (see Figure 2.2); and a second step

$$\mathcal{I} \mapsto \mathcal{F} \in \{0, 1\}^l$$

in which the set of integer identifiers (i.e. the set if circular subgraphs) is transformed into a binary vector of predefined length l. We refer to the first step as *substructure enumeration* and to the second step as *substructure pooling*. In this chapter, we focus on substructure pooling and we will give a precise mathematical definition of it in the context of supervised molecular machine learning.

Note that the technical parallels between ECFPs and message-passing GNNs (Section 2.6) are striking: In both cases, a compound is first transformed into an unordered set-representation whereby different compounds can be transformed to sets of different cardinalities. For ECFPs, this set-representation is given by the set of initial and updated integer atom identifiers in $\mathcal{I}$ (which correspond to circular substructures) while for GNNs this set-representation is given by the set of initial and updated atom feature vectors.[2] For both methods, the pooling operation then plays the crucial role of reducing the given set-representation to a single feature vector that describes the entire compound and that can readily be fed into a standard machine learning model.

While considerable work has been done to develop and investigate a variety of pooling methods for modern GNN-architectures [31, 152, 153, 154, 155], almost no

[2]To be precise, one uses multisets (i.e. sets with counts) instead of standard sets in the case of GNN-architectures. This is to be able to distinguish identical atom feature vectors belonging to distinct atoms. Similarly, one would use multisets instead of sets when dealing with ECFPs with counts instead of binary ECFPs. However, as mentioned above, in this work we focus on binary ECFPs without counts.

analogous research exists that describes and explores alternative substructure pooling methods for ECFPs. The canonical way for substructure pooling [16] for ECFPs is based on the use of a deterministic hash function as was already described in Section 2.5 of this thesis. The hash function is used to map a set of integer identifiers $\mathcal{I}$ into a set $\{1, ..., l\}$ of much smaller range that can then be transformed into a binary vectorial fingerprint $\mathcal{F} \in \{0, 1\}^l$ of desired length l. This straightforward type of hashing is the current default substructure-pooling technique for ECFPs and is used almost universally in the molecular property prediction literature, although alternative hashing procedures for ECFPs and closely related circular fingerprints have been explored by Probst and Reymond [156] for analog searches in big data settings. In spite of its widespread use, the default hashing technique comes with a considerable downside: It is well-known that standard hash-based substructure pooling for ECFPs suffers from the technical problem of *bit collisions*, which occur when distinct integer identifiers (i.e. distinct substructures) are hashed to the same component of the output vector $\mathcal{F}$. Bit collisions necessarily occur when the fingerprint dimension l is smaller than the number of detected circular substructures which is almost always the case in standard settings; moreover, the smaller the fingerprint dimension l relative to the number of detected substructures, the more bit collisions emerge. The ambiguities introduced by bit collisions into the fingerprint not only compromise its interpretability but also its predictive performance in machine-learning applications.

Gütlein and Kramer [157] published a high-quality study which represents one of the few existing works that systematically explores alternative substructure pooling strategies for ECFPs to circumvent the problem of bit collisions. They explore an advanced supervised substructure selection scheme (i.e. a feature selection scheme) as a pooling method to construct what they referred to as *filtered* fingerprints. We will discuss filtered fingerprints as introduced by Gütlein and Kramer [157] in more detail below.

In this chapter, we describe an extremely simple and surprisingly effective alternative to hashing for substructure-pooling of ECFP-substructures which we call *Sort & Slice*. In a nutshell, Sort & Slice is based on first sorting all unique circular substructures in the training set according to their frequency of occurrence within training compounds and then slicing away the least frequent substructures to arrive at a fingerprint of desired length. From a formal point of view, Sort & Slice can be seen as a very simple unsupervised feature selection scheme. In spite of this simplicity, we are able to mathematically show a strong overlap between Sort & Slice and a

more complex unsupervised feature selection method based on the maximisation of information entropy [158, 159].

Sort & Slice fully removes bit collisions at the level of integer identifiers; each vectorial component in the final fingerprint corresponds to the presence or absence of one and only one integer identifier in $\mathcal{I}$. This implies that each fingerprint component is associated with a unique circular substructure (if one ignores the slim chance of hash collisions during the ECFP-substructure enumeration that could lead to two substructures being assigned the same integer identifier). As a result, vectorial ECFP-representations generated via Sort & Slice are more straightforward to interpret than hashed ECFPs, which regularly contain components that correspond to multiple integer identifiers due to colliding bits. Furthermore, note that the slicing procedure, while massively reducing the dimension of the fingerprint, tends to preserve the vast majority of the chemical information contained in the training set. This is because in common real-world molecular data sets, the least frequent substructures usually only appear in a few compounds (very often only in a single compound); removing such substructures thus corresponds to the removal of almost-constant features with little-to-no information that could be leveraged by a prediction algorithm.

We show via a series of rigorous ECFP-based computational experiments that Sort & Slice regularly and sometimes substantially outperforms (i) standard hash-based substructure pooling [16], (ii) filtered fingerprints as developed by Gütlein and Kramer [157], and (iii) a popular supervised feature selection scheme based on mutual-information maximisation (MIM) [158, 159]. In particular, we show that Sort & Slice consistently leads to higher predictive performance than hashing, filtering, or MIM

- across a diverse set of supervised molecular property prediction tasks involving regression as well as balanced and imbalanced classification,
- across distributional shifts caused by distinct data splitting strategies (random, scaffold),
- across machine learning models (random forest, multilayer perceptron),
- across fingerprint radii ($R \in \{1, 2, 3\}$),
- across fingerprint lengths ($l \in \{512, 1024, 2048, 4096\}$), and
- across initial atomic invariants (standard, pharmacophoric).

We are thus able to demonstrate that the predictive advantage provided by Sort & Slice over both hashing and two advanced supervised feature selection techniques is highly robust and generalises across a large number of settings.

Note that we do not dare to claim that the Sort & Slice technique we investigate in this chapter was necessarily first discovered and implemented by us; in fact, the simplicity of the method makes it possible that versions of it have already been applied by other researchers in the past in a variety of contexts. In particular, we acknowledge the recent work of MacDougall [48] which we discovered during our literature search: he proposed a procedure that is almost identical to Sort & Slice, with the only technical difference to our method appearing to be associated with the slicing procedure. While the slicing technique from MacDougall [48] only allows limited control over the length of the fingerprint, our slicing scheme can generate fingerprints of any predefined arbitrary length. We will discuss this difference in more detail below when mathematically describing Sort & Slice. Our goal is to provide the following novel contributions in this chapter:

1. We give a precise and very general mathematical definition of substructure pooling and suggest it as a potential research avenue to boost the performance of structural fingerprints in molecular machine learning.

2. We mathematically describe our version of Sort & Slice as a straightforward alternative to hashing for substructure pooling that is very easy to implement, allows full control over the fingerprint length and exhibits markedly higher interpretability due to an absence of bit collisions.

3. We show via a series of strict computational experiments that for ECFPs Sort & Slice consistently leads to higher predictive performance than hashing and two relevant supervised feature selection schemes for molecular property prediction across a large number of scenarios; and that frequently the performance gains associated with Sort & Slice are surprisingly large.

4. We recommend that due to its technical simplicity, dimensional customisability, improved interpretability and superior predictive performance, Sort & Slice should canonically replace hashing as the default substructure pooling method to vectorise ECFPs for supervised molecular machine learning.

5.2 Methods and Experimental Methodology

´

5.2.1 Substructure Pooling: Mathematical Description

Definition 5.1 (Substructure Pooling)**.** *Let*

$$\mathfrak{C} = \{\mathcal{C}_1, ..., \mathcal{C}_m\}$$

be a (potentially very large) set of chemical substructures. The substructures $\mathcal{C}_1, ..., \mathcal{C}_m$ *could take the form of SMILES strings, molecular graphs, or another computational representation such as hashed integer identifiers. Now let the power set of* $\mathfrak{C}$*, i.e. the set of all possible subsets of* $\mathfrak{C}$*, be denoted by*

$$P(\mathfrak{C}) := \{\mathfrak{A} \mid \mathfrak{A} \subseteq \mathfrak{C}\}.$$

A substructure-pooling method of dimension $l \in \mathbb{N}$ *for the chemical substructures in* $\mathfrak{C}$ *is an operator*

$$\Psi : P(\mathfrak{C}) \to \mathbb{R}^l$$

that maps subsets of $\mathfrak{C}$ *to* l*-dimensional real-valued vectors.*

Substructure pooling naturally appears in the context of supervised molecular machine learning for the vectorisation of structural fingerprints for molecular featurisation. To see this, consider a supervised molecular property prediction task specified by a training set of n compounds

$$\mathfrak{D} = \{\mathcal{R}_1, ..., \mathcal{R}_n\} \subset \mathfrak{R}$$

and an associated function

$$c : \mathfrak{D} \to \mathbb{R}$$

that assigns regression or classification labels to the training set. The training compounds $\mathcal{R}_1, ..., \mathcal{R}_n$ form part of a larger chemical space $\mathfrak{R}$ whose elements could for example be represented via SMILES strings or molecular graphs. Analogous to Definition 5.1, let

$$\mathfrak{C} = \{\mathcal{C}_1, ..., \mathcal{C}_m\}$$

be a set of m chemical substructures of interest and let

$$P(\mathfrak{C}) = \{\mathfrak{A} \mid \mathfrak{A} \subseteq \mathfrak{C}\}$$

be its power set. Now let

$$\varphi : \mathfrak{R} \to P(\mathfrak{C})$$

be a structural-fingerprinting algorithm that maps an input compound in $\mathfrak{R}$ to the set of substructures in $\mathfrak{C}$ that appear in the input compound. Via φ one can transform each training compound $\mathcal{R}_i$ into a set-representation consisting of r_i substructures:

$$\varphi(\mathcal{R}_i) = \{\mathcal{C}_{i,1}, ..., \mathcal{C}_{i,r_i}\} \subseteq \mathfrak{C}.$$

The elements $\mathcal{C}_{i,1}, ..., \mathcal{C}_{i,r_i} \in \mathfrak{C}$ corresponds to chemical substructures that belong to the larger set of considered substructures $\mathfrak{C}$ and that are present in $\mathcal{R}_i$.

A straightforward example for φ is of course the ECFP-algorithm described in Section 2.5. In this case, $\mathfrak{C}$ is the set of all circular chemical fragments up to a predefined radius R represented via their respective integer identifiers in the hash space $\{1, ..., 2^{32}\}$. Another option for φ that has been frequently used in the past is the 166-bit MACCS fingerprint [74] for which $\mathfrak{C}$ becomes a fixed set of 166 chemical substructures represented via their respective SMARTS strings [160].

By composing a substructure-fingerprinting algorithm

$$\varphi : \mathfrak{R} \to P(\mathfrak{C})$$

with a substructure-pooling operator

$$\Psi : P(\mathfrak{C}) \to \mathbb{R}^l$$

one can finally transform each molecular set-representation $\varphi(\mathcal{R}_i)$ into a real-valued vector $\Psi(\varphi(\mathcal{R}_i)) \in \mathbb{R}^l$. The vectorised training data set

$$\Psi(\varphi(\mathfrak{D})) \subset \mathbb{R}^l$$

can then be fed into a standard machine learning model such as a random forest or a multilayer perceptron that can be trained to predict the labels specified by c. Note that the substructure-pooling operator Ψ can be constructed leveraging knowledge from the training data; in other words, Ψ is allowed to depend on $\mathfrak{D}$ and the labelling function c. Furthermore, Ψ does not necessarily need to be a fixed function; it could also be a trainable deep network. For instance, later in Section 6.2 we will introduce a trainable substructure-pooling technique based on a differentiable self-attention mechanism.

The problem of substructure pooling can be directly translated into a mathematical problem that closely resembles GNN pooling as described in Section 2.6. This can be achieved if one employs a substructure-embedding function of some dimension w:

$$\gamma : \mathfrak{C} \to \mathbb{R}^w.$$

Using this embedding, one can straightforwardly transform sets of substructures into sets of vectors via

$$\Gamma_\gamma : P(\mathfrak{C}) \to \{A \subset \mathbb{R}^w \mid A \text{ is finite}\}, \quad \Gamma_\gamma(\{\mathcal{C}_1, ..., \mathcal{C}_r\}) = \{\gamma(\mathcal{C}_1), ..., \gamma(\mathcal{C}_r)\}.$$

Given a substructure-fingerprinting algorithm φ, the composite function

$$\Gamma_\gamma \circ \varphi : \mathfrak{R} \to \{A \subset \mathbb{R}^w \mid A \text{ is finite}\}$$

then maps a chemical compound to a set of real-valued vectors. This vectorial set-representation can be seen analogously to the outputs of an iteratively applied GNN-layer which computes a representation of the molecule in the form of layerwise sets $f_i(A)$ of initial and updated atom feature vectors

$$f_0(A), ..., f_R(A) \subset \{A \subset \mathbb{R}^w \mid A \text{ is finite}\}.$$

Whether the vectors within a set-representation generated by $\Gamma_\gamma \circ \varphi$ can also be associated with specific radii and central atoms within the input compound like the feature vectors in $f_0(A), ..., f_R(A)$ depends on the specifics of the fingerprinting-algorithm φ. In the case of ECFPs, for example, such a layerwise and atomwise interpretation that is equivalent to its GNN-counterpart is possible and highly natural since each ECFP-generated integer identifier is associated with a substructure of radius $r \in \{0, ..., R\}$ around a central atom. Note though that even if such an interpretation is not possible, both techniques still lead to a representation of the input compound in terms of sets of vectors. We can thus see that all GNN-pooling methods that correspond to graph-topology-independent permutation-invariant set-functions

$$\bigoplus : \{A \subset \mathbb{R}^w \mid A \text{ is finite}\} \to \mathbb{R}^l$$

can immediately be repurposed to vectorise the sets produced by $\Gamma_\gamma \circ \varphi$ for downstream machine learning. Given a suitable embedding γ one can therefore reuse techniques from the literature on GNN pooling for substructure pooling. As an example, consider the sum operator that is frequently used for GNN pooling:

$$\sum : \{A \subset \mathbb{R}^w \mid A \text{ is finite}\} \to \mathbb{R}^w, \quad \sum(\{v_1, ..., v_r\}) = \sum_{i=1}^{r} v_i.$$

By composing the sum operator Σ with a substructure embedding γ we immediately gain a substructure-pooling operator whose output dimension l is equal to the embedding-dimension w:

$$\Psi : P(\mathfrak{C}) \to \mathbb{R}^w, \quad \Psi(\{\mathcal{C}_1, ..., \mathcal{C}_r\}) = \Big(\sum \circ \Gamma_\gamma\Big)(\{\mathcal{C}_1, ..., \mathcal{C}_r\}) = \sum_{i=1}^{r} \gamma(\mathcal{C}_i).$$

From this example it becomes clear that a pair $(\oplus, \gamma)$ consisting of a permutation-invariant set-function $\oplus$ and a substructure-embedding function γ is sufficient to fully determine an associated substructure-pooling method $\Psi = \oplus \circ \Gamma_\gamma$. Note that just like in the case of GNN pooling, the operator $\oplus$ could correspond to simple summation or averaging, or could be modelled by a more complex trainable deep network as is the case for the differentiable graph pooling method proposed by Navarin et al. [31].

We now take a look at two natural example techniques for the embedding of substructures.

Example 5.1 (One-Hot Embedding)**.** Perhaps the simplest possible substructure embedding is given via *one-hot encoding*. Denote with $u_{m,i} \in \mathbb{R}^m$ the m-dimensional unit vector which is equal to 1 only in its i-th component and equal to 0 everywhere else. Furthermore, let

$$s : \mathfrak{C} \to \{1, ..., m\}$$

be a bijective function. Note that s imposes a linear order on $\mathfrak{C}$ by assigning a unique rank $s(\mathcal{C}) \in \{1, ..., m\}$ to each substructure $\mathcal{C} \in \mathfrak{C}$. Then the one-hot encoded substructure embedding associated with s is simply given by

$$\gamma_s : \mathfrak{C} \to \mathbb{R}^m, \quad \gamma_s(\mathcal{C}) = u_{m,s(\mathcal{C})}.$$

We see that in the case of one-hot encoding, the embedding dimension w is equal to the number of substructures $|\mathfrak{C}| = m$ and can thus be extremely large.

Example 5.2 (Physicochemical Embedding)**.** Another natural way to embed substructures is given via physicochemical-descriptor vectors (PDVs) as described in Section 2.4. For example, the 200 descriptors specified in Table 2.2 can be computed for each substructure, leading to a meaningful embedding function

$$\gamma_{\text{PDV}} : \mathfrak{C} \to \mathbb{R}^{200}.$$

Some advantages of this embedding over one-hot encoding are its high *a priori* content of interpretable and potentially useful chemical information and the fact that similar substructures are likely to end up close in the embedding space. A potential disadvantage of physicochemical embeddings over one-hot embeddings might be that certain frequently-used permutation-invariant set functions such as the summation or averaging operator potentially incur a considerable loss of information when applied to sets of PDVs. In contrast, for sets of one-hot encoded vectors, summation and averaging are invertible operations that do not incur any loss of information. To see

this, note that every positive entry in a vector that represents the sum or average of a set of one-hot encoded vectors corresponds to exactly one particular one-hot encoded vector in the original set.[3]

5.2.2 Investigated Substructure-Pooling Techniques

Note that substructure pooling as given in Definition 5.1 is a highly general operation that encompasses hashing, feature selection and more complex methods based on combining substructure-embeddings γ with differentiable permutation-invariant set functions $\oplus$. In spite of this, substructure-pooling beyond hashing remains largely unexplored. Below we go on to describe four substructure-pooling methods for ECFPs that we chose to experimentally investigate: the canonically used hashing procedure, the Sort & Slice method which is the main focus of this study, and two technically mature supervised feature selection schemes. A high-level overview of the four evaluated substructure-pooling techniques can be found in Figure 5.1.

The substructure-pooling methods we describe in this section can in principle be used with any structural fingerprint; however, in this study we focus on ECFPs. We therefore assume from now on that the set of chemical substructures under consideration $\mathfrak{C}$ consists of *circular* substructures up to a predefined radius R represented via a set of integer identifiers in a large hash space:

$$\mathfrak{C} = \{\mathcal{J}_1, ..., \mathcal{J}_m\} \subseteq \{1, ..., 2^{32}\}.$$

Remark 5.1 (Potential Hash Collisions during ECFP-Substructure Enumeration)**.** Note that strictly speaking there is not always a perfect one-to-one correspondence between circular chemical substructures and integer ECFP-identifiers in the hash space $\{1, ..., 2^{32}\}$; theoretically hash collisions could occur during the ECFP-substructure enumeration process (within one compound or across distinct compounds) that could lead to two circular fragments being assigned the same integer identifier $\mathcal{J}_i$. However, hash collisions of this type are very rare [16, 157]. For simplicity we therefore assume that each integer identifier $\mathcal{J}_i \in \mathfrak{C}$ can indeed always be mapped to a unique circular chemical fragment.

[3]In the case of ECFPs with counts, detected substructures in a compound would be encoded as multisets of one-hot encoded vectors instead of sets. Summing a multiset of one-hot encoded vectors is still invertible, but averaging is not. For instance, the multiset $\{(1,0),(1,0),(0,1),(0,1)\}_{\text{mul}}$ can easily be uniquely reconstructed from its sum $(2,2)$, but not from its average $(1/2,1/2)$, since the multiset $\{(1,0),(0,1)\}_{\text{mul}}$ corresponds to the same average.

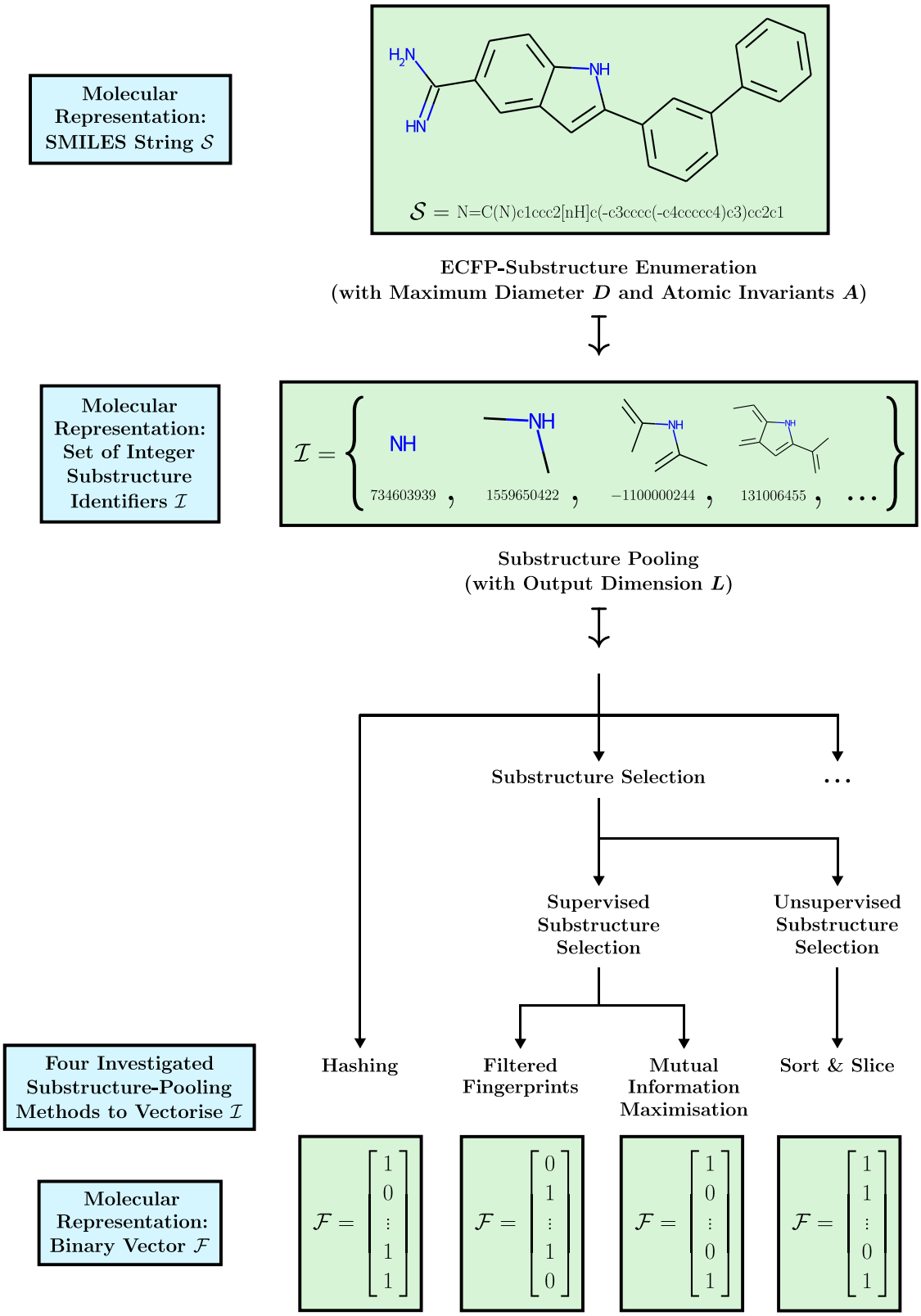

Figure 5.1: Schematic overview of the four investigated substructure-pooling methods for the vectorisation of ECFPs.

We imagine that the described substructure-pooling methods are used in the context of a supervised molecular property-prediction task specified via a set of n training compounds

$$\mathfrak{D} = \{\mathcal{R}_1, ..., \mathcal{R}_n\} \subset \mathfrak{R}$$

that are elements of some larger chemical space $\mathfrak{R}$ and that are given via their associated SMILES strings. We also assume that we are given a labelling function

$$c : \mathfrak{D} \to \mathbb{R}$$

that assigns regression or classification labels to the training compounds. The ECFP-algorithm is denoted via

$$\varphi : \mathfrak{R} \to P(\mathfrak{C})$$

and turns SMILES strings $\mathcal{R} \in \mathfrak{R}$ into sets of integer identifiers

$$\varphi(\mathcal{R}) = \{\mathcal{J}_1, ..., \mathcal{J}_r\} \subseteq \mathfrak{C}$$

that correspond to circular chemical substructures that appear in the input compound $\mathcal{R}$. For each substructure $\mathcal{J} \in \mathfrak{C}$, the set of all training compounds that contain $\mathcal{J}$ is called the support of $\mathcal{J}$ and is denoted via

$$\text{supp}(\mathcal{J}) := \{\mathcal{R} \in \mathfrak{D} \mid \mathcal{J} \in \varphi(\mathcal{R})\}.$$

The set

$$\mathfrak{C}_{\mathfrak{D}} := \bigcup_{\mathcal{R} \in \mathfrak{D}} \varphi(\mathcal{R})$$

contains all integer identifiers in the entire training set, i.e. all circular substructures that appear in any of the n training compounds.

5.2.2.1 Hashing

The canonical way for substructure pooling [16] is via a deterministic hash function

$$\tilde{h} : \{1, ..., 2^{32}\} \to \{1, ..., l\}$$

that compresses the set of integer identifiers in $\mathfrak{C}$ into a much smaller set of integers $\{1, ..., l\}$ whose cardinality corresponds to the desired fingerprint dimension l. Formally, one can employ h to define a substructure-pooling method via

$$\Psi : P(\mathfrak{C}) \to \mathbb{R}^l, \quad \Psi(\{\mathcal{J}_1, ..., \mathcal{J}_r\})_i = \begin{cases} 1 & \exists\, k \in \{1, ..., r\} : h(\mathcal{J}_k) = i, \\ 0 & \text{else.} \end{cases}$$

This means that the map

$$\Psi \circ \varphi : \mathfrak{R} \to \mathbb{R}^l$$

transforms a chemical compound $\mathcal{R} \in \mathfrak{R}$ into an l-dimensional binary vectors whose i-th component is 1 if and only if (at least) one of the substructures in $\varphi(\mathcal{R}) = \{\mathcal{J}_1, ..., \mathcal{J}_r\}$ gets hashed to the integer i. If l gets small then hash collisions start to occur that can for instance lead to distinct substructures in $\mathfrak{C}_{\mathfrak{D}}$ being mapped by h to the same component of the fingerprint. This degrades its interpretability and predictive performance. Note that hash-based substructure-pooling is independent of the training set $\mathfrak{D}$ and the labelling function c.

5.2.2.2 Sort & Slice

Let

$$f : \mathfrak{C} \to \{0, ..., n\}, \quad f(\mathcal{J}) = |\text{supp}(\mathcal{J})| = |\{\mathcal{R} \in \mathfrak{D} \mid \mathcal{J} \in \varphi(\mathcal{R})\}|,$$

be a function that maps every substructure-identifier $\mathcal{J} \in \mathfrak{C}$ to the number of training compounds in $\mathfrak{D} = \{\mathcal{R}_1, ..., \mathcal{R}_n\}$ which contain $\mathcal{J}$ (i.e. to its frequency in the training set). We can use f to define a linear order $\prec$ on the set of all substructures $\mathfrak{C}$ via

$$\mathcal{J} \prec \tilde{\mathcal{J}} \iff f(\mathcal{J}) < f(\tilde{\mathcal{J}}) \text{ or } [f(\mathcal{J}) = f(\tilde{\mathcal{J}}) \text{ and } \mathcal{J} < \tilde{\mathcal{J}}].$$

We see that the order defined by $\prec$ considers a substructure larger than another substructure if it appears in more training compounds. If two substructures appear in the same number of training compounds, ties are broken using the (arbitrary) ordering defined by the integer identifiers themselves. The relation $\prec$ defines a total order on $\mathfrak{C}$ which means that each substructure $\mathcal{J} \in \mathfrak{C}$ can be assigned a unique rank with respect to $\prec$. Let

$$s : \mathfrak{C} \to \{1, ..., m\}$$

be a bijective *sorting function* that assigns the ranks determined by $\prec$. Here rank 1 is assigned to the largest substructure with respect to $\prec$. If there are no ties then $s(\mathcal{J}) = 1$ implies that $\mathcal{J}$ appears in more training compounds than any other substructure in $\mathfrak{C}$. Now let

$$\gamma_s : \mathfrak{C} \to \mathbb{R}^m, \quad \gamma_s(\mathcal{J}) = u_{m,s(\mathcal{J})}$$

be the one-hot embedding associated with s (note that one-hot embeddings are described in Example 5.1). Furthermore, let $m_{\mathfrak{D}} := |\mathfrak{C}_{\mathfrak{D}}|$ be the total number of substructures that appear in the training set. Based on $m_{\mathfrak{D}}$ and the desired fingerprint

length l we define a *slicing function*

$$\eta_{m_{\mathfrak{D}},l} : \mathbb{R}^m \to \mathbb{R}^l, \quad \eta_{m_{\mathfrak{D}},l}(v_1, ..., v_m) = \begin{cases} (v_1, ..., v_l) & m_{\mathfrak{D}} \geq l, \\ (v_1, ..., v_{m_{\mathfrak{D}}}, 0, ..., 0) & m_{\mathfrak{D}} < l. \end{cases}$$

Then the *Sort & Slice* substructure-pooling operator is defined via

$$\Psi : P(\mathfrak{C}) \to \mathbb{R}^l, \quad \Psi(\{\mathcal{J}_1, ..., \mathcal{J}_r\}) = \eta_{m_{\mathfrak{D}},l}\Big(\sum_{i=1}^{r} \gamma_s(\mathcal{J}_i)\Big).$$

The sum expression

$$\sum_{i=1}^{r} \gamma_s(\mathcal{J}_i) \in \{0,1\}^m$$

is equal to a very long binary vector; each component of this vector indicates the presence or absence of a particular circular substructure in $\mathfrak{C}$ in the input compound. The substructures appear according to the frequency by which they occur in the training set such that substructures that occur in more training compounds appear earlier in the vector.

If $m_{\mathfrak{D}} \geq l$, which is usually the case, then the function $\eta_{m_{\mathfrak{D}},l}$ slices away the less frequent substructures from the vector to produce a final representation with the desired dimension l. In the unusual case where l is set to be larger than the total number of substructures in the training set $m_{\mathfrak{D}}$, then all training substructures are contained in the fingerprint and it is simply padded with additional 0s at the end to reach length l.

In simple terms, the map

$$\Psi \circ \varphi : \mathfrak{R} \to \mathbb{R}^l$$

outputs binary fingerprints of length l that indicate the presence or absence of the l substructures that appear most frequently in the training set $\mathfrak{D}$. In particular, note that these fingerprints do not exhibit hash collisions; each vectorial component can be assigned to a unique substructure. This clarity comes at the price of losing the information contained in the less frequent substructures in $\mathfrak{C}_{\mathfrak{D}}$ that are sliced away. Note however that in real-world chemical data sets a vast portion of substructures that exist in the training set only occur in a few compounds. In a machine-learning context, slicing away such highly infrequent substructures should thus lead to minimal loss of information since it corresponds to removing almost-constant features (i.e. almost-constant columns the the training-set feature matrix). The Sort & Slice operator Ψ is dependent on the training set $\mathfrak{D}$ but not on the training labels c. It can be interpreted as a simple unsupervised feature selection technique that selects substructures according to the frequency with which they appear in the training set.

In our literature review we discovered that MacDougall [48] proposed a version of the ECFP that closely resembles the ECFP generated via Sort & Slice. The main difference between the substructure-pooling method proposed by MacDougall and ours appears to be in the slicing operation. MacDougall proposes to arrange substructures in discrete packages according to the level sets of f:

$$\mathfrak{C}_{j,f} := \{\mathcal{J} \in \mathfrak{C} \mid f(\mathcal{J}) = j\}.$$

It is clear that the sets $\mathfrak{C}_{n,f}, ..., \mathfrak{C}_{0,f}$ form a partition of $\mathfrak{C}$. MacDougall only absorbs substructures in $\mathfrak{C}_{n,f}, ..., \mathfrak{C}_{n-k,f}$ into the final fingerprint whereby k is a tunable *slicing parameter*. In this setting, tuning k gives limited control over the fingerprint dimension as going from k to $k+1$ leads to a discrete jump in fingerprint length of magnitude $|\mathfrak{C}_{n-(k+1),f}|$. This coarse-graining of the fingerprint length might be motivated by an effort to avoid dealing with ties when sorting substructures according to frequency. In contrast, our Sort & Slice technique solves the problem of ties by introducing a second (arbitrary) layer of ordering based on the magnitude of the integer identifiers. This allows us to assign a unique rank to every substructure $\mathcal{J} \in \mathfrak{C}$. As a consequence, our framework easily allows full control over the fingerprint length l as the substructures in $\mathfrak{C}$ can be sorted in an unambiguous way. Subsequently, all but the top l substructures can be sliced away for any desired l.

Remark 5.2 (Information-Theoretic Interpretation of Sort & Slice)**.** Let us assume that no substructure appears in more than half of the training compounds:

$$\max_{\mathcal{J} \in \mathfrak{C}_{\mathfrak{D}}} f(\mathcal{J}) \leq \frac{n}{2}.$$

In other words, we assume that

$$\forall \mathcal{J} \in \mathfrak{C}_{\mathfrak{D}} : \quad p(\mathcal{J}) := \frac{f(\mathcal{J})}{n} \leq \frac{1}{2}.$$

Here $p(\mathcal{J}) \in (0, 1]$ is the probability to find substructure $\mathcal{J} \in \mathfrak{C}_{\mathfrak{D}}$ in a training compound that was chosen uniformly at random from $\mathfrak{D}$. It is also an empirical estimate of the probability to find substructure $\mathcal{J}$ in a compound that was sampled in the same way as the training compounds which are assumed to have been generated via independent and identically distributed draws from the larger chemical space $\mathfrak{R}$. In real-world data sets the assumption that $p(\mathcal{J}) \leq 1/2$ holds approximately true since usually only a very small fraction of all circular substructures in $\mathfrak{C}_{\mathfrak{D}}$ actually appear in the majority of compounds; in fact, in real-word data sets almost all substructures in $\mathcal{J} \in \mathfrak{C}_{\mathfrak{D}}$ tend to appear in only a few compounds. If indeed no substructure appears

in the majority of training compounds, the ordering imposed on $\mathfrak{C}_{\mathfrak{D}}$ by the frequency-function f is equivalent to the ordering imposed by the empirical *information entropy* $H \circ p$ of a substructure with respect to $\mathfrak{D}$. The binary information entropy function H [158] is defined via

$$H : [0,1] \to [0,1], \quad H(p) = -p \log_2(p) - (1-p) \log_2(1-p)$$

with $0 * \log_2(0) := 0$. We naturally define the empirical information entropy of a substructure in the training set $\mathfrak{D}$ via

$$H \circ p : \mathfrak{C}_{\mathfrak{D}} \to [0,1], \quad (H \circ p)(\mathcal{J}) = H(p(\mathcal{J})).$$

The quantity $H(p(\mathcal{J}))$ peaks at $p(\mathcal{J}) = 1/2$ and provides an empirical measure for how informative the substructure $\mathcal{J}$ is when used as a binary feature in a fingerprint. It represents a simple plug-in entropy estimator based on the empirical probability estimate $p(\mathcal{J})$ to find substructure $\mathcal{J}$ in a compound. $H(p)$ is strictly increasing for $p \in [0, 1/2]$ and is strictly decreasing for $p \in [1/2, 1]$. To see that f and $H \circ p$ induce the exact same ordering on $\mathfrak{C}_{\mathfrak{D}}$ note the following two facts:

$$\begin{aligned}
\forall \mathcal{J}, \tilde{\mathcal{J}} \in \mathfrak{C}_{\mathfrak{D}} : \quad & f(\mathcal{J}) < f(\tilde{\mathcal{J}}) \iff p(\mathcal{J}) < p(\tilde{\mathcal{J}}) \iff (H \circ p)(\mathcal{J}) < (H \circ p)(\tilde{\mathcal{J}}), \\
\forall \mathcal{J}, \tilde{\mathcal{J}} \in \mathfrak{C}_{\mathfrak{D}} : \quad & f(\mathcal{J}) = f(\tilde{\mathcal{J}}) \iff p(\mathcal{J}) = p(\tilde{\mathcal{J}}) \iff (H \circ p)(\mathcal{J}) = (H \circ p)(\tilde{\mathcal{J}}).
\end{aligned}$$

Here in each of both rows the first equivalence is trivial and the second equivalence holds due to our the strict monotonicity of H on $[0, 1/2]$ and our current assumption that $p(\mathcal{J}), p(\tilde{\mathcal{J}}) \in [0, 1/2]$.

This argument shows that in realistic settings (i.e. in settings where almost all substructures appear in no more than half of the training compounds) Sort & Slice tends to automatically lead to a fingerprint that only contains the most informative substructures from an entropic point of view, all while being much simpler to theoretically understand, implement and interpret than an approach explicitly built on empirical information entropy.

5.2.2.3 Filtering

In this section we describe the substructure-pooling technique proposed by Gütlein and Kramer [157] referred to as *filtering*. Their original method was published in `Java`; we used the technical description from their article to create a reimplementation in `Python` that integrates with the rest of our code base.

Let us first assume that we are given a binary molecular classification problem, i.e. we assume that our labelling function is given by

$$c : \mathfrak{D} \to \{0, 1\}.$$

If instead the initial labels specified by c correspond to a continuous regression problem with labels in $\mathbb{R}$, then we set all labels below or above the label median to 0 or 1 respectively, to still arrive at a binary classification problem of the above form.

Now let R be a random compound that was drawn from $\mathfrak{R}$ according to some probability distribution. Moreover, we imagine that the given training set

$$\mathfrak{D} = \{\mathcal{R}_1, ..., \mathcal{R}_n\}$$

represents a statistical sample of n independent and identically distributed draws of R from $\mathfrak{R}$. Then the available training labels

$$\hat{c} := (c(\mathcal{R}_1), ..., c(\mathcal{R}_n)) \in \{0, 1\}^n$$

form a statistical sample of size n for the random labels of R. Furthermore, for each substructure $\mathcal{J} \in \mathfrak{C}$, the expression

$$g_{\mathcal{J}}(R) = \begin{cases} 1 & \mathcal{J} \in \varphi(R), \\ 0 & \text{else}, \end{cases}$$

forms a random variable that is equal to 1 if and only if substructure $\mathcal{J}$ is contained in R. The binary sequence

$$\hat{g}_{\mathcal{J}} := (g_{\mathcal{J}}(\mathcal{R}_1), ..., g_{\mathcal{J}}(\mathcal{R}_n)) \in \{0, 1\}^n$$

represents a statistical sample of size n for $g_{\mathcal{J}}(R)$.

We now define a function

$$f : \mathfrak{C} \to [0, 1], \quad f(\mathcal{J}) = p_{\chi^2}(\hat{c}, \hat{g}_{\mathcal{J}}),$$

that assigns to each substructure $\mathcal{J} \in \mathfrak{C}$ its p-value in a statistical χ^2 independence-test [161] between $\hat{c}$ and $\hat{g}_{\mathcal{J}}$. As is the case for Sort & Slice, this function allows one to define a total order on $\mathfrak{C}$ via

$$\mathcal{J} \prec \tilde{\mathcal{J}} \iff f(\mathcal{J}) > f(\tilde{\mathcal{J}}) \text{ or } [f(\mathcal{J}) = f(\tilde{\mathcal{J}}) \text{ and } \mathcal{J} > \tilde{\mathcal{J}}].$$

The larger the p-value, the smaller the substructure according to $\prec$. We now reduce the number of substructures we consider in our fingerprint to the desired dimension l via the following scheme:

- **Step 0:** The set of selected substructures is initialised via $\mathfrak{C}_l := \mathfrak{C}$.
- **Step 1:**. A substructure $\mathcal{J} \in \mathfrak{C}_l$ that fulfills $|\text{supp}(\mathcal{J})| \leq 1$ is randomly chosen and removed from $\mathfrak{C}_l$. This is repeated until all substructures in $\mathfrak{C}_l$ appear in at least two training compounds or until $|\mathfrak{C}_l| = l$.
- **Step 2:** A substructure $\mathcal{J} \in \mathfrak{C}_l$ that is *non-closed* is randomly chosen and removed from $\mathfrak{C}_l$. This is repeated until all remaining substructures in $\mathfrak{C}_l$ are closed or until $|\mathfrak{C}_l| = l$. Note that a substructure $\mathcal{J} \in \mathfrak{C}_l$ is called *non-closed* if there exists another substructure $\tilde{\mathcal{J}} \in \mathfrak{C}_l$ such that $\text{supp}(\mathcal{J}) = \text{supp}(\tilde{\mathcal{J}})$ and $\mathcal{J}$ contains a proper subgraph that is isomorphic to $\tilde{\mathcal{J}}$.
- **Step 3:** The smallest element of $\mathfrak{C}_l$ with respect to the linear order $\prec$ is chosen and removed. This is repeated until $|\mathfrak{C}_l| = l$.

Step 1 is performed to remove almost-constant substructural features that contain little information. Step 2 represents a graph-theoretic attempt to reduce feature redundancy via the removal of substructures that contain smaller substructures that match the exact same set of training compounds. Finally, Step 3 is performed to select the l substructures that show the strongest statistical dependence on the training label as quantified by a χ^2-test. Using the selection of substructures $\mathfrak{C}_l$ one can construct a one-hot embedding (see Example 5.1)

$$\gamma_s : \mathfrak{C} \to \mathbb{R}^l, \quad \gamma_s(\mathcal{J}) = \begin{cases} u_{l,s(\mathcal{J})} & \mathcal{J} \in \mathfrak{C}_l, \\ 0 & \text{else}, \end{cases}$$

whereby

$$s : \mathfrak{C}_l \to \{1, ..., l\}$$

is some arbitrary bijective sorting function. Substructure pooling by means of filtered fingerprints can now be described via:

$$\Psi : P(\mathfrak{C}) \to \mathbb{R}^l, \quad \Psi(\{\mathcal{J}_1, ..., \mathcal{J}_r\}) = \sum_{i=1}^{r} \gamma_s(\mathcal{J}_i).$$

The map

$$\Psi \circ \varphi : \mathfrak{R} \to \mathbb{R}^l$$

transforms chemical compounds into hash collision-free binary fingerprints that only indicate the presence or absence of substructures in $\mathfrak{C}_l$. Note again that $\mathfrak{C}_l$ contains the l substructures that exhibit the lowest p-values in a χ^2-test with respect to the training label; these substructures thus have a comparatively high statistical dependence with

the target variable and might therefore be useful features for a machine-learning system. Ψ depends on both the training compounds in $\mathfrak{D}$ and the training labels c. Filtered fingerprints form a type of supervised feature selection scheme.

5.2.2.4 Mutual-Information Maximisation

In this section we continue to assume the same setting as in the previous section, i.e. we assume that our labelling function is binary (or has been binarised),

$$c : \mathfrak{D} \to \{0, 1\},$$

and we consider the binary sequences $\hat{c} \in \{0, 1\}^n$ and $\hat{g}_{\mathcal{J}} \in \{0, 1\}^n$ for $\mathcal{J} \in \mathfrak{C}$ to be statistical samples of size n of two random variables that describe whether or not the binary label of a randomly chosen compound is positive and whether or not the compound contains substructure $\mathcal{J}$. Based on these samples derived from our training set, we compute the empirical *mutual information* I [158, 159] between substructure $\mathcal{J}$ and the training label via

$$I(\hat{c}, \hat{g}_{\mathcal{J}}) = H(\hat{c}) + H(\hat{g}_{\mathcal{J}}) - H(\hat{c}, \hat{g}_{\mathcal{J}}).$$

Here H denotes an empirical estimate of the information entropy of a random variable based on a statistical sample. H could be implemented using a variety of strategies for entropy estimation; since we are dealing with the relatively easy case of discrete binary variables, we implement H as the simple plug-in entropy estimator based on the relative frequencies of binary outcomes that was already used in Remark 5.2. $I(\hat{c}, \hat{g}_{\mathcal{J}})$ is a nonnegative, symmetric and nonlinear measure of the statistical dependence between $\hat{c}$ and $\hat{g}_{\mathcal{J}}$. The larger $I(\hat{c}, \hat{g}_{\mathcal{J}})$, the more information the presence of substructure $\mathcal{J}$ in a compound conveys about the value of its training label and *vice versa*.

We now define a function

$$f : \mathfrak{C} \to [0, \infty), \quad f(\mathcal{J}) = I(\hat{c}, \hat{g}_{\mathcal{J}}),$$

that assigns to each substructure $\mathcal{J} \in \mathfrak{C}$ its empirical mutual information with the training label. Once again this function allows one to define a total order on $\mathfrak{C}$ via

$$\mathcal{J} \prec \tilde{\mathcal{J}} \iff f(\mathcal{J}) < f(\tilde{\mathcal{J}}) \text{ or } [f(\mathcal{J}) = f(\tilde{\mathcal{J}}) \text{ and } \mathcal{J} < \tilde{\mathcal{J}}].$$

We go on to reduce the number of substructures we consider in our fingerprint to the desired length l via the following scheme:

- **Step 0:** The set of substructures is initialised via $\mathfrak{C}_l := \mathfrak{C}$.
- **Step 1:** If two substructures $\mathcal{J}, \tilde{\mathcal{J}} \in \mathfrak{C}_l$ appear in the exact same set of training compounds, i.e. if $\text{supp}(\mathcal{J}) = \text{supp}(\tilde{\mathcal{J}})$, then one of the substructures is chosen uniformly at random and removed from $\mathfrak{C}_l$. This is repeated until no two substructures have the same support or until $|\mathfrak{C}_l| = l$.
- **Step 2:** The smallest element of $\mathfrak{C}_l$ with respect to the linear order $\prec$ is chosen and removed. This is repeated until $|\mathfrak{C}_l| = l$.

Step 1 is performed in an attempt to reduce feature redundancy via the removal of substructural features that are identical in the training set. Then, Step 2 is performed to select only the l most informative substructures with respect to the training label. Using the selection $\mathfrak{C}_l$ one can construct a one-hot embedding (see Example 5.1)

$$\gamma_s : \mathfrak{C} \to \mathbb{R}^l, \quad \gamma_s(\mathcal{J}) = \begin{cases} u_{l,s(\mathcal{J})} & \mathcal{J} \in \mathfrak{C}_l, \\ 0 & \text{else}, \end{cases}$$

whereby

$$s : \mathfrak{C}_l \to \{1, ..., l\}$$

is some arbitrary bijective sorting function. Substructure pooling based on mutual-information maximisation (MIM) can now be described with the following operator:

$$\Psi : P(\mathfrak{C}) \to \mathbb{R}^l, \quad \Psi(\{\mathcal{J}_1, ..., \mathcal{J}_r\}) = \sum_{i=1}^{r} \gamma_s(\mathcal{J}_i).$$

The map

$$\Psi \circ \varphi : \mathfrak{R} \to \mathbb{R}^l$$

transforms chemical compounds into hash-collision-free binary fingerprints that exclusively indicate the presence or absence of substructures in the tailored set $\mathfrak{C}_l$. Remember that $\mathfrak{C}_l$ contains the l substructures that exhibit the highest mutual information with the training label and should thus be highly predictive in a supervised machine-learning setting. Ψ depends on the training compounds in $\mathfrak{D}$ and the training labels c. It can be seen as a supervised feature selection scheme.

5.2.3 Experimental Setup

We computationally evaluated the predictive performance of the four substructure-pooling techniques introduced in the previous section (Hash, Sort & Slice, Filter and MIM) using five molecular property prediction data sets (see Table 5.1). The data sets

were chosen to cover a diverse set of chemical regression and binary classification tasks: the prediction of lipophilicity, aqueous solubility, binding affinity, and mutagenicity. We also included a LIT-PCBA virtual screening data set [162] that represents a highly imbalanced binary classification problem. Also note that we once again experimented with the same SARS-CoV-2 main protease binding affinity data set that we already explored in Chapters 3 and 4.

All data sets were cleaned in the following manner: SMILES strings were algorithmically standardised and desalted using the ChEMBL structure pipeline [124]. This step also removed solvents and isotopic information. Afterwards, SMILES strings that generated error messages upon being turned into an `RDKit` mol object were deleted. Furthermore, a scan for duplicate SMILES strings was performed; if two SMILES strings were found to be identical, one of the SMILES strings was deleted uniformly at random along with its training label. Finally, we also detected rare instances where SMILES strings appeared to encode several disconnected fragments instead of one connected compound; such SMILES strings were too deleted from the data.

As a data splitting strategy, we implemented k-fold cross validation repeated with m random seeds using $(k, m) = (2, 3)$; thus each model was independently trained and tested $km = 6$-times. Performance results were recorded as the average and standard deviation over these 6 splits, using the mean absolute error (MAE) for the three regression data sets and the area under the receiver operating characteristic curve (AUROC) for the balanced mutagenicity classification data set. To measure performance on the LIT-PCBA estrogen receptor alpha antagonism classification data

Prediction Task	**Task Type**	**Compounds**	**Source**
Lipophilicity [logD]	Regression	4200	MoleculeNet [163]
Aqueous Solubility [logS]	Regression	9335	Sorkun et al. [164]
SARS-CoV-2 Main Protease Binding Affinity [pIC_{50}]	Regression	1924	COVID Moonshot Project [123]
Ames Mutagenicity	Classification	3496 positives 3009 negatives	Hansen et al. [165]
Estrogen Receptor Alpha Antagonism	Classification	88 positives 3833 negatives	LIT-PCBA [162]

Table 5.1: Overview of the five cleaned molecular property prediction data sets used to experimentally evaluate the predictive performance of distinct substructure-pooling techniques for ECFPs.

Machine-Learning Model Hyperparameters	
Random Forest	**Multilayer Perceptron**
NumberOfTrees = 100 MaxDepth = None MinSamplesLeaf= 1 MinSamplesSplit = 2 Bootstrapping = True MaxFeatures = Sqrt Criterion (regression) = SquaredError Criterion (classification) = Gini	NumberOfHiddenLayers = 5 NeuronsPerHiddenLayer = 512 HiddenActivation = ReLU UseBiasVectors = True DropoutRateHiddenLayers = 0.25 BatchNormHiddenLayers = True BatchSize = 64 LearningRate = 1e-3 LRDecayFactor = $\max\{0.98^{\text{epoch}}, \text{1e-2}\}$ WeightDecayFactor = 0.1 NumberOfEpochs = 250 Optimiser = AdamW [134] OutputActivation (regression) = Identity Loss (regression) = MeanSquaredError OutputActivation (classification) = Sigmoid Loss (classification) = BinaryCrossEntropy

Table 5.2: Selected hyperparameters for the two prediction models used in our substructure-pooling experiments: random forests (RFs) and multilayer perceptrons (MLPs).

set, we used the area under the precision recall curve (AUPRC) which quantifies the tradeoff between sensitivity (= recall) and precision; the AUPRC is a suitable and commonly used metric for highly imbalanced problems in which positives are of stronger natural interest than negatives. We experimented with two distinct splitting techniques within our cross-validation framework: standard uniform *random* splitting and *scaffold* splitting [166]. For the LIT-PCBA classification data set we used *stratified* random splitting instead of standard random splitting in order to stabilise the small number of positives across training and test sets (we still refer to this split simply as a random split). Unlike random splitting, scaffold splitting generates a partition of a chemical data set in which the molecular scaffolds of all training-set compounds are distinct from the molecular scaffolds of all test-set compounds. This creates a distributional shift between training and test set which leads to a more challenging prediction scenario where a model is trained in one structural area of chemical space but tested in another.

As prediction algorithms we selected two standard machine-learning models: random forests (RFs) and multilayer perceptrons (MLPs). The hyperparameter choices for both models are listed in Table 5.2. All MLPs were trained on a single NVIDIA GeForce RTX 3060 GPU. In the case of RFs, the chosen hyperparameters are identical to the default ones from scikit-learn [130] with the exception for MaxFeatures

which was set to "Sqrt" instead of 1.0 in the case of RF-regressors in order to add randomness.[4]

We conducted a thorough investigation of the ECFP hyperparameter space. For each data set, for each data splitting technique (random vs. scaffold), and for each prediction model (RF vs. MLP), we evaluated 24 different ECFPs based on a complete exploration of the following grid:

- fingerprint dimension $l \in \{512, 1024, 2048, 4096\}$,
- substructure diameter $D \in \{2, 4, 6\}$,
- atomic invariants $\in$ {standard (ECFP), pharmacophoric (FCFP)},
- active tetrahedral R-S chirality flags.

Each of the 24 ECFP-versions was further combined with all four substructure-pooling methods (Hash, Sort & Slice, Filter, MIM). This resulted in 96 distinct vectorial ECFPs used for each combination of data set, splitting type and machine-learning model. From a bird's-eye view, the conducted experiments are organised according to a robust combinatorial methodology of the following form:

$$
\begin{gathered}
|\{\text{Lipophilicity Data Set}, ..., \text{Estrogen Receptor Alpha Antagonism Data Set}\}| \\
\times \\
|\{\text{Random Data Split},\ \text{Scaffold Data Split}\}| \\
\times \\
|\{\text{Random Forest},\ \text{Multilayer Perceptron}\}| \\
\times \\
|\{\text{512-Bit ECFP2}, ..., \text{4096-Bit FCFP6}\}| \\
\times \\
|\{\text{Hash},\ \text{Filter},\ \text{MIM},\ \text{Sort \& Slice}\}| \\
= \\
5 * 2 * 2 * 24 * 4 = 1920.
\end{gathered}
$$

[4]The fact that the default scikit-learn setting of MaxFeatures = 1.0 for RF regressors actually does not generate a classical random forest based on trees built via random subsets of features but rather simply a set of bagged decision trees via bootstrap aggregation was pointed out in a social media post by Greg Landrum on Twitter via @dr_greg_landrum on 1:57 PM, Feb 28, 2023: *I assume there's a reason for it, but I really don't think it's a feature that the default parameters for a scikit-learn RandomForestRegressor don't actually build a random forest.*

Each of the 1920 modelling scenarios resulting from this combinatorial setup was evaluated on the chosen data set via 2-fold cross validation with 3 random seeds as mentioned before. In total we therefore trained $1920 * 6 = 11520$ models, half of which were deep-learning models.

5.3 Results and Discussion

The detailed experimental results for each data set can be found in Figures 5.2 to 5.6. A comprehensive overview of all results is depicted in Figure 5.7 where it becomes evident that Sort & Slice outperforms hashing in almost all scenarios and that frequently the achieved performance gains are non-negligible. For example, a random forest trained on a random split of the AqSolDB solubility data set [164] achieves a median MAE of about 1.045 when combined with the hashed versions of the 24 investigated ECFPs but a median MAE of about 0.998 if the ECFPs are vectorised via Sort & Slice instead. This corresponds to a relative improvement of the median MAE of approximately 4.5%.

The fine-grained results in Figures 5.2 to 5.6 give more detailed insights into the relative performance of substructure-pooling techniques for various ECFP hyperparameters. In particular, this allows us to track the performance of the highly popular 1024-bit ECFP4 which has been used as one of the common fingerprints in countless applications. We see that the Sort & Slice version of the 1024-bit ECFP4 surpasses the predictive performance of the hashed 1024-bit ECFP4 in all but a few cases. For instance, Figure 5.2 shows that replacing hashing with Sort & Slice when using a 1024-bit ECFP4 with an MLP on a random split of the lipophilicity data set leads to a rather remarkable relative MAE-improvement of 11.37%.

The results in Figure 5.6 suggest that the advantage of Sort & Slice over hashing remains robust even in a highly imbalanced classification scenario. The superior AUPRC of Sort & Slice indicates a better trade-off between sensitivity and precision in a setting with very few positives where standard algorithms tend to generate predictions that are heavily biased towards the negative class (i.e. heavily biased towards extremely high precision and extremely low sensitivity). Our observations are consistent with the hypothesis that Sort & Slice exerts a mitigating effect on this bias relative to hashing by increasing sensitivity favourably at the cost of precision, leading to a more balanced classifier with stronger overall performance. The predictive power of the RFs and MLPs trained on the imbalanced LIT-PCBA estrogen receptor alpha antagonism data set could potentially be further increased by combining them

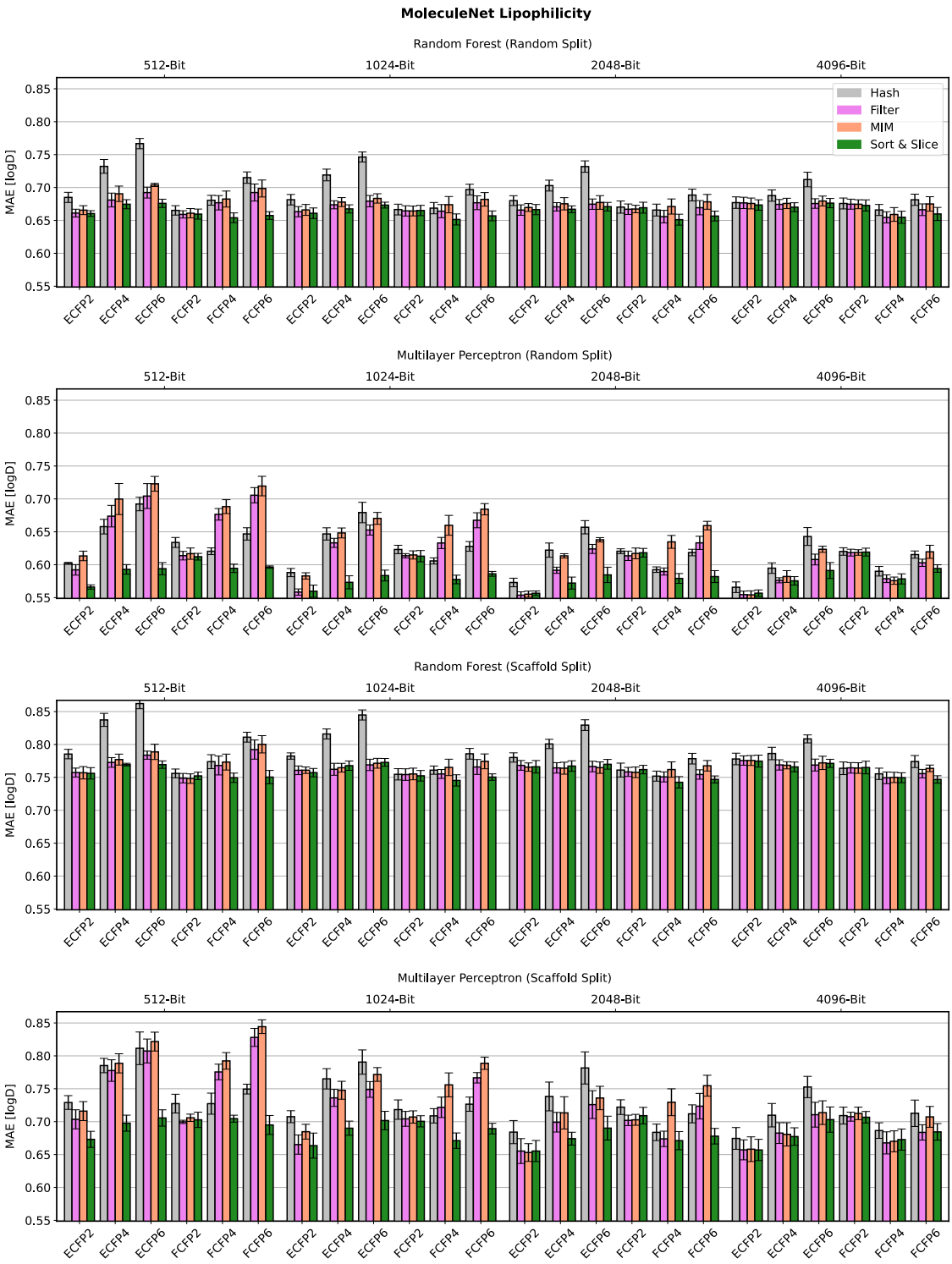

Figure 5.2: Predictive performance of the four substructure-pooling methods (indicated by colours) for the lipophilicity regression data set using varying data splitting techniques, prediction models and ECFP hyperparameters. Each coloured bar shows the average mean absolute error (MAE) of the respective model across a k-fold cross validation scheme repeated with m random seeds for $(m, k) = (3, 2)$. The length of each error bar equals twice the standard deviation of the performance measured over the $mk = 6$ trained models.

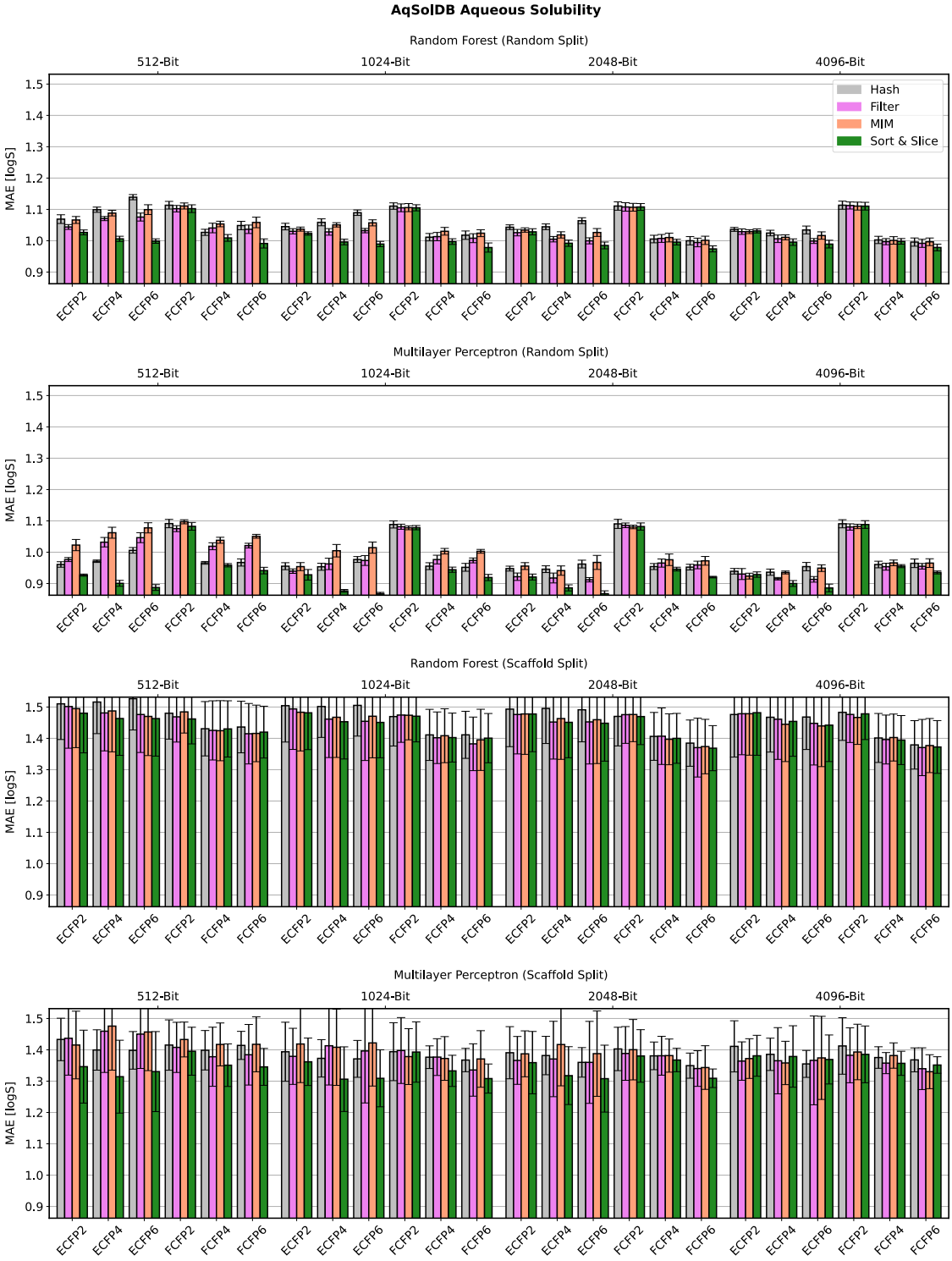

Figure 5.3: Predictive performance of the four substructure-pooling methods (indicated by colours) for the solubility regression data set using varying data splitting techniques, prediction models and ECFP hyperparameters. Each coloured bar shows the average mean absolute error (MAE) of the respective model across a k-fold cross validation scheme repeated with m random seeds for $(m, k) = (3, 2)$. The length of each error bar equals twice the standard deviation of the performance measured over the $mk = 6$ trained models.

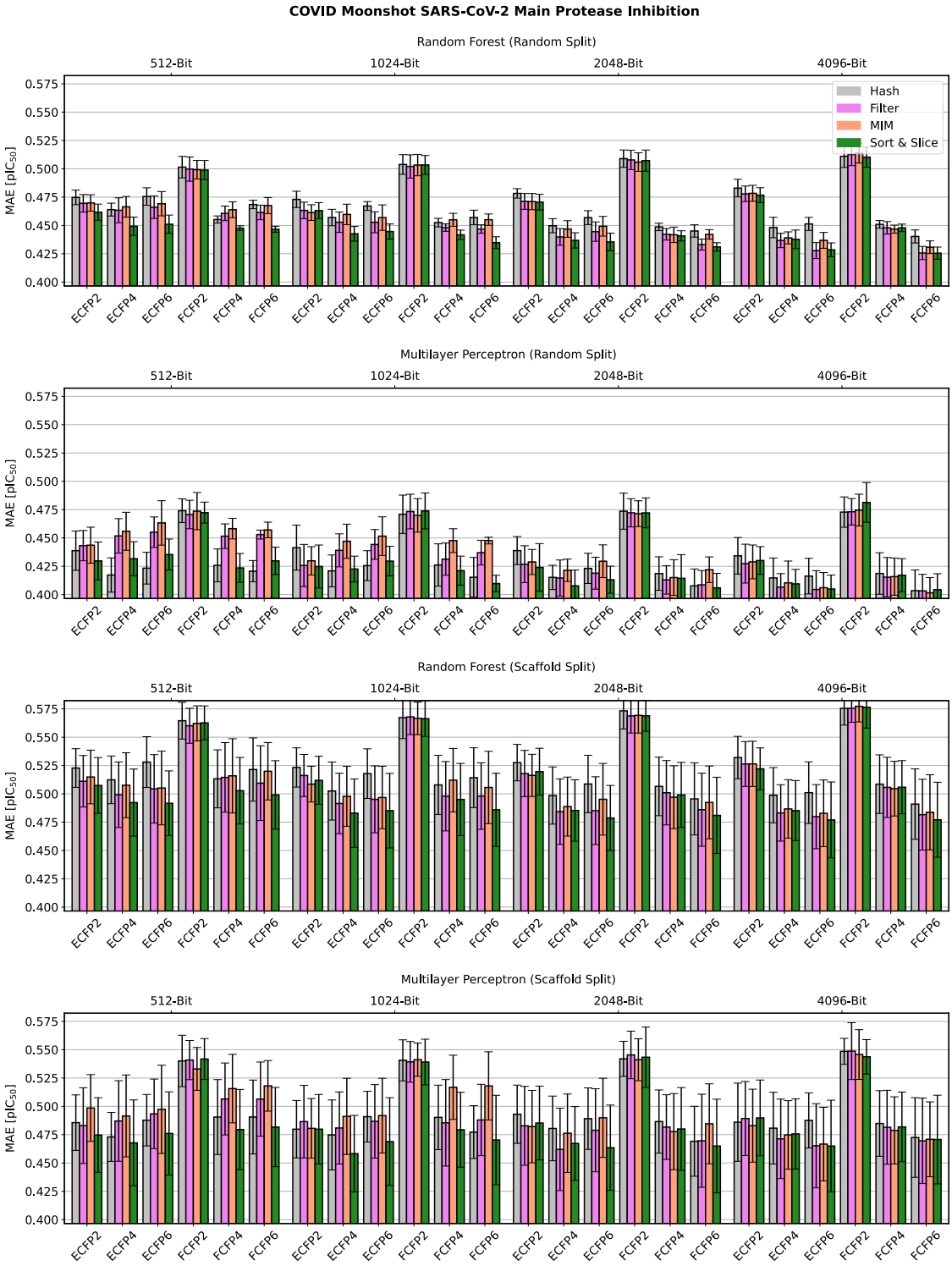

Figure 5.4: Predictive performance of the four substructure-pooling methods (indicated by colours) for the SARS-CoV-2 main protease binding affinity regression data set using varying data splitting techniques, prediction models and ECFP hyperparameters. Each coloured bar shows the average mean absolute error (MAE) of the respective model across a k-fold cross validation scheme repeated with m random seeds for $(m, k) = (3, 2)$. The length of each error bar equals twice the standard deviation of the performance measured over the $mk = 6$ trained models.

Figure 5.5: Predictive performance of the four substructure-pooling methods (indicated by colours) for the balanced mutagenicity classification data set using varying data splitting techniques, prediction models and ECFP hyperparameters. Each coloured bar shows the average area under the receiver operating characteristic curve (AUROC) of the respective model across a k-fold cross validation scheme repeated with m random seeds for $(m, k) = (3, 2)$. The length of each error bar equals twice the standard deviation of the performance measured over the $mk = 6$ trained models.

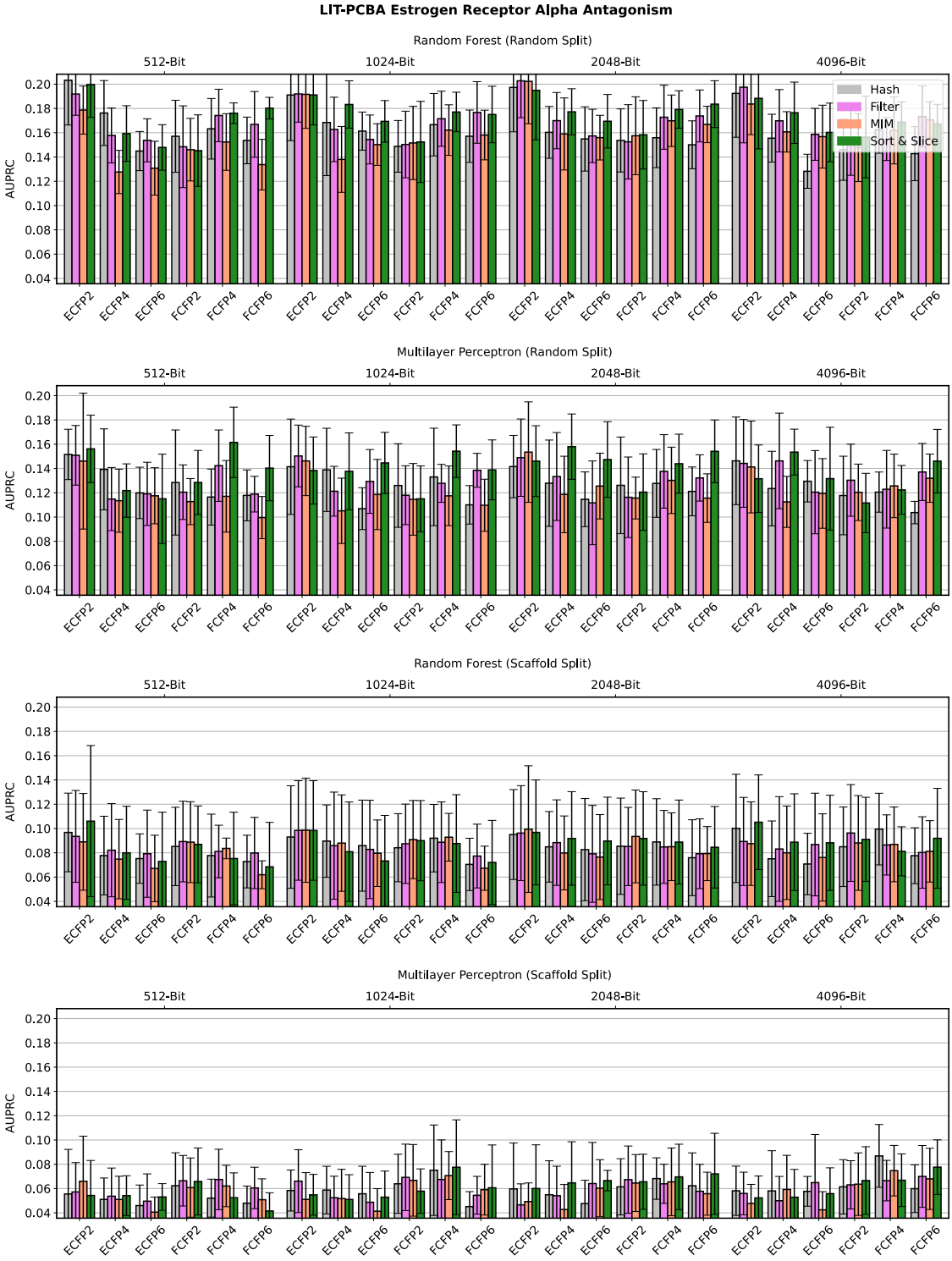

Figure 5.6: Predictive performance of the four substructure-pooling methods (indicated by colours) for the imbalanced estrogen receptor alpha antagonism classification data set using varying data splitting techniques, prediction models and ECFP hyperparameters. Each coloured bar shows the average area under the precision recall curve (AUPRC) of the respective model across a k-fold cross validation scheme repeated with m random seeds for $(m, k) = (3, 2)$. The length of each error bar equals twice the standard deviation of the performance measured over the $mk = 6$ trained models.

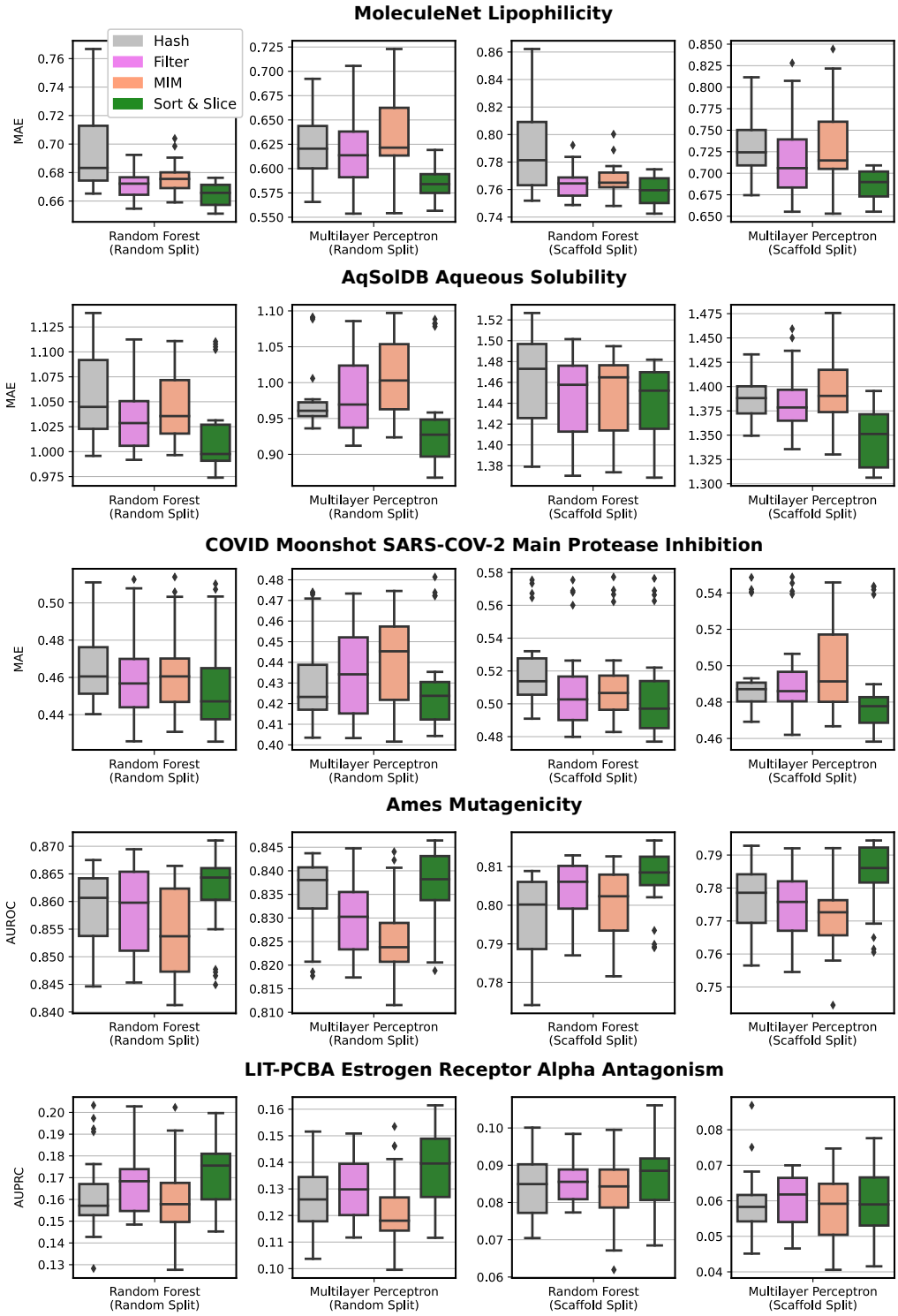

Figure 5.7: Overview of the predictive performance of the four investigated substructure-pooling methods (indicated by colours) across regression and classification data sets, data splitting techniques and prediction models. Each boxplot visualises the performance of a substructure-pooling method on top of 24 distinct ECFP-types generated by combining fingerprint dimensions $l \in \{512, 1024, 2048, 4096\}$, fingerprint diameters $D \in \{2, 4, 6\}$ and initial atomic invariants $\in$ {standard, pharmacophoric}.

with computational techniques explicitly tailored to counteract class imbalance (oversampling, undersampling, tree-wise undersampling, weighted loss, balanced training batches, ...). Note though that commonly used balancing techniques such as oversampling or undersampling may interact with methods like Sort & Slice by changing the relative frequencies of substructures in the training set. Since the primary goal of this study was to evaluate the relative performance of substructure-pooling techniques in a clear and technically straightforward setting, we thus decided not to add this additional layer of complexity to our experiments. However, combining techniques to counteract class imbalance with substructure-pooling methods like Sort & Slice might reveal unknown synergies and could form an interesting project for future research.

Parts of Figures 5.2 to 5.6 seem to suggest that the improvements achieved via Sort & Slice over hashing tend to become more pronounced as the fingerprint length decreases, the fingerprint diameter increases and as standard atomic invariants are used instead of pharmacophoric invariants. These observations strongly support the idea that the predictive advantage of Sort & Slice over hashing stems at least partially from an absence of bit collisions: unlike ECFPs generated via Sort & Slice, hashed ECFPs exhibit more and more bit collisions as the dimension of the fingerprint descreases relative to the number of substructures identified in the data set. The number of identified substructures in turn increases with the fingerprint diameter and when switching from pharmacophoric to standard atomic invariants. An increase in bit collisions then seems to degrade the predictive performance of hashed ECFPs relative to those generated by Sort & Slice.

A question that remains, however, is whether the predictive advantage of Sort & Slice is merely a product of the general avoidance of bit collisions via the selection of a subset of substructures instead of the hashing of all substructures; or if and to what extent the particular unsupervised substructure selection scheme underlying Sort & Slice (i.e. sorting of substructures according to their frequency in the training set and subsequent exclusion of rare substructures) independently contributes to the performance gain. Surprisingly, Figure 5.7 shows that Sort & Slice not only beats hashing, but it also consistently outperforms the two other investigated substructure-pooling techniques (filtering and MIM), whose respective performance measurements tend to fall between hashing and Sort & Slice. Note that just like Sort & Slice, both of these techniques are based on substructure selection and lead to fingerprints that are entirely free of bit collisions.

These observations reveal two points. Firstly, the performance gains provided by Sort & Slice, filtering and MIM over standard hashing are not purely the result of

avoiding bit collisions via substructure selection; but the specific strategy by which substructures are selected for the final fingerprint does indeed make an important difference for downstream predictive performance. Secondly, and perhaps remarkably, the extremely simple frequency-based substructure selection strategy implemented by Sort & Slice outperforms the more technically advanced substructure selection schemes underlying filtering and MIM. This is in spite of the fact that, unlike MIM and filtering, Sort & Slice is an unsupervised technique that does not utilise any information associated with the training label. It appears surprising that Sort & Slice would beat technically sophisticated supervised feature selection methods such as filtering or MIM that select substructures using task-specific information. While the reasons for this are not obvious, it is generally conceivable that exploiting the training label when selecting substructures could potentially harm the generalisation abilities of a machine-learning system by contributing to its risk of overfitting to the training data (just like any other aspect of supervised model training could).

A natural extension of our study for a future research project could be to include additional substructure-selection techniques. One interesting approach would be to only select substructures that maximise feature variance based on the training set. If $p(\mathcal{J}) \in (0, 1]$ represents the fraction of training compounds in which a detected substructure $\mathcal{J}$ is present, then its associated empirical feature variance in the context of a binary fingerprint is given by $p(\mathcal{J})(1 - p(\mathcal{J}))$. The maximisation of feature variance would correspond to the removal of almost-constant columns of the feature matrix and would reflect a common data preparation strategy from traditional QSAR modelling. Both the empirical feature-variance $p(\mathcal{J})(1 - p(\mathcal{J}))$ of a substructure $\mathcal{J}$ and its empirical information entropy $H(p(\mathcal{J}))$ as introduced in Remark 5.2 peak when $p(\mathcal{J}) = 1/2$, i.e. when $\mathcal{J}$ is present in exactly half of all training compounds. It is easy to prove that, in the case of binary fingerprints, ranking substructures according to $p(\mathcal{J})(1 - p(\mathcal{J}))$ is equivalent to ranking them according to $H(p(\mathcal{J}))$. In this sense, feature-variance maximisation is equivalent to entropy maximisation and both methods translate to the removal of high-frequency as well as low-frequency substructures. At first glance, this approach might seem significantly different from Sort & Slice which is based on the exclusion of only low-frequency substructures. However, note that if there are no high-frequency substructures, then naturally Sort & Slice, feature-variance maximisation and entropy maximisation all simply slice away low-frequency substructures from the binary fingerprint and are thus all the same. In Remark 5.2, we have given a mathematical proof that Sort & Slice, entropy maximisation, and therefore also feature-variance maximisation are in fact strictly

equivalent under the assumption that no substructure appears in more than half of all training compounds. Since in common chemical data sets it is usually true that only very few substructures exist in more than half of all training compounds, Sort & Slice should be expected to closely approximate feature-variance maximisation (and entropy maximisation) in realistic settings while arguably being somewhat easier to describe and implement. It would be interesting to computationally compare the performance of substructure selection via feature-variance maximisation with Sort & Slice, to check whether in practice the exclusion of a small number of high-frequency substructures has a significant effect after all, or whether indeed both methods lead to a very similar level of performance as suggested by the theoretical arguments in Remark 5.2.

Another compelling feature selection technique that could be used for substructure pooling is given by *conditional* MIM as described by Fleuret [167]. Conditional MIM can be seen as a more sophisticated version of MIM that iteratively selects features that maximise the mutual information with the training label conditional on the information contained in any feature already picked. While MIM and conditional MIM both select features that are individually informative about the training label, conditional MIM is additionally designed to reduce redundancy by selecting features that also exhibit weak pairwise dependence and thus contain distinct pieces of information about the target variable. Conditional MIM can be a stronger choice than MIM in scenarios where there is a large informational overlap between features; on the other hand, if all features are perfectly independent, then MIM and conditional MIM become mathematically equivalent. A limitation of conditional MIM in the context of substructure pooling for ECFPs is its computational cost when it comes to the selection of large numbers of features; even the fast implementation of conditional MIM provided by Fleuret [167] may be slow to select hundreds or even thousands of substructures out of an even larger substructure pool. This might make the generation of vectorial ECFPs with usual lengths such as 1024 or 2048 bits impractical or even intractable when conditional MIM is used for substructure pooling. One way to address this problem is by instead using simple MIM as we did in our study; MIM can be interpreted as a natural simplification of conditional MIM that remains computationally feasible even in very high feature dimensions at the price of potentially leading to more feature redundancy. In a future study, it might still be worthwhile to explore the predictive abilities of low-dimensional vectorial ECFPs generated via substructure-pooling operators based on conditional MIM; it is conceivable that conditional MIM could generate a short yet effective and information-dense

ECFP-vectorisation whose performance may match or possibly even surpass the one of much longer hashed ECFPs.

Finally, note that the results for the SARS-CoV-2 main protease data set in Figure 5.4 that are based on a random data split are fully comparable to the QSAR-prediction results of the nine models that we investigated in our computational study in Chapter 3 (see Figure 3.9). In both studies, we used the same data set, the same data splitting technique (random split), and the same evaluation scheme (2-fold cross validation repeated with the same 3 random seeds across both studies). One difference is that, in the previous study from Chapter 3, we fully optimised the kNN, RF and MLP hyperparameters, but only experimented with a single type of ECFP (hashed 2048-bit ECFP4), while in the current study we kept the RF and MLP hyperparameters constant but explored a large chunk of the ECFP hyperparameter space. We can see that the strongest QSAR-predictor in Figure 3.9 is given by a hashed 2048-bit ECFP4 combined with a hyperparameter-optimised MLP which reaches an MAE slightly above 0.42. In contrast, Figure 5.4 shows that the same 2048-bit ECFP4 combined with an MLP based on our intuitively set hyperparameters from Table 5.2 reaches an MAE slightly below 0.42. This shows that in this setting our custom MLP-hyperparameter choice is essentially as performant as the computationally optimised MLP hyperparameters from our previous study. We further see in Figure 5.4 that Sort & Slice once again leads to slightly better performance than hashing for the 2048-bit ECFP4 combined with our custom MLP on a random split. This suggests that the predictive performance of the best QSAR-predictor for SARS-CoV-2 main protease binding affinity from our previous computational study from Chapter 3 could still have been slightly improved by vectorising the used 2048-bit ECFP4s via Sort & Slice instead of hashing.

5.4 Conclusions

We have introduced a general mathematical framework for the vectorisation of structural fingerprints via a formal operation we refer to as *substructure pooling*. For structural fingerprints, substructure pooling is the natural analogue to node feature vector pooling in modern GNN architectures. Unlike GNN pooling, substructure pooling remains largely unexplored and is almost always performed via the hashing of substructures into a vector of predefined length. Our proposed mathematical framework encompasses hash-based substructure pooling, but also pooling operations based on a diverse set of alternative techniques such as supervised and unsupervised

feature selection. Trainable permutation-invariant set functions operating on sets of substructure embeddings also fit into the given framework, and the future exploration of such advanced substructure-pooling methods might form an interesting opportunity for novel research. For example, in Section 6.2 below we introduce our idea of a novel trainable substructure-pooling technique based on a differentiable self-attention mechanism.

As part of our work, we have mathematically described and experimentally evaluated a method we refer to as *Sort & Slice* as an alternative to hashing for substructure pooling of ECFP-substructures. In a nutshell, Sort & Slice is based on first ranking all identified substructures in the training set according to their frequency of occurrence and then constructing a binary fingerprint that only indicates the presence or absence of the most frequent substructures. Sort & Slice is easy to implement and interpret and leads to increased predictive performance for supervised molecular machine learning tasks. Formally, Sort & Slice can be interpreted as a simple unsupervised feature selection scheme. We have given a mathematical proof that, under reasonable theoretical assumptions that are approximately valid for realistic data sets, Sort & Slice filters out all but the most informative substructures from an information-entropic perspective.

Due to its natural simplicity, variations of Sort & Slice might have already been used by other researchers in practical scenarios in the past. However, we are not aware of any occurrence of our version of Sort & Slice in a formal research paper. In particular, we are not aware of any rigorous experimental comparison of Sort & Slice and standard hash-based substructure pooling outside of this work. To the best of our knowledge, only one other version of Sort & Slice has been systematically explored [48]; however, unlike our version of Sort & Slice, this slightly different technique only allows limited control over the dimension of the vectorial fingerprint and was evaluated in a less general experimental setting.

In summary, our experiments show that Sort & Slice tends to generate higher (and sometimes substantially higher) downstream predictive performance than hashing for a variety of molecular property prediction tasks. This predictive advantage seems to exist across regression and classification data sets, balanced and imbalanced tasks, data splitting techniques, machine-learning models, and ECFP hyperparameters, and appears to increase with the expected number of bit collisions in the hashed ECFP. Perhaps surprisingly, Sort & Slice not only seems to outcompete hashing but also two more technically sophisticated supervised substructure selection schemes. This suggests that simply sorting substructures according to frequency of occurrence in the

training set and then discarding infrequent substructures is a relatively (and maybe unexpectedly) strong feature selection strategy. Based on the predictive advantage of Sort & Slice, its technical simplicity, and its ability to improve fingerprint interpretability by avoiding bit collisions, we recommend that it should canonically replace hashing as the standard substructure-pooling technique for supervised molecular machine learning.

Chapter 6

Future Directions

In this Chapter, we briefly describe two ideas we developed that could potentially form the seeds for two future research projects.

6.1 A Graph-Based Self-Supervised Learning Strategy to Make Classical Molecular Featurisations Trainable

In our study from Chapter 3, we showed that classical ECFPs consistently outperform trainable GINs at QSAR-prediction in a rigorous evaluation setting involving a robust series of random data splits and full hyperparameter-optimisation loops. This runs counter to the hopes that message-passing GNNs might be able to beat classical featurisations at molecular property prediction via their abilities to extract chemical knowledge directly from molecular graphs in a differentiable manner.

To enable graph-based featurisation methods to reach their full predictive potential and possibly break through the performance ceiling posed by ECFPs, we have developed a novel self-supervised learning strategy for GNNs based on predicting precomputed ECFPs from a potentially giant corpus of unlabelled molecular graphs. The pre-trained GNN can then be seamlessly combined with an ECFP-MLP model trained on a supervised molecular property prediction task such as QSAR-prediction. Our suggested learning strategy is partially motivated by recent observations that self-supervised pre-training followed by task-specific supervised fine-tuning can lead to impressive results in the image domain [168].

Our proposed scheme is divided into three steps that are visualised in Figure 6.1. Step 1 is based on pre-training a GNN to predict ECFPs from a large number of unlabelled molecular graphs, Step 2 represents training of a standard ECFP-MLP model

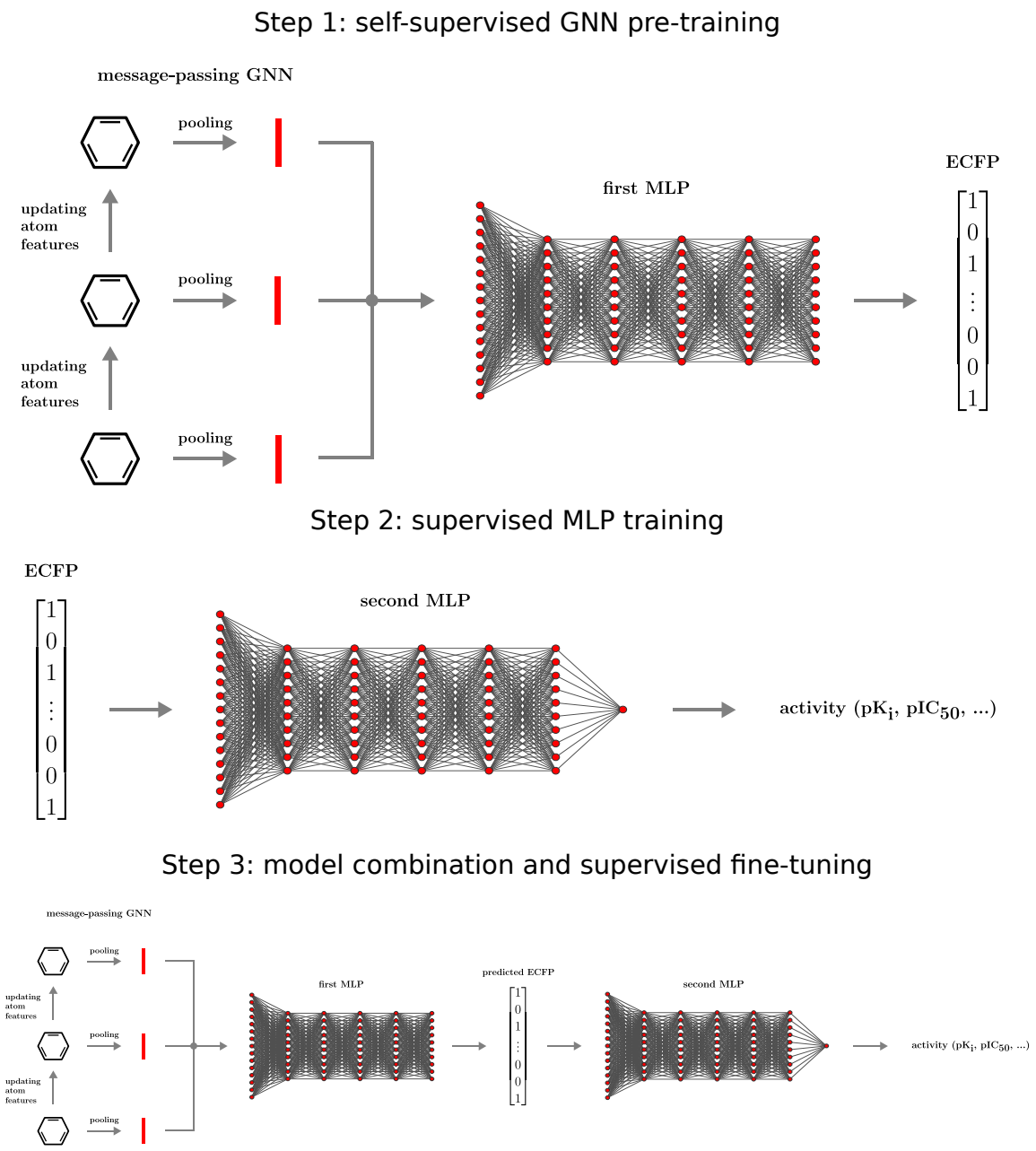

Figure 6.1: Step 1: Self-supervised pre-training of a graph neural network (GNN) to predict precomputed extended-connectivity fingerprints (ECFPs) from a large corpus of unlabelled molecular graphs. Step 2: Supervised training of a standard ECFP-based multilayer perceptron (MLP) on a given molecular property prediction task of interest. Step 3: Combination of both pre-trained models and fine-tuning of the resulting end-to-end model on the supervised task from Step 2.

on a supervised molecular property prediction task, and Step 3 involves plugging together both models from the two previous steps to create a graph-based predictor. The performance of this graph-based predictor must necessarily match the one of the standard ECFP-MLP model if the GNN-part has indeed managed to successfully learn to generate ECFPs from molecular graphs. Notably, in Step 3 the combined

end-to-end model can be further fine-tuned on the supervised task whereby the training signal then flows directly from the molecular graph to the training label. This final fine-tuning step might improve the performance of the combined model above that from the standard ECFP-MLP model and in this manner beat the state of the art. By using Sort & Slice ECFPs instead of hashed ECFPs, we can build on our already improved baseline.

During the supervised fine-tuning process, the learnt ECFP-representation generated by the GNN is expected to morph in a task-specific manner that benefits the predictive performance of the larger model. From this perspective, the proposed scheme can be interpreted as a way to make non-trainable classical precomputed molecular featurisations such as ECFPs differentiable and trainable. Note that the training strategy outlined in Figure 6.1 is not limited to ECFPs but can also be employed with PDVs, MACCS fingerprints or any other classical molecular featurisation method, as long as it can easily be generated for a large number of compounds.

The features extracted by the early layers of the pre-trained GNN that are close to the molecular graph can be seen as a novel type of neural fingerprint whose information content and predictive power could be explored. Finally, attempting to use message-passing GNNs to learn a differentiable mapping from molecular graphs to ECFPs might reveal their practical (in)abilities to correctly decipher chemical substructures; such insights could guide the way to further improvements of graph-based molecular featurisation methods in drug discovery.

6.2 Trainable Substructure Pooling via Differentiable Self-Attention

The four substructure-pooling methods investigated in our study in Chapter 5 are all either based on hashing or on some type of supervised or unsupervised feature selection strategy. However, it is also possible to devise more complex differentiable substructure-pooling operators based on trainable deep networks. To the best of our knowledge, this research avenue is currently unexplored.

We propose to investigate substructure pooling via *self-attention*. Self-attention is the key deep learning component in the famous transformer model that was introduced in the seminal paper from Vaswani et al. [169] and is still leading to state-of-the-art results in natural language processing. Given a set of input vectors, self-attention intuitively enables the updating of the representation of each input vector in a learnable and context-sensitive manner, i.e. in a manner that not only depends

on the vector itself but also on the learnt interactions between the vector and all the other vectors in the input set. In this sense, each element in the set of input vectors metaphorically *pays attention* to all other elements that are present, or from another perspective, the set of input vectors as a whole pays attention to *itself* by considering the interactions between all of its elements (hence the name *self*-attention).

To explore self-attention in the context of substructure pooling we once again consider the formal setting from Section 5.2.1. Let

$$\mathfrak{C} = \{\mathcal{C}_1, ..., \mathcal{C}_m\}$$

be a (potentially very large) set of m chemical substructures and let

$$P(\mathfrak{C}) = \{\mathfrak{A} \mid \mathfrak{A} \subseteq \mathfrak{C}\}$$

be its power set. Furthermore, let

$$\{\mathcal{C}_1, ..., \mathcal{C}_r\} \in P(\mathfrak{C})$$

be a representation of some input compound $\mathcal{R}$ as a set of r substructures in $\mathfrak{C}$. We imagine that $\mathcal{R}$ was transformed into $\{\mathcal{C}_1, ..., \mathcal{C}_r\}$ via some structural fingerprinting-method such as the ECFP or the MACCS-algorithm. We can now use some (injective) substructure embedding

$$\gamma : \mathfrak{C} \to \mathbb{R}^w$$

to generate a representation of $\mathcal{R}$ as a set of vectors:

$$\{\gamma(\mathcal{C}_1), ..., \gamma(\mathcal{C}_r)\} \subset \mathbb{R}^w.$$

The embedding γ could for instance be based on one-hot encoding of substructures or on physicochemical substructure descriptors. Our goal is to update the representations of the vectors in $\{\gamma(\mathcal{C}_1), ..., \gamma(\mathcal{C}_r)\}$ in a trainable way using self-attention.

In its simplest form, a classical self-attention layer [169, 170] is defined via three trainable weight matrices $W^Q, W^K \in \mathbb{R}^{w_q \times w}$ and $W^V \in \mathbb{R}^{w_v \times w}$. We now focus on a specific substructure representation $\gamma(\mathcal{C}_i) \in \mathbb{R}^w$ whose representation we want to update using these weight matrices. We start by generating a *query* vector

$$q_i := W^Q \gamma(\mathcal{C}_i) \in \mathbb{R}^{w_q},$$

a *key* vector

$$k_i := W^K \gamma(\mathcal{C}_i) \in \mathbb{R}^{w_q},$$

and a *value* vector

$$v_i \coloneqq W^V \gamma(\mathcal{C}_i) \in \mathbb{R}^{w_v}.$$

We proceed by computing weights

$$\alpha_{i,1}, ..., \alpha_{i,r} \in \mathbb{R}$$

by calculating the dot product between the query vector q_i and the key vectors $k_1, ..., k_r$ of all the other vectors in the input set:

$$\forall j \in \{1, ..., r\} : \quad \alpha_{i,j} \coloneqq q_i^T k_j \in \mathbb{R}.$$

Each quantity $\alpha_{i,j}$ can be intuitively interpreted as a measure for how much attention the vector $\gamma(\mathcal{C}_i)$ pays to the vector $\gamma(\mathcal{C}_j)$ during its updating process. The attention weights are usually further normalised via a nonlinear softmax activation function:

$$(\bar{\alpha}_{i,1}, ..., \bar{\alpha}_{i,r}) \coloneqq \text{softmax}(\alpha_{i,1}, ..., \alpha_{i,r}),$$

such that

$$\bar{\alpha}_{i,1}, ..., \bar{\alpha}_{i,r} > 0 \quad \text{and} \quad \sum_{j=1}^{r} \bar{\alpha}_{i,j} = 1.$$

Finally, the updated representation of $\gamma(\mathcal{C}_i)$ is given by a weighted sum of all value vectors:

$$\gamma(\mathcal{C}_i)_{\text{upt}} \coloneqq \sum_{j=1}^{r} \bar{\alpha}_{i,j} v_j \in \mathbb{R}^{w_v}.$$

The transformation

$$\mathbb{R}^w \supset \{\gamma(\mathcal{C}_1), ..., \gamma(\mathcal{C}_r)\} \mapsto \{\gamma(\mathcal{C}_1)_{\text{upt}}, ..., \gamma(\mathcal{C}_r)_{\text{upt}}\} \subset \mathbb{R}^{w_v}$$

is interpreted as the application of one self-attention layer to the set of input vectors $\{\gamma(\mathcal{C}_1), ..., \gamma(\mathcal{C}_r)\}$. This layer can be trained like any other deep learning component by adapting its defining weight matrices W^Q, W^K, W^V via some form of gradient descent. Several self-attention layers can naturally be stacked on top of each other to eventually generate a final vector-set representation of the input compound $\mathcal{R}$ in the form of iteratively updated substructure representations that encode structural and contextual information:

$$\{\gamma(\mathcal{C}_1)_{\text{final}}, ..., \gamma(\mathcal{C}_r)_{\text{final}}\} \subset \mathbb{R}^{w_{\text{final}}}.$$

Using a standard pooling function

$$\bigoplus : \{A \subset \mathbb{R}^{w_{\text{final}}} \mid A \text{ is finite}\} \to \mathbb{R}^l,$$

i.e. a permutation-invariant set function $\bigoplus$ such as summation, averaging or componentwise maximum, one can finally represent $\mathcal{R}$ as a single vector

$$\bigoplus\{\gamma(\mathcal{C}_1)_{\text{final}}, ..., \gamma(\mathcal{C}_r)_{\text{final}}\} \in \mathbb{R}^l$$

that can be fed into a standard multilayer perceptron for further processing. Note that iteratively updating substructural embeddings via stacked self-attention layers and then aggregating the final substructural representations via a permutation-invariant set function formally defines a (trainable, differentiable) substructure-pooling method:

$$\Psi : P(\mathfrak{C}) \to \mathbb{R}^l, \quad \Psi(\{\mathcal{C}_1, ..., \mathcal{C}_r\}) = \bigoplus\{\gamma(\mathcal{C}_1)_{\text{final}}, ..., \gamma(\mathcal{C}_r)_{\text{final}}\}.$$

The operator Ψ once again satisfies Definition 5.1 introduced in Chapter 5. This underlines the generality of our definition of substructure pooling which encompasses techniques such as hashing, unsupervised and supervised feature-selection, and the trainable updating of sets of substructural embeddings via modern deep learning architectures.

Self-attention-based substructure pooling on top of structural fingerprints has several properties that could potentially make it an interesting featurisation method for chemical prediction tasks. The self-attention mechanism should provide a useful inductive bias to learn substructural representations that are influenced by molecular context, i.e. by the presence or absence of other substructures. As a result, self-attention should explicitly support the learning of task-specific compound-level featurisations that depend not only on individually present substructures but also on their interactions. Note that this includes long-range interactions between substructures located at physically distant parts of the input compound. This might represent an important advantage over molecular featurisation via message-passing GNNs: the receptive field of GNNs is strictly local and thus does not allow for information flow between physically distant parts of an input compound during message-passing.

We conducted a literature search and were only able to identify one other work that has explored a technique similar to the one proposed in this section: Kim et al. [171] investigate a dual architecture that combines a GNN-branch operating on molecular graphs and a self-attention-branch operating on substructural embeddings. They pre-train their architecture to predict precomputed physicochemical descriptors using a large corpus of unlabelled compounds and obtain encouraging results when fine-tuning their model on a range of supervised molecular property prediction tasks. Further work of this kind could attempt to refine substructural self-attention mechanisms, for example by developing more powerful pre-training schemes or by studying the effects

of different types of initial substructural embeddings (such as one-hot embeddings versus physicochemical embeddings).

It might also be particularly interesting to explore the abilities of models involving self-attention-based substructure pooling to correctly predict *non-additivity* [172, 173]. In its most narrow form, non-additivity refers to a phenomenon observed in protein-ligand binding where the change of two substructures in a ligand results in much higher or lower binding affinity than would be expected from the respective additive contributions of the single changes alone. From a theoretical point of view, self-attention-based substructure pooling appears to be well-suited to detect such non-additivity events via its ability to learn distinct representations for a given substructure conditional on the presence or absence of other substructures.

Chapter 7

Conclusions and Further Thoughts

In this work, we have studied classical and graph-based molecular featurisation methods in a variety of important machine-learning scenarios for computational drug discovery. We have put a particular focus on the under-researched challenge of activity-cliff prediction which is of natural interest in compound optimisation and the elucidation of structure-activity relationships. We have (i) systematically explored the capabilities of physicochemical-descriptor vectors, extended-connectivity fingerprints and graph isomorphism networks for the prediction of quantitative structure-activity relationships, activity cliffs and potency directions, (ii) have designed a novel twin neural network model that can naturally learn to featurise compound pairs for the prediction of activity cliffs and potency directions, and (iii) have described an easily implementable method for the vectorisation of extended-connectivity fingerprints that robustly outperforms hashing at supervised molecular property prediction. We have also outlined two further research ideas in the area of molecular featurisation that can be seen as two distinct attempts to bring together the strengths of classical non-trainable featurisers and trainable deep learning components such as graph neural networks and self-attention.

Detailed conclusions from our main research projects can be found at the respective ends of Chapters 3 to 5. Overall, our investigations provide further evidence for the vital role that molecular featurisation plays for the performance of molecular machine learning tasks. In Chapter 3 we saw that switching from one featurisation technique to another can easily lead to substantial shifts in performance for both quantitative structure-activity relationship and activity-cliff prediction. The balanced activity-cliff classification performance of our twin neural network model from Chapter 4, compared to the imbalanced performance of the evaluated baseline quantitative structure-activity relationship predictors at the same task, supports the idea that it might generally pay off to naturally adapt the featuriser to the given

problem rather than trying to adapt the problem to a pre-existing featuriser. Our results from Chapter 5 show how even seemingly minor technical decisions such as the procedure chosen to vectorise a set of identified substructures can have a surprisingly large and consistent impact on the predictive accuracy of a structural fingerprint.

Extracting powerful features from arbitrary molecular structures is a difficult research challenge. Recent work has shown that in some cases self-supervised pre-training strategies on unlabelled molecular graphs can substantially boost the performance of graph neural networks, and graph isomorphism networks in particular [9, 21]. Considering these results and the accessibility of large databases with millions of unlabelled compounds, it may be worthwhile to continue exploring the limits of self-supervised pre-training of graph neural networks in the molecular domain, for example by investigating the pre-training strategy we propose in Section 6.1.

However, although in this work we have only experimented with graph isomorphism networks as prototypical examples of graph neural networks in the 1-WL class, we still hypothesise that differentiable graph-based message-passing, while relatively useful in certain contexts such as activity-cliff prediction, might not yet be the correct learning paradigm to truly and subtantially outperform the technically related and more traditional extended-connectivity fingerprints in the same way that convolutional neural networks have outperformed classical feature-engineering methods in computer vision. A fruitful area for future research might be the development of methods to overcome some of the technical shortcomings shared by both extended-connectivity fingerprints and graph neural networks such as a strictly local circular receptive field and limited theoretical expressivity. For example, a notable attempt in this direction has recently been made by Bouritsas et al. [90] who managed to increase the theoretical expressivity of message-passing graph neural networks via a technique based on subgraph isomorphism counting. Another interesting avenue has been explored by Ying et al. [174] who introduced a transformer-based graph-featuriser with a global receptive field that has achieved strong results across a variety of benchmarks.

It is also worth noting that molecular graphs are not entirely general but rather obey certain constraints dictated by the laws of chemistry; current message-passing graph neural networks, on the other hand, are highly general architectures that can essentially operate on any graph structure. One can speculate that it might be possible to somehow constrain the neural architecture of graph-based machine learning methods in a way that more directly leverages the chemical rules that govern the structure of molecules.

Finally, it might be useful to consider that one of the central limitations of current molecular featurisation methods may not be in the technical details of the featurisation itself, but in the information-content of the original molecular representation from which the features are extracted. In the vast majority of cases, molecular featurisations for supervised prediction tasks are derived either from molecular string representations such as SMILES strings or from molecular graphs. Both of these representations usually fully encode the chemical composition and two-dimensional connectivity structure of an input compound, along with simple 3D attributes such as tetrahedral R-S chirality and E-Z double bond geometry. While this appears comprehensive, it is possible to imagine that a real molecule might have other relevant physicochemical properties that cannot be easily derived from these pieces of information alone, such as more complex stereochemical features based on its ensemble of conformers or even quantum-chemical characteristics associated with its electronic structure. Developing novel featurisation methods adapted to more realistic molecular representations whose information-content strictly surpasses that of molecular graphs and SMILES strings may be a promising area for future research.

Summary of Research Contributions

Published Peer-Reviewed Research Papers

- Markus Dablander, Thierry Hanser, Renaud Lambiotte, and Garrett M. Morris. Exploring QSAR models for activity-cliff prediction. *Journal of Cheminformatics*, 15(1), 47, 2023. Link to paper.

- Julius Berner, Markus Dablander, and Philipp Grohs. Numerically solving parametric families of high-dimensional Kolmogorov partial differential equations via deep learning. *Advances in Neural Information Processing Systems*, 33, 16615-16627, 2020. Link to paper.

  My friend and colleague Julius Berner and I wrote this NeurIPS paper as shared first authors under the supervision of Prof. Philipp Grohs from the University of Vienna. This independent research project was conducted by us in parallel to my main doctoral studies.

Technical Reports from Industrial Study Groups

- Ruzanna Ab Razak, Markus Dablander, Noriszura Ismail, Jaeseong Jeong, Brady Metherall, Constantin Puiu, and Ann Smith. Tourism Forecasting and Environment. *Mathematics in Industry Reports*, 2023. (Submitted, not yet published.)

- Simone Appella, Anvarbek Atayev, Oliver Bond, Ben Collins, Markus Dablander, Nikolai Fadeev, Andrew Lacey, Piotr Morawiecki, Hilary Ockendon, Davide Polvara, Ellen Powell, Lorenzo Quintavalle Laval, Eddie Wilson, and Yang Zhou. Determining the conductance of networks created by randomly dispersed cylinders. *Mathematics in Industry Reports*, 2021. Link to ESGI report.

Conference Presentations

- Markus Dablander, Thierry Hanser, Renaud Lambiotte, and Garrett M. Morris. Exploring molecular machine learning models for activity-cliff prediction. Poster presentation at the 10th International Congress on Industrial and Applied Mathematics (ICIAM). In-person, Tokyo, 2023. Link to poster.

- Markus Dablander, Thierry Hanser, Renaud Lambiotte, and Garrett M. Morris. Siamese neural networks work for activity cliff prediction. Poster presentation at the 4th RSC-BMCS / RSC-CICAG Artificial Intelligence in Chemistry Symposium. Virtual, 2021. Link to poster. (See Awards and Prizes below.)

- Julius Berner, Markus Dablander, and Philipp Grohs. Numerically solving parametric families of high-dimensional Kolmogorov partial differential equations via deep learning. Poster presentation at the Thirty-fourth Conference on Neural Information Processing Systems (NeurIPS). Virtual, 2020. Link to poster.

Visited Industrial Study Groups

- ESGI 171 in Edinburgh, UK, in-person (2023).

- ESGI 156 in Ålesund, Norway, in-person (2022).

- ESGI 165 in Durham, UK, virtual (2021).

- ESGI 162 in Leeds, UK, virtual (2020).

Awards and Prizes

- Winner of InFoMM Doctoral Prize Scheme. Associated with InFoMM-funded post-doctoral research position at the Mathematical Institute, University of Oxford.

- Silver Medalist at the Smith Institute's 2021 TakeAIM research competition. For showcasing the potential impact of my computational research on activity cliffs via a short text accessible to non-experts.

- Winner of the Royal Society of Chemistry Prize for the Best Scientific Poster at the 4th RSC-BMCS / RSC-CICAG Artificial Intelligence in Chemistry Symposium.

Published Code

- Codebase to reproduce and extend the computational experiments from our published paper *Exploring QSAR Models for Activity-Cliff Prediction* [45]. Link to GitHub.

Bibliography

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[2] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[3] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Proceedings of the European Conference on Computer Vision*, pages 818–833, 2014.

[4] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[6] Tomaž Stepišnik, Blaž Škrlj, Jörg Wicker, and Dragi Kocev. A comprehensive comparison of molecular feature representations for use in predictive modeling. *Computers in Biology and Medicine*, 130:104197, 2021.

[7] Andreas Mayr, Günter Klambauer, Thomas Unterthiner, Marvin Steijaert, Jörg K Wegner, Hugo Ceulemans, Djork-Arné Clevert, and Sepp Hochreiter. Large-scale comparison of machine learning methods for drug target prediction on ChEMBL. *Chemical Science*, 9(24):5441–5451, 2018.

[8] Dejun Jiang, Zhenxing Wu, Chang-Yu Hsieh, Guangyong Chen, Ben Liao, Zhe Wang, Chao Shen, Dongsheng Cao, Jian Wu, and Tingjun Hou. Could

graph neural networks learn better molecular representation for drug discovery? A comparison study of descriptor-based and graph-based models. *Journal of Cheminformatics*, 13(1):1–23, 2021.

[9] Yuyang Wang, Jianren Wang, Zhonglin Cao, and Amir Barati Farimani. MolCLR: Molecular contrastive learning of representations via graph neural networks. *arXiv preprint arXiv:2102.10056*, 2021.

[10] Janosch Menke and Oliver Koch. Using domain-specific fingerprints generated through neural networks to enhance ligand-based virtual screening. *Journal of Chemical Information and Modeling*, 61(2):664–675, 2021.

[11] Seyone Chithrananda, Gabe Grand, and Bharath Ramsundar. ChemBERTa: Large-Scale self-supervised pretraining for molecular property prediction. *arXiv preprint arXiv:2010.09885*, 2020.

[12] María Virginia Sabando, Ignacio Ponzoni, Evangelos E. Milios, and Axel J. Soto. Using molecular embeddings in QSAR modeling: Does it make a difference? *arXiv preprint arXiv:2104.02604*, 2021.

[13] Robin Winter, Floriane Montanari, Frank Noé, and Djork-Arné Clevert. Learning continuous and data-driven molecular descriptors by translating equivalent chemical representations. *Chemical Science*, 10(6):1692–1701, 2019.

[14] Roberto Todeschini and Viviana Consonni. *Handbook of Molecular Descriptors*. John Wiley & Sons, 2008.

[15] Dávid Bajusz, Anita Rácz, and Károly Héberger. Fingerprints, and other molecular descriptions for database analysis and searching. 2017.

[16] David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of Chemical Information and Modeling*, 50(5):742–754, 2010.

[17] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pages 1263–1272. PMLR, 2017.

[18] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[19] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: Moving beyond fingerprints. *Journal of Computer-Aided Molecular Design*, 30(8):595–608, 2016.

[20] David K. Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P. Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in Neural Information Processing Systems*, pages 2224–2232, 2015.

[21] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*, 2019.

[22] Kevin Yang, Kyle Swanson, Wengong Jin, Connor Coley, Philipp Eiden, Hua Gao, Angel Guzman-Perez, Timothy Hopper, Brian Kelley, Miriam Mathea, et al. Analyzing learned molecular representations for property prediction. *Journal of Chemical Information and Modeling*, 59(8):3370–3388, 2019.

[23] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S. Yu Philip. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[24] Oliver Wieder, Stefan Kohlbacher, Mélaine Kuenemann, Arthur Garon, Pierre Ducrot, Thomas Seidel, and Thierry Langer. A compact review of molecular property prediction with graph neural networks. *Drug Discovery Today: Technologies*, 2020.

[25] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

[26] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. In *Advances in Neural Information Processing Systems*, pages 4502–4510, 2016.

[27] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852, 2016.

[28] Ke Liu, Xiangyan Sun, Lei Jia, Jun Ma, Haoming Xing, Junqiu Wu, Hua Gao, Yax Sun, Florian Boulnois, and Jie Fan. Chemi-Net: A molecular graph convolutional network for accurate drug property prediction. *International Journal of Molecular Sciences*, 20(14):3389, 2019.

[29] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[30] Meng Liu, Hongyang Gao, and Shuiwang Ji. Towards deeper graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 338–348, 2020.

[31] Nicolò Navarin, Dinh Van Tran, and Alessandro Sperduti. Universal readout for graph convolutional neural networks. In *Proceedings of International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2019.

[32] Jiahua Rao, Shuangjia Zheng, Yutong Lu, and Yuedong Yang. Quantitative evaluation of explainable graph neural networks for molecular property prediction. *Patterns*, page 100628, 2022.

[33] Patrick Hop, Brandon Allgood, and Jessen Yu. Geometric deep learning autonomously learns chemical features that outperform those engineered by domain experts. *Molecular Pharmaceutics*, 15(10):4371–4377, 2018.

[34] Chao Shang, Qinqing Liu, Ko-Shin Chen, Jiangwen Sun, Jin Lu, Jinfeng Yi, and Jinbo Bi. Edge attention-based multi-relational graph convolutional networks. *arXiv preprint arXiv: 1802.04944*, 2018.

[35] Junying Li, Deng Cai, and Xiaofei He. Learning graph-level representation for drug discovery. *arXiv preprint arXiv:1709.03741*, 2017.

[36] Zhaoping Xiong, Dingyan Wang, Xiaohong Liu, Feisheng Zhong, Xiaozhe Wan, Xutong Li, Zhaojun Li, Xiaomin Luo, Kaixian Chen, Hualiang Jiang, et al. Pushing the boundaries of molecular representation for drug discovery with the graph attention mechanism. *Journal of Medicinal Chemistry*, 63(16):8749–8760, 2019.

[37] Derek van Tilborg, Alisa Alenicheva, and Francesca Grisoni. Exposing the limitations of molecular machine learning with activity cliffs. *ChemRxiv*, 2022. doi: 10.26434/chemrxiv-2022-mfq52.

[38] Carlo Silipo and Antonio Vittoria. QSAR, rational approaches to the design of bioactive compounds. In *Proceedings of European Symposium on Quantitative Structure-Activity Relationships*. Distributors for the US and Canada, Elsevier Science, 1991.

[39] Gerald M. Maggiora. On outliers and activity cliffs: Why QSAR often disappoints. *Journal of Chemical Information and Modeling*, 46(4):1535–1535, 2006.

[40] Robert P. Sheridan, Prabha Karnachi, Matthew Tudor, Yuting Xu, Andy Liaw, Falgun Shah, Alan C. Cheng, Elizabeth Joshi, Meir Glick, and Juan Alvarez. Experimental error, kurtosis, activity cliffs, and methodology: What limits the predictivity of quantitative structure–activity relationship models. *Journal of Chemical Information and Modeling*, 60(4):1969–1982, 2020.

[41] Maykel Cruz-Monteagudo, José L. Medina-Franco, Yunierkis Pérez-Castillo, Orazio Nicolotti, M. Natália D. S. Cordeiro, and Fernanda Borges. Activity cliffs in drug discovery: Dr Jekyll or Mr Hyde? *Drug Discovery Today*, 19(8): 1069–1080, 2014.

[42] Dagmar Stumpfe, Ye Hu, Dilyana Dimova, and Jürgen Bajorath. Recent progress in understanding activity cliffs and their utility in medicinal chemistry: miniperspective. *Journal of Medicinal Chemistry*, 57(1):18–28, 2014.

[43] Dagmar Stumpfe, Huabin Hu, and Jürgen Bajorath. Evolving concept of activity cliffs. *ACS Omega*, 4(11):14360–14368, 2019.

[44] Dagmar Stumpfe, Huabin Hu, and Jürgen Bajorath. Advances in exploring activity cliffs. *Journal of Computer-Aided Molecular Design*, 34(9):929–942, 2020.

[45] Markus Dablander, Thierry Hanser, Renaud Lambiotte, and Garrett M. Morris. Exploring QSAR models for activity-cliff prediction. *Journal of Cheminformatics*, 15(1):47, 2023.

[46] Markus Dablander, Thierry Hanser, Renaud Lambiotte, and Garrett M. Morris. Exploring molecular machine learning models for activity-cliff prediction. Poster presentation at the 10th International Congress on Industrial and Applied Mathematics (ICIAM). In-person, Tokyo, 2023. URL `https://www.researchgate.net/publication/373976767_Exploring_Molecular_Machine_Learning_Models_for_Activity-Cliff_Prediction`.

[47] Markus Dablander, Thierry Hanser, Renaud Lambiotte, and Garrett M. Morris. Siamese neural networks work for activity cliff prediction. Poster presentation at the 4th RSC-BMCS / RSC-CICAG Artificial Intelligence in Chemistry Symposium. Virtual, 2021.

[48] Thomas MacDougall. (2022) Reduced collision fingerprints and pairwise molecular comparisons for explainable property prediction using deep learning. M.Sc. thesis. Université de Montréal. URL `https://doi.org/1866/26533`. Accessed on 05.10.2023.

[49] Garrett B. Goh, Charles Siegel, Abhinav Vishnu, Nathan O. Hodas, and Nathan Baker. Chemception: A deep neural network with minimal chemistry knowledge matches the performance of expert-developed QSAR/QSPR models. *arXiv preprint arXiv:1706.06689*, 2017.

[50] Atsushi Yoshimori. Prediction of molecular properties using molecular topographic map. *Molecules*, 26(15):4475, 2021.

[51] Javed Iqbal, Martin Vogt, and Jürgen Bajorath. Prediction of activity cliffs on the basis of images using convolutional neural networks. *Journal of Computer-Aided Molecular Design*, 2021.

[52] Joshua Schrier. Can one hear the shape of a molecule (from its Coulomb matrix eigenvalues)? *Journal of Chemical Information and Modeling*, 60(8):3804–3811, 2020.

[53] Gengmo Zhou, Zhifeng Gao, Qiankun Ding, Hang Zheng, Hongteng Xu, Zhewei Wei, Linfeng Zhang, and Guolin Ke. Uni-Mol: A universal 3D molecular representation learning framework. *ChemRxiv*, 2022. doi: 10.26434/chemrxiv-2022-jjm0j-v2.

[54] Norman Biggs, E. Keith Lloyd, and Robin J. Wilson. *Graph Theory, 1736-1936*. Oxford University Press, 1986.

[55] Agnieszka Pocha, Tomasz Danel, and Łukasz Maziarka. Comparison of atom representations in graph neural networks for molecular property prediction. *arXiv preprint arXiv:2012.04444*, 2020.

[56] David Weininger. SMILES, a chemical language and information system. *Journal of Chemical Information and Computer Sciences*, 28(1):31–36, 1988.

[57] David Weininger, Arthur Weininger, and Joseph L. Weininger. Algorithm for generation of unique SMILES notation. *Journal of Chemical Information and Computer Sciences*, 29(2):97–101, 1989.

[58] David Weininger. Graphical depiction of chemical structures. *Journal of Chemical Information and Computer Sciences*, 30(3):237–243, 1990.

[59] Fdardel (original) and DMacks (edited). Image: Deriving the SMILES representation of a chemical molecule, Shown example: ciprofloxacin, a fluoroquinolone antibiotic. URL `https://commons.wikimedia.org/wiki/File:SMILES.png`. CC BY-SA 3.0 License, via Wikimedia Commons. Accessed on 14.11.2022.

[60] Stephen Heller, Alan McNaught, Stephen Stein, Dmitrii Tchekhovskoi, and Igor Pletnev. InChI - the worldwide chemical structure identifier standard. *Journal of Cheminformatics*, 5(1):1–9, 2013.

[61] Rafael Gómez-Bombarelli, Jennifer N. Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4(2):268–276, 2018.

[62] Mario Krenn, Florian Häse, Akshat-Kumar Nigam, Pascal Friederich, and Alan Aspuru-Guzik. Self-referencing embedded strings (SELFIES): A 100% robust molecular string representation. *Machine Learning: Science and Technology*, 1 (4):045024, 2020.

[63] Noel O'Boyle and Andrew Dalke. DeepSMILES: An adaptation of SMILES for use in machine-learning of chemical structures. *ChemRxiv*, 2018. doi: 10.26434/chemrxiv.7097960.v1.

[64] Tomasz Puzyn, Jerzy Leszczynski, and Mark T. Cronin. *Recent advances in QSAR studies: Methods and applications*, volume 8. Springer Science & Business Media, 2010.

[65] Huixiao Hong, Qian Xie, Weigong Ge, Feng Qian, Hong Fang, Leming Shi, Zhenqiang Su, Roger Perkins, and Weida Tong. Mold^2, molecular descriptors from 2D structures for chemoinformatics and toxicoinformatics. *Journal of Chemical Information and Modeling*, 48(7):1337–1344, 2008.

[66] Ling Xue and Jürgen Bajorath. Molecular descriptors in chemoinformatics, computational combinatorial chemistry, and virtual screening. *Combinatorial Chemistry & High Throughput Screening*, 3(5):363–372, 2000.

[67] Viviana Consonni and Roberto Todeschini. Molecular descriptors. In *Recent Advances in QSAR Studies*, pages 29–102. Springer, 2010.

[68] Christopher A. Lipinski, Franco Lombardo, Beryl W. Dominy, and Paul J. Feeney. Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. *Advanced Drug Delivery Reviews*, 23(1-3):3–25, 1997.

[69] Scott A. Wildman and Gordon M. Crippen. Prediction of physicochemical parameters by atomic contributions. *Journal of Chemical Information and Computer Sciences*, 39(5):868–873, 1999.

[70] Greg Landrum. `RDKit`: Open-source cheminformatics. 2006. URL `http://www.rdkit.org`. Accessed on 05.10.2023.

[71] Alexandru T. Balaban. Highly discriminating distance-based topological index. *Chemical Physics Letters*, 89(5):399–404, 1982.

[72] Benedek Fabian, Thomas Edlich, Héléna Gaspar, Marwin Segler, Joshua Meyers, Marco Fiscato, and Mohamed Ahmed. Molecular representation learning with language models and domain-relevant auxiliary tasks. *arXiv preprint arXiv:2011.13230*, 2020.

[73] Harry L. Morgan. The generation of a unique machine description for chemical structures—A technique developed at chemical abstracts service. *Journal of Chemical Documentation*, 5(2):107–113, 1965.

[74] Joseph L. Durant, Burton A. Leland, Douglas R. Henry, and James G. Nourse. Reoptimization of MDL keys for use in drug discovery. *Journal of Chemical Information and Computer Sciences*, 42(6):1273–1280, 2002.

[75] Online description of PubChem substructure fingerprints. URL `https://ftp.ncbi.nlm.nih.gov/pubchem/specifications/pubchem_fingerprints.pdf`. Accessed on 01.10.2023.

[76] Lianyi Han, Yanli Wang, and Stephen H. Bryant. Developing and validating predictive decision tree models from mining chemical structural fingerprints and high-throughput screening data in PubChem. *BMC Bioinformatics*, 9:1–8, 2008.

[77] Online description of Daylight substructure fingerprints. URL `https://www.daylight.com/dayhtml/doc/theory/theory.finger.html`. Accessed on 01.10.2023.

[78] Sereina Riniker and Greg Landrum. Open-source platform to benchmark fingerprints for ligand-based virtual screening. *Journal of Cheminformatics*, 5(1): 26, 2013.

[79] Henry E. Webel, Talia B. Kimber, Silke Radetzki, Martin Neuenschwander, Marc Nazaré, and Andrea Volkamer. Revealing cytotoxic substructures in molecules using deep learning. *Journal of Computer-aided Molecular Design*, 34(7):731–746, 2020.

[80] David Rogers, Robert D. Brown, and Mathew Hahn. Using extended-connectivity fingerprints with Laplacian-modified Bayesian analysis in high-throughput screening follow-up. *Journal of Biomolecular Screening*, 10(7):682–686, 2005.

[81] Jonathan Alvarsson, Martin Eklund, Ola Engkvist, Ola Spjuth, Lars Carlsson, Jarl E. S. Wikberg, and Tobias Noeske. Ligand-based target prediction with signature fingerprints. *Journal of Chemical Information and Modeling*, 54(10): 2647–2653, 2014.

[82] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: Going beyond Euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[83] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

[84] Vijay Prakash Dwivedi, Chaitanya K. Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020.

[85] Mohammadamin Tavakoli and Pierre Baldi. Continuous representation of molecules using graph variational autoencoder. *arXiv preprint arXiv:2004.08152*, 2020.

[86] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in Neural Information Processing Systems*, 33:22118–22133, 2020.

[87] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[88] Byung-Hoon Kim and Jong Chul Ye. Understanding graph isomorphism network for rs-fMRI functional connectivity analysis. *Frontiers in Neuroscience*, page 630, 2020.

[89] Boris Weisfeiler and Andrei Lehman. The reduction of a graph to canonical form and the algebra which appears therein. *NTI, Series*, 2(9):12–16, 1968.

[90] Giorgos Bouritsas, Fabrizio Frasca, Stefanos P. Zafeiriou, and Michael Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[91] Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and Leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4602–4609, 2019.

[92] Wei Jin, Xiaorui Liu, Yao Ma, Charu Aggarwal, and Jiliang Tang. Feature overcorrelation in deep graph neural networks: A new perspective. *arXiv preprint arXiv:2206.07743*, 2022.

[93] Wentao Zhang, Zeang Sheng, Yuezihan Jiang, Yikuan Xia, Jun Gao, Zhi Yang, and Bin Cui. Evaluating deep graph neural networks. *arXiv preprint arXiv:2108.00955*, 2021.

[94] Jonathan Godwin, Michael Schaarschmidt, Alexander L. Gaunt, Alvaro Sanchez-Gonzalez, Yulia Rubanova, Petar Veličković, James Kirkpatrick, and

Peter Battaglia. Simple GNN regularisation for 3D molecular property prediction and beyond. In *International Conference on Learning Representations*, 2021.

[95] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3438–3445, 2020.

[96] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with `PyTorch Geometric`. *arXiv preprint arXiv:1903.02428*, 2019.

[97] Alexander Golbraikh, Eugene Muratov, Denis Fourches, and Alexander Tropsha. Data set modelability by QSAR. *Journal of Chemical Information and Modeling*, 54(1):1–4, 2014.

[98] Jr. Leadley et al. Coagulation factor Xa inhibition: Biological background and rationale. *Current Topics in Medicinal Chemistry*, 1(2):151–159, 2001.

[99] Martin Vogt, Yun Huang, and Jürgen Bajorath. From activity cliffs to activity ridges: Informative data structures for SAR analysis. *Journal of Chemical Information and Modeling*, 51(8):1848–1856, 2011.

[100] Dilyana Dimova, Dagmar Stumpfe, Ye Hu, and Jürgen Bajorath. Activity cliff clusters as a source of structure–activity relationship information. *Expert Opinion on Drug Discovery*, 10(5):441–447, 2015.

[101] José L. Medina-Franco. Activity cliffs: Facts or artifacts? *Chemical Biology & Drug Design*, 81(5):553–556, 2013.

[102] Maykel Cruz-Monteagudo, José L. Medina-Franco, Yunier Perera-Sardiña, Fernanda Borges, Eduardo Tejera, Cesar Paz-y Mino, Yunierkis Pérez-Castillo, Aminael Sánchez-Rodríguez, Zuleidys Contreras-Posada, Natália DS Cordeiro, et al. Probing the hypothesis of SAR continuity restoration by the removal of activity cliffs generators in QSAR. *Current Pharmaceutical Design*, 22(33): 5043–5056, 2016.

[103] David A. Winkler and Tu C. Le. Performance of deep and shallow neural networks, the universal approximation theorem, activity cliffs, and QSAR. *Molecular Informatics*, 36(1-2):1600118, 2017.

[104] Kathrin Heikamp, Xiaoying Hu, Aixia Yan, and Jürgen Bajorath. Prediction of activity cliffs using support vector machines. *Journal of Chemical Information and Modeling*, 52(9):2354–2365, 2012.

[105] Shunsuke Tamura, Tomoyuki Miyao, and Kimito Funatsu. Ligand-based activity cliff prediction models with applicability domain. *Molecular Informatics*, 39 (12):2000103, 2020.

[106] Antonio De la Vega de León and Jürgen Bajorath. Prediction of compound potency changes in matched molecular pairs using support vector regression. *Journal of Chemical Information and Modeling*, 54(10):2654–2663, 2014.

[107] Jeremy M. Beck and Clayton Springer. Quantitative structure-activity relationship models of chemical transformations from matched pairs analyses. *Journal of Chemical Information and Modeling*, 54(4):1226–1234, 2014.

[108] Vigneshwaran Namasivayam and Jürgen Bajorath. Searching for coordinated activity cliffs using particle swarm optimization. *Journal of Chemical Information and Modeling*, 52(4):927–934, 2012.

[109] Vigneshwaran Namasivayam, Preeti Iyer, and Jürgen Bajorath. Prediction of individual compounds forming activity cliffs using emerging chemical patterns. *Journal of Chemical Information and Modeling*, 53(12):3131–3139, 2013.

[110] Jarmila Husby, Giovanni Bottegoni, Irina Kufareva, Ruben Abagyan, and Andrea Cavalli. Structure-based predictions of activity cliffs. *Journal of Chemical Information and Modeling*, 55(5):1062–1076, 2015.

[111] Dragos Horvath, Gilles Marcou, Alexandre Varnek, Shilva Kayastha, Antonio de la Vega de León, and Jürgen Bajorath. Prediction of activity cliffs using condensed graphs of reaction representations. *Journal of Chemical Information and Modeling*, 56(9):1631–1640, 2016.

[112] Laura Pérez-Benito, Nil Casajuana-Martin, Mireia Jiménez-Rosés, Herman van Vlijmen, and Gary Tresadern. Predicting activity cliffs with free-energy perturbation. *Journal of Chemical Theory and Computation*, 15(3):1884–1895, 2019.

[113] Yasunobu Asawa, Atsushi Yoshimori, Jürgen Bajorath, and Hiroyuki Nakamura. Prediction of an MMP-1 inhibitor activity cliff using the SAR matrix approach and its experimental validation. *Scientific Reports*, 10(1):14710, 2020.

[114] Mohammad Reza Keyvanpour, Mehrnoush Barani Shirzad, and Farhaneh Moradi. PCAC: A new method for predicting compounds with activity cliff property in QSAR approach. *International Journal of Information Technology*, 13(6):2431–2437, 2021.

[115] Junhui Park, Gaeun Sung, SeungHyun Lee, SeungHo Kang, and ChunKyun Park. ACGCN: Graph convolutional networks for activity cliff prediction between matched molecular pairs. *Journal of Chemical Information and Modeling*, 2022.

[116] Hengwei Chen, Martin Vogt, and Jürgen Bajorath. DeepAC—Conditional transformer-based chemical language model for the prediction of activity cliffs formed by bioactive compounds. *Digital Discovery*, 2022.

[117] Frank Hoonakker, Nicolas Lachiche, Alexandre Varnek, and Alain Wagner. Condensed graph of reaction: Considering a chemical reaction as one single pseudo molecule. *International Journal on Artificial Intelligence Tools*, 20(2):253–270, 2011.

[118] Philippe Jauffret, Thierry Hanser, Christian Tonnelier, and Gérard Kaufmann. Machine learning of generic reactions: 1. Scope of the project; The GRAMS program. *Tetrahedron Computer Methodology*, 3(6):323–333, 1990.

[119] Philip Seeman. Dopamine receptors and the dopamine hypothesis of schizophrenia. *Synapse*, 1(2):133–152, 1987.

[120] Sven Ullrich and Christoph Nitsche. The SARS-CoV-2 main protease as drug target. *Bioorganic & Medicinal Chemistry Letters*, 30(17):127377, 2020.

[121] Helen M. Berman, John Westbrook, Zukang Feng, Gary Gilliland, Talapady N. Bhat, Helge Weissig, Ilya N. Shindyalov, and Philip E. Bourne. The Protein Data Bank. *Nucleic Acids Research*, 28(1):235–242, 2000. URL `https://www.rcsb.org`.

[122] Tiqing Liu, Yuhmei Lin, Xin Wen, Robert N. Jorissen, and Michael K. Gilson. BindingDB: A web-accessible database of experimentally determined protein-ligand binding affinities. *Nucleic Acids Research*, 35:D198–D201, 2007.

[123] Hagit Achdout, Anthony Aimon, Elad Bar-David, Haim Barr, Amir Ben-Shmuel, James Bennett, Melissa L. Bobby, Juliane Brun, BVNBS Sarma, Mark

Calmiano, et al. COVID moonshot: Open science discovery of SARS-CoV-2 main protease inhibitors by combining crowdsourcing, high-throughput experiments, computational simulations, and machine learning. *BioRxiv*, 2020.

[124] A. Patrícia Bento, Anne Hersey, Eloy Félix, Greg Landrum, Anna Gaulton, Francis Atkinson, Louisa J. Bellis, Marleen de Veij, and Andrew R. Leach. An open source chemical structure curation pipeline using `RDKit`. *Journal of Cheminformatics*, 12(1):1–16, 2020.

[125] Peter W. Kenny and Jens Sadowski. Structure modification in chemical databases. *Chemoinformatics in Drug Discovery*, 23:271–285, 2005.

[126] Ye Hu and Jürgen Bajorath. Extending the activity cliff concept: Structural categorization of activity cliffs and systematic identification of different types of cliffs in the ChEMBL database. *Journal of Chemical Information and Modeling*, 52(7):1806–1811, 2012.

[127] Andrew Dalke, Jerome Hert, and Christian Kramer. `mmpdb`: An open-source matched molecular pair platform for large multiproperty data sets. *Journal of Chemical Information and Modeling*, 58(5):902–910, 2018.

[128] Jürgen Bajorath. Exploring activity cliffs from a chemoinformatics perspective. *Molecular Informatics*, 33(6-7):438–442, 2014.

[129] Xiaoying Hu, Ye Hu, Martin Vogt, Dagmar Stumpfe, and Jürgen Bajorath. MMP-cliffs: Systematic identification of activity cliffs on the basis of matched molecular pairs. *Journal of Chemical Information and Modeling*, 52(5):1138–1145, 2012.

[130] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in `Python`. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[131] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. `PyTorch`: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 2019.

[132] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2623–2631, 2019.

[133] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of Machine Learning Research*, pages 448–456, 2015.

[134] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[135] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[136] Davide Chicco. Siamese neural networks: An overview. *Artificial Neural Networks*, pages 73–94, 2021.

[137] Jane Bromley, James W. Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. Signature verification using a "Siamese" time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(04):669–688, 1993.

[138] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, volume 2. Lille, 2015.

[139] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1708, 2014.

[140] Devendra Singh Dhami, Gautam Kunapuli, David Page, and Sriraam Natarajan. Predicting drug-drug interactions from molecular structure images. In *Proceedings of AAAI Fall Symposium on AI for Social Good*, 2019.

[141] Yi Zhong, Xueyu Chen, Yu Zhao, Xiaoming Chen, Tingfang Gao, and Zuquan Weng. Graph-augmented convolutional networks on drug-drug interactions prediction. *arXiv preprint arXiv:1912.03702*, 2019.

[142] Kyriakos Schwarz, Ahmed Allam, Nicolas Andres Perez Gonzalez, and Michael Krauthammer. AttentionDDI: Siamese attention-based deep learning method for drug-drug interaction predictions. *arXiv preprint arXiv:2012.13248*, 2020.

[143] Luis Torres, Nelson Monteiro, Josè Oliveira, Joel Arrais, and Bernardete Ribeiro. Exploring a Siamese neural network architecture for one-shot drug discovery. In *Proceedings of 20th International Conference on Bioinformatics and Bioengineering (BIBE)*, pages 168–175, 2020.

[144] Igor I. Baskin, Vladimir A. Palyulin, and Nikolai S. Zefirov. Neural networks in building QSAR models. In *Artificial Neural Networks*, pages 133–154. Springer, 2006.

[145] Paulino A. Alvarez and Jaime Pahissa. QT alterations in psychopharmacology: Proven candidates and suspects. *Current Drug Safety*, 5(1):97–104, 2010.

[146] Muhao Chen, Chelsea J-T Ju, Guangyu Zhou, Xuelu Chen, Tianran Zhang, Kai-Wei Chang, Carlo Zaniolo, and Wei Wang. Multifaceted protein-protein interaction prediction based on Siamese residual RCNN. *Bioinformatics*, 35 (14):i305–i314, 2019.

[147] Daniel Fernández-Llaneza, Silas Ulander, Dea Gogishvili, Eva Nittinger, Hongtao Zhao, and Christian Tyrchan. Siamese recurrent neural network with a self-attention mechanism for bioactivity prediction. *ACS Omega*, 6(16):11086–11094, 2021.

[148] Minji Jeon, Donghyeon Park, Jinhyuk Lee, Hwisang Jeon, Miyoung Ko, Sunkyu Kim, Yonghwa Choi, Aik-Choon Tan, and Jaewoo Kang. ReSimNet: Drug response similarity prediction using Siamese neural networks. *Bioinformatics*, 35(24):5249–5256, 2019.

[149] Nicholas Roberts, Poornav S. Purushothama, Vishal T. Vasudevan, Siddarth Ravichandran, Chen Zhang, William H. Gerwick, and Garrison W. Cottrell. Using deep Siamese neural networks to speed up natural products research. *ICLR 2019 Conference Blind Submission*, 2018.

[150] Esmaeil Nourani, Ehsaneddin Asgari, Alice C. McHardy, and Mohammad R. K. Mofrad. TripletProt: Deep representation learning of proteins based on Siamese networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 19(6):3744–3753, 2021.

[151] Kyle Yingkai Gao, Achille Fokoue, Heng Luo, Arun Iyengar, Sanjoy Dey, and Ping Zhang. Interpretable drug-target prediction using deep neural representation. In *Proceedings of International Joint Conference on Artificial Intelligence*, volume 2018, pages 3371–3377, 2018.

[152] Cătălina Cangea, Petar Veličković, Nikola Jovanović, Thomas Kipf, and Pietro Liò. Towards sparse hierarchical graph classifiers. *arXiv preprint arXiv:1811.01287*, 2018.

[153] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-attention graph pooling. In *International Conference on Machine Learning*, pages 3734–3743. PMLR, 2019.

[154] Ekagra Ranjan, Soumya Sanyal, and Partha Talukdar. Asap: Adaptive structure aware pooling for learning hierarchical graph representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5470–5477, 2020.

[155] Zheng Ma, Junyu Xuan, Yu Guang Wang, Ming Li, and Pietro Liò. Path integral based convolution and pooling for graph neural networks. *Advances in Neural Information Processing Systems*, 33:16421–16433, 2020.

[156] Daniel Probst and Jean-Louis Reymond. A probabilistic molecular fingerprint for big data settings. *Journal of Cheminformatics*, 10:1–12, 2018.

[157] Martin Gütlein and Stefan Kramer. Filtered circular fingerprints improve either prediction or runtime performance while retaining interpretability. *Journal of Cheminformatics*, 8(1):1–16, 2016.

[158] Claude E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[159] Thomas M. Cover, Joy A. Thomas, et al. Entropy, relative entropy and mutual information. *Elements of Information Theory*, 2(1):12–13, 1991.

[160] SMARTS Theory Manual. *Daylight Chemical Information Systems*. URL `https://www.daylight.com/dayhtml/doc/theory/theory.smarts.html`. Accessed on 05.10.2023.

[161] Karl Pearson. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London,*

Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 50(302): 157–175, 1900.

[162] Viet-Khoa Tran-Nguyen, Célien Jacquemard, and Didier Rognan. LIT-PCBA: An unbiased data set for machine learning and virtual screening. *Journal of Chemical Information and Modeling*, 60(9):4263–4273, 2020.

[163] Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay Pande. MoleculeNet: A benchmark for molecular machine learning. *Chemical Science*, 9(2):513–530, 2018.

[164] Murat Cihan Sorkun, Abhishek Khetan, and Süleyman Er. AqSolDB, a curated reference set of aqueous solubility and 2D descriptors for a diverse set of compounds. *Scientific Data*, 6(1):143, 2019.

[165] Katja Hansen, Sebastian Mika, Timon Schroeter, Andreas Sutter, Antonius Ter Laak, Thomas Steger-Hartmann, Nikolaus Heinrich, and Klaus-Robert Muller. Benchmark data set for *in silico* prediction of Ames mutagenicity. *Journal of Chemical Information and Modeling*, 49(9):2077–2081, 2009.

[166] Guy W. Bemis and Mark A. Murcko. The properties of known drugs: Molecular frameworks. *Journal of Medicinal Chemistry*, 39(15):2887–2893, 1996.

[167] François Fleuret. Fast binary feature selection with conditional mutual information. *Journal of Machine Learning Research*, 5(9), 2004.

[168] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E. Hinton. Big self-supervised models are strong semi-supervised learners. *Advances in Neural Information Processing Systems*, 33:22243–22255, 2020.

[169] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

[170] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pages 3744–3753. PMLR, 2019.

[171] Jiye Kim, Seungbeom Lee, Dongwoo Kim, Sungsoo Ahn, and Jaesik Park. Substructure-atom cross attention for molecular representation learning. *arXiv preprint arXiv:2210.08243*, 2022.

[172] Dea Gogishvili, Eva Nittinger, Christian Margreitter, and Christian Tyrchan. Nonadditivity in public and inhouse data: Implications for drug design. *Journal of Cheminformatics*, 13:1–18, 2021.

[173] Karolina Kwapien, Eva Nittinger, Jiazhen He, Christian Margreitter, Alexey Voronov, and Christian Tyrchan. Implications of additivity and nonadditivity for machine learning and deep learning models in drug design. *ACS Omega*, 7 (30):26573–26581, 2022.

[174] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? *Advances in Neural Information Processing Systems*, 34, 2021.