

Few-Shot Learning for Low Data Drug Discovery

Daniel Vella

Supervised by Dr Jean-Paul Ebejer

Department of Artificial Intelligence
Faculty of ICT
University of Malta

March, 2022

A dissertation submitted in partial fulfilment of the requirements for the degree of M.Sc. in Artificial Intelligence.



L-Università
ta' Malta

University of Malta Library – Electronic Thesis & Dissertations (ETD) Repository

The copyright of this thesis/dissertation belongs to the author. The author's rights in respect of this work are as defined by the Copyright Act (Chapter 415) of the Laws of Malta or as modified by any successive legislation.

Users may access this full-text thesis/dissertation and can make use of the information contained in accordance with the Copyright Act provided that the author must be properly acknowledged. Further distribution or reproduction in any format is prohibited without the prior permission of the copyright holder.



**L-Università
ta' Malta**

Copyright ©2022 University of Malta

WWW.UM.EDU.MT

First edition, Friday 25th March, 2022

To my father

For showing me perseverance

Acknowledgements

I would firstly like to sincerely thank my supervisor, Dr Jean-Paul Ebejer, to whom this acknowledgement will not do any justice. I first had the pleasure of being taught by him in my first postgraduate year, where he captured my distracted attention with his passion and unique way of teaching. Apart from the subject matter, the passion for what he does was one of the prime reasons I sought him out to be my tutor. His patience, dedication and guidance were invaluable, in addition to his empathy and motivation to push through difficult times. Having given me his number to contact him conveniently about any dissertation related queries, I cannot help but include his WhatsApp status, as this contemporarily consolidates what I am trying to say about him perfectly: "Minghajr il-passjoni insiru robots" (*Without passion, we become robots*).

My gratitude also goes to my family, who have supported me with kindness and words of encouragement throughout my years of studies. This gratitude also extends to my friends, who have heard the phrase "*I need to work on my dissertation*" so much, that I'm afraid of losing a part of my personality upon its completion. To whoever heard this phrase throughout the past months, I sincerely appreciate you. I consider myself lucky to be surrounded by such people, and I hope I can reciprocate the support they showed me whenever they need it. Special thanks goes to Becky Micallef for proofreading this dissertation.

Abstract

Humans exhibit a remarkable ability to learn quickly from just few examples. A child seeing a cat for the first time can effectively identify the animal as a cat upon future encounters. This learning ability is in stark contrast with conventional machine learning (ML) techniques which are data hungry. This data requirement poses a challenge for the application of ML for virtual screening (VS) in drug discovery. The main goal in ligand-based VS (LBVS) is to identify active molecules that exhibit desired therapeutic activity against a biological target, based on information on known ligands against these targets. Data acquisition on a compound's activity against a biological target is resource-intensive and difficult to obtain. Hence, the aim of this study is to quantify whether few-shot ML can be effectively used for low-data drug discovery.

Meta-learning techniques aim to achieve the *learning how to learn* ability observed in humans. Therefore, we explore few-shot ML for this problem domain using the Tox21, MUV and the GPCR subset of the DUD-E datasets. In few-shot ML, we train a model using data from a number of experimental assays, and then use this model to generalise for new experimental assays using only a small (1-10 per class) support set. We build on the state of the art work, which is based on Matching Networks, by Altae-Tran et al. (2017), and use their work as a foundation to introduce two new architectures, the Prototypical Networks and Relation Networks, to this domain. Additionally, we also evaluate results using PR-AUC, in addition to ROC-AUC as in the original work, providing better interpretability of the performance of the proposed models on highly imbalanced data.

Our results are consistent with those of the state of the art on Tox21 and MUV datasets. To the best of our knowledge, the DUD-E dataset has not been previously explored for the few-shot learning domain. Our application of the Prototypical Networks, improved with the iterative-refinement LSTM, achieves overall better performance than the state of the art on Tox21 data. On MUV data, the baseline models outperform the few-shot learning models. On the GPCR subset of DUD-E, our results are not conclusive as on one target the models obtained outstanding performance and inferior performance on the other. We also experiment with different embeddings on the Tox21 data and find that learned graph embeddings consistently perform better than extended-connectivity fingerprints, a popular LBVS approach.

Based on our findings, we can conclude that the effectiveness of few-shot learning is highly dependent on the nature of the data available. The few-shot learning models struggle to perform consistently on MUV and DUD-E data, in which the active compounds are structurally distinct. However, on Tox21 data, which is typically used for lead optimisation, the few-shot ML models perform well and our contribution of the Prototypical Networks even outperforms the state of the art. Additionally, training these networks is much faster (up to 190% faster) and for comparable, or better results, take a fraction of the time to train.

Contents

1	Introduction	1
1.1	Motivation	5
1.2	Aims and Objectives	6
1.3	Approach	7
1.4	Document Structure	9
2	Background & Literature Overview	11
2.1	Drug Discovery Process	11
2.2	Virtual Screening	13
2.2.1	Structure-based Virtual Screening (SBVS)	14
2.2.2	Ligand-based Virtual Screening (LBVS)	14
2.3	Small-Molecule Databases	16
2.3.1	Toxicology in the 21st Century (Tox21)	17
2.3.2	Maximum Unbiased Validation (MUV)	18
2.3.3	Directory of Useful Decoys - Enhanced (DUD-E)	19
2.4	Machine Learning	19
2.4.1	Feed-forward Neural Networks	21
2.4.2	Convolutional Neural Networks	23
2.4.3	Recurrent Neural Networks	25
2.4.4	Graph Neural Networks (GNNs)	26
2.4.5	Evaluation Metrics	34
2.5	Learning with Low Data	36
2.5.1	Problem Definition	37
2.5.2	Siamese Networks	38
2.5.3	Matching Networks	40

2.5.4	Prototypical Networks	41
2.5.5	Relation Network	42
2.6	Molecular Machine Learning	43
2.6.1	Molecular Representation	44
2.6.2	Open-source Libraries	50
2.7	Related Work	52
2.8	Summary	56
3	Methodology	57
3.1	Overview	57
3.2	Data Acquisition	60
3.3	Generating the Molecular Representation	61
3.3.1	Standardise SMILES Molecules	63
3.3.2	Generate Molecular Features	64
3.3.3	Molecular Graph Generation	65
3.4	Few-Shot Machine Learning	66
3.4.1	Episodic Learning	67
3.4.2	Learning a Molecular Embedding	69
3.4.3	Training a Few-shot Machine Learning Model	73
3.4.4	ECFPs vs GCNs Learned Embeddings Experiments	76
3.5	Training Process and Hyper-parameters	77
3.5.1	Performance Monitoring	77
3.6	Testing	78
3.7	Evaluation	79
3.8	System and Software Specifications	80
3.9	Summary	81
4	Results & Evaluation	83
4.1	Revisiting aims and objectives	84
4.2	Benchmark Machine Learning Models	85
4.3	Few-shot Machine Learning Results	86
4.3.1	Evaluation Overview	87
4.3.2	ROC-AUC and PR-AUC Scores	89
4.3.3	ECFP vs GCN Learned Embeddings on Tox21	103
4.4	Machine Learning Models Training Run Times	104
4.5	Discussion	104
4.6	Summary	112

5	Conclusions	115	
5.1	Revisiting this Study's Aims and Objectives	117	
5.2	Critique and Limitations	119	
5.3	Future Work	120	
5.4	Final Remarks	122	
	Appendix A	Running Jupyter Notebooks	125
	References		127

List of Figures

1.1	Overview of the Drug Discovery Process	2
1.2	Few-shot Classification Example on Tox-21	4
2.1	Drug Discovery Process	12
2.2	Ligand and target binding	12
2.3	Screening methods	14
2.4	SBVS and LBVS Methods Criteria	15
2.5	Structural similarity spectrum.	16
2.6	MUV Data Selection Process	18
2.7	Model capacity	21
2.8	Simple Neural Network	22
2.9	Activation functions	23
2.10	2D CNN	24
2.11	Recurrent Neural Network	25
2.12	LSTM	26
2.13	Graph Deep Learning Operations	28
2.14	2D vs Graph Convolutions	29
2.15	Graph Convolution Function	30
2.16	Graph Convolutions	32
2.17	Convolutional GNN Architecture	34
2.18	Confusion Matrix	35
2.19	Confusion Matrix Visualisation	35
2.20	Meta-training and meta-testing	38
2.21	High level schematic of Siamese network	39
2.22	Matching Networks Architecture	41

2.23	Prototypical Networks	42
2.24	Relation Networks	43
2.25	SMILES generation in canonical order	45
2.26	ECFP Iterative stage.	47
2.27	Graph representation of acetic acid	48
2.28	Learned Embedding through a GCN	50
2.29	Molecule Net Datasets	51
2.30	Embedding functions for molecules	53
2.31	Iterative Refinement of Embeddings	54
2.32	Schematic of one-shot learning in drug discovery	55
3.1	Schematic of the major parts in our architecture	59
3.2	Molecular Representation Schematic	62
3.3	SMILES to RDKit Molecule	63
3.4	DGL Graph Visualisation	66
3.5	Episodic Learning Schematic	67
3.6	Graph Neural Network Schematic	69
3.7	Batching of Graphs	70
3.8	Learning an embedding through a GCN.	71
3.9	Layers for graph processing in our GCN.	72
4.1	Tox21 Test Targets ROC Box Plots for 10+/10- Support Set	90
4.2	Tox21 Test Targets PRC Box Plots for 10+/10- Support Set	90
4.3	ROC Scores for Matching Networks for target MUV-858	101
4.4	Visualising Training Times for Machine Learning Models	106
4.5	PRC plot for Tox21 SR-HSE - 10+/10- support set.	108
4.6	PRC plot for Tox21 SR-HSE - 1+/1- support set.	108
4.7	PRC plot for Tox21 SR-MMP - 10+/10- support set.	108
4.8	PRC plot for Tox21 SR-MMP - 1+/1- support set.	108
4.9	PRC plot for Tox21 SR-p53 - 10+/10- support set.	108
4.10	PRC plot for Tox21 SR-p53 - 1+/1- support set.	108
4.11	Confusion Matrix for Tox21 SR-HSE - 10+/10- support set.	109
4.12	Confusion Matrix for Tox21 SR-HSE - 1+/1- support set.	109
4.13	Confusion Matrix for Tox21 SR-MMP - 10+/10- support set.	109
4.14	Confusion Matrix for Tox21 SR-MMP - 1+/1- support set.	109
4.15	Confusion Matrix for Tox21 SR-p53 - 10+/10- support set.	109
4.16	Confusion Matrix for Tox21 SR-p53 - 1+/1- support set.	109

4.17	ROC plot for MUV-832 with a 1+/10- support set.	111
4.18	Confusion Matrix for MUV-832 with a 1+/10- support set.	111
4.19	ROC-AUC scores between embeddings on Tox21 PN 10-shot support sets. . .	113
4.20	PR-AUC scores between embeddings on Tox21 PN 10-shot support sets. . . .	113

List of Tables

2.1	Examples of SMILES representations.	46
2.2	GCN Architecture used in the state of the art work.	54
3.1	Tox21 Dataset Composition	61
3.2	MUV Dataset Composition	62
3.3	Composition of the GPCR subset from DUD-E	62
3.4	Feature vector size for molecular graphs	64
3.5	Support set composition	68
3.6	Graph Convolution Network Architecture	72
3.7	Benchmark Neural Network for Few-Shot Learning	74
3.8	The architecture for generating the relation score using function g_{θ}	76
3.9	Neural Network Architecture for ECFPs.	77
3.10	Hyperparameters and Optimisation	78
3.11	Python libraries utilised for this project.	80
3.12	Hardware provisioned in Google Colab.	81
4.1	Dataset actives and inactive/decoys proportions.	85
4.2	ROC-AUC and PR-AUC Scores for TOX21 SR-HSE Target.	92
4.3	ROC-AUC and PR-AUC Scores for TOX21 SR-MMP Target.	93
4.4	ROC-AUC and PR-AUC Scores for TOX21 SR-p53 Target.	94
4.5	ROC-AUC and PR-AUC Scores for MUV MUV-832 Target.	96
4.6	ROC-AUC and PR-AUC Scores for MUV MUV-846 Target.	98
4.7	ROC-AUC and PR-AUC Scores for MUV MUV-852 Target.	99
4.8	ROC-AUC and PR-AUC Scores for MUV MUV-858 Target.	100
4.9	ROC-AUC and PR-AUC Scores for MUV MUV-859 Target.	101
4.10	ROC-AUC and PR-AUC Scores for DUD-E GPCR adrb2 Target.	102

4.11 ROC-AUC and PR-AUC Scores for DUD-E GPCR cxcr4 Target.	103
4.12 ROC-AUC and PR-AUC Scores for Prototypical Networks on Tox21 with ECFP or GCN embeddings	105
4.13 Comparing our best ROC-AUC scores with the SOTA results on Tox21.	110

List of Abbreviations

ADMET Absorption, Distribution, Metabolism, Excretion, and Toxicity

ADR Adverse drug reactions

ANN Artificial Neural Network

AUC Area under the Curve

CNN Convolutional Neural Network

DGL Deep Graph Library

DL Deep Learning

DNN Deep Neural Network

DUD Directory of Useful Decoys

DUD-E Database of Useful (Docking) Decoys - Enhanced

ECFP Extended Connectivity Fingerprints

GAT Graph Attention Network

GCN Graph Convolutional Network

GNN Graph Neural Network

HTS High-Throughput Screening

LBVS Ligand-based virtual screening

LSTM Long-Short Term Memory

- MN** Matching Network
- ML** Machine Learning
- MLP** Multilayer perceptron
- MUV** Maximum Unbiased Validation
- RecGNN** Recurrent Graph Neural Network
- ReLU** Rectified Linear Unit
- RF** Random Forest
- RNN** Recurrent Neural Network
- ROC** Receiver Operator Characteristic
- SAR** structure-activity relationships
- SBVS** Structure-based virtual screening
- SIDER** Side-Effect Resource
- SMILES** Simplified Molecular Input Line Entry System
- SPP** Similar Property Principle
- TanH** Hyperbolic Tangent
- Tox21** Toxicology in the 21st Century Program
- PN** Prototypical Network
- PRC** Precision-Recall Curve
- RN** Relation Network
- VS** Virtual Screening

Introduction

We humans exhibit a remarkable ability to learn new concepts fast and efficiently. A child seeing a cat for the first time is able to discriminate future encounters with cats from other animals. This ability is in stark contrast with conventional supervised end-to-end machine learning, which is data hungry and requires a plethora of data points to develop an effective model. Meta-learning reframes the traditional machine learning problem, allowing machine learning models to learn using only a few examples. Humans have an innate capability to *learn how to learn*, and bridging this gap between human and machine learning is beneficial, particularly in domains where data availability or acquisition is difficult, such as the drug-discovery domain.

The drug-discovery process is known for its exorbitant costs and resource expenditure. Hughes et al. (2011) report that from the start of the process to the launch of the medicinal product, costs can exceed one billion dollars and the process can take up to 15 years. Every stage visualised in Figure 1.1 can take years to complete. Moreover, data acquisition for lead identification and optimisation is also expensive and difficult to acquire, as this requires testing of a large number of compounds *in-vitro* and *in-vivo*. In lead identification, molecules which exhibit biological activity against a target are identified. Following identification, molecules need to pass the lead optimisation step, where characteristics such as absorption or toxicity in the body are determined.

The main goal in the drug discovery process is the development of active compounds that exhibit therapeutic effects against targets. These compounds, known as ligands, etymologically meaning *to bind*, are molecules that upon binding to a target trigger some biological activity. A target is a broad term, encompassing genes, proteins, and molecular functions or pathways associated with a disease. When ligands exhibit the desired biological activity, they are referred to as leads (Arya and Coumar, 2021).

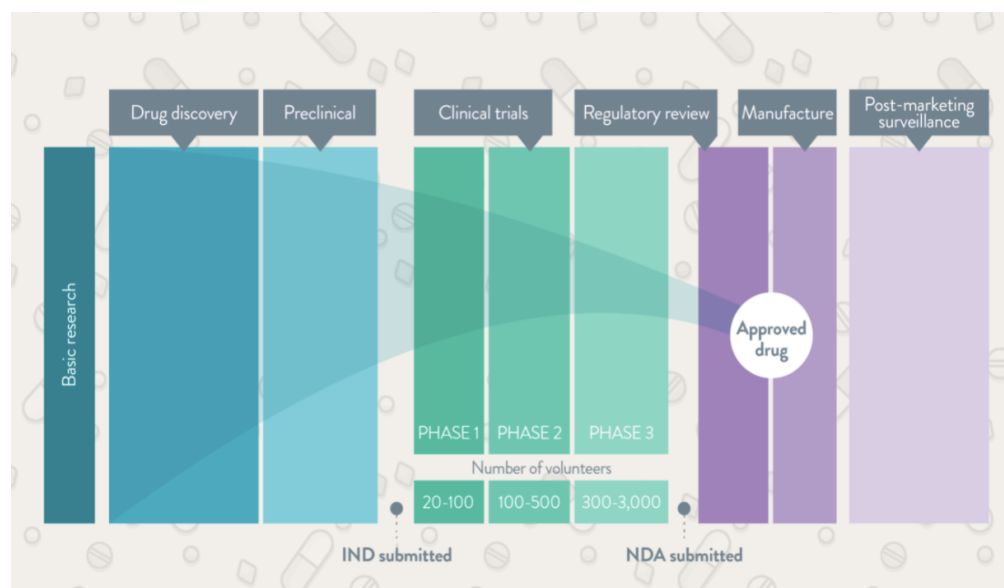


Figure 1.1: Overview of the drug discovery, development and approval process. Thousands of compounds are filtered down to one approved drug-like compound. Reproduced Figure.¹

The dominant approach for the identification of new leads is High-Throughput Screening (HTS), a physical *in-vitro* screening process where large numbers of compounds are tested against a biological target. HTS makes use of robotics to screen a large number of compounds efficiently, but this process is costly in time and resources. Computational advancements have enabled the use of *in-silico* methods to speed up the identification and optimisation of leads. Virtual screening is an *in-silico* compound database search with the goal of finding compounds which have a high probability of exhibiting activity against a specific biological target. One of the benefits of this approach is to narrow down the search of thousands of compounds that make their way to physical laboratories, effectively reducing resource expenditure. Virtual screening can be sub-divided into two key categories, namely *structure-based* (SBVS) and *ligand-based* virtual screening (LBVS). The former uses prior knowledge on the biological target, while the latter, around which this study is focused, makes use of knowledge of known ligands. This method of screening is a neighbourhood search, utilising information on known active molecules to identify new ligands. LBVS capitalises on structure-activity relationships (SARs), whereby ligands are identified through some similarity function against known active ligands (Hamza et al., 2012). This notion is based on the similar property principle (SPP), which states that molecules showing similar structural conformations often

¹ Accessed From: <https://bit.ly/3mdKTV8> - Last Accessed: 26 October 2021.

exhibit similar biological activity (Johnson and Maggiora, 1990), allowing known active ligands to be used as *templates* for new leads.

The datasets used for ligand-based virtual screening consist of a set of active ligands (the positive class) along with inactives or decoys for a particular assay (the negative class), usually for multiple targets. One common representation for molecules in these datasets are Simplified Molecular Input Line Entry System (SMILES) strings (Weininger, 1988). The datasets used for this study contain a set of experimental assays with binary labels showing whether a molecule is active or not for that specific experimental assay. One challenge for machine learning techniques on LBVS is that these datasets are highly imbalanced, containing a much higher proportion of inactives or decoys than actives. Data on active compounds is both resource-intensive and difficult to obtain. Hence, this imbalance should be taken into consideration when evaluating the performance of machine learning models.

Molecules are complex structures, consisting of atoms and bonds, which make up different conformations of the molecule. One challenge in the field of cheminformatics is the representation of molecules in the computational space. The classical notation of compounds is the empirical formula such as $C_3H_7NO_2$, however, this holds no information on how the atoms are linked together. In fact, this particular formula can refer to alanine, sarcosine, and lactamide. Molecular representations such as Extended Connectivity Fingerprints (ECFP) (Rogers and Hahn, 2010) and graph convolution learned embeddings (Duvenaud et al., 2015) contain more information on the properties of the molecule, and can be used as inputs to machine learning networks. A graph is formally defined as a set of nodes and a set of edges, where each edge connects a pair of nodes. This notion intuitively translates to molecular representations where atoms are the set of nodes, and the bonds are the set of edges. Graphs are 2D objects, so spatial properties of a molecule such as bond angles and chirality are not inherent to the data object, but are instead encoded as node or edge attributes (David et al., 2020). Using graph convolutional neural networks, embeddings of molecular graphs, augmented with atom feature information can be learned, which could be of benefit over topological molecular representations such as ECFP (Wu et al., 2018).

Molecular embeddings in computational space can be subsequently used as inputs to train machine learning models. The cost and difficulty of data acquisition in the drug discovery domain has been established from the start. Building on this notion, in this study, we aim to explore few-shot learning for virtual screening to address the low-data problem in this problem domain. The ability for a machine learning model to learn new concepts fast with just a few training examples would be invaluable for virtual screening. Meta-learning aims to achieve this generalising capability for new environments

that have not been encountered during training time. Meta-learning models are trained using a variety of training tasks and optimised for performance over a distribution of tasks, including unseen ones. Few-shot classification is a type of meta-supervised learning. Learning consists of a series of episodes, each consisting of a N -way K -shot classification task, effectively simulating the conditions at testing time. The *way* refers to the number of classes we have per task and the number of samples we have is the *shot* component. These samples make up the support set (Snell et al., 2017). During test time, a small support set is sampled from new, previously unseen targets, and these few data points are used by the model to generalise for the activity of query molecules against this new target (Vinyals et al., 2016).

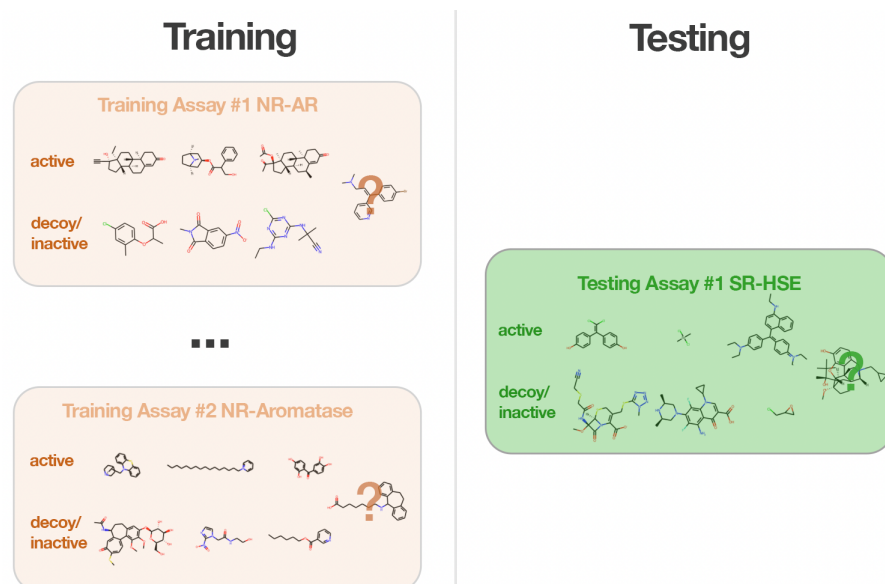


Figure 1.2: 2-way 3-shot few-shot classification. Training a meta-learner on a set of experimental assays, and generalising for an unseen assay in the Tox-21 dataset.

We highlight that few-shot learning in the domain under study is in contrast to other domains such as computer vision, where the model is trained to recognise new classes. For example, given a few images of a lion as the support set, the model must generalise for new unseen images of a lion. In this domain, the challenge is to generalise for the behaviour of molecules in experimental assays which are related but not identical to the assays in the training collection. Given a few molecules from new experimental assays, can the model predict the activity of other molecules in this new assay? Thus, the molecules might have already been seen during training, but the corresponding

label would be for another distinct experimental assay. Figure 1.2 illustrates this method of learning using only 2 assays as training data for a 2-way 3-shot few-shot classifier.

In this study, we explore the application of a number of few-shot learning architectures including, in chronological order, siamese networks (Koch et al., 2015), Matching Networks (Vinyals et al., 2016), Prototypical Networks (Snell et al., 2017) and Relation Networks (Sung et al., 2018). This group of architectures fall under the umbrella of metric-based meta-learning. In our study, we embed molecule representations using graph convolution networks, and then use or learn a distance function over these embeddings. Effectively, metric-based learners seek to learn a relationship between the input embeddings in the task space. For the purposes of this study, few-shot learning refers to training with as little as one example per class, to a maximum of 10 examples per class. Training with only one example per class is referred to as one-shot learning (Koch et al., 2015; Vinyals et al., 2016).

1.1 | Motivation

The discovery and development of drugs is an expensive and resource-intensive process. Moreover, the probability of success for drug development programs is low (Hay et al., 2014; Wong et al., 2019). Even upon identification of a small-molecule compound that achieves a desired therapeutic activity, attrition rates are high as the compound usually fails for other reasons such as poor absorption, distribution, metabolism and excretion (ADME) or toxicology characteristics (Waring et al., 2015). It is difficult to predict such characteristics about the candidate molecule when only a small amount of related biological data is available. Therefore, the lead identification and optimisation step in drug discovery is essentially a low-data problem (Altae-Tran et al., 2017). The applicability of machine learning to the lead identification and optimisation step of the drug discovery process is hindered by the requirements for the plethora of data required to train a model. In the past years, machine learning for the low-data domain has been applied with success for the computer vision domain (Koch et al., 2015; Snell et al., 2017; Vinyals et al., 2016). Studies have shown that it is possible to train a model using only a few data points. Few-shot learning relieves the burden of collecting large-scale labelled data and makes the learning of rare cases possible (Wang et al., 2020). However, similar research ventures in the drug discovery are limited. Thus, in this work, we explore the generalising capabilities of machine learning models using only a few training examples. The end goal is to successfully classify the predicted activity of a molecule in an unseen experimental assay, using only a few known ligands from this assay. This falls

within the domain of ligand-based virtual screening as we make use of information on known ligands to identify new ones for a specific experimental assay.

Altae-Tran et al. (2017) introduce a deep-learning architecture for few-shot learning in drug discovery, building on past work in metric-based meta-learning. The authors propose the iterative refinement long short-term memory (LSTM) which builds further on the Matching Networks (Vinyals et al., 2016), originally proposed for the computer vision domain, by introducing iterative refinement of embeddings using long-short term memory (LSTM) networks. We will explore and quantify the effectiveness of such machine learning techniques in the drug discovery domain. Additionally, to our knowledge, no attempts have been made to explore subsequent developments in the few-shot learning domain following the Matching Networks application by Altae-Tran et al. (2017). Thus, in this study, we will draw inspiration from successful few-shot learning approaches in other domains, such as the Prototypical (Snell et al., 2017) and Relation (Sung et al., 2018) Networks, and apply the same concepts to this problem domain.

1.2 | Aims and Objectives

The aim of this study is to determine the efficacy of low-data machine learning for Ligand-based virtual screening (LBVS) for the lead identification and optimisation part of the drug discovery process. We explore whether a machine learning model can generalise well enough to classify the activity of molecules in a new experimental assay using only a few examples as the support set, after being trained for similar tasks on other related, but not identical experimental assays. In this study, the following objectives are achieved to satisfy the defined aim:

1. Build molecular representations for the computational space using ECFP and neural graphs.
2. Establish conventional supervised machine-learning models as a baseline benchmark for the few-shot learning approaches.
3. Investigate the performance of different few-shot learning architectures, taking into consideration the unbalanced nature of the datasets at hand.
4. Investigate the performance of low-data machine learning models on different datasets/tasks.

5. Determine whether one molecular representation is superior to the other for few-shot learning.

Our aim and objectives are essentially designed to answer two main research questions, which serve as the foundation for this research study.

Research Questions

1. Since we have limited active data, a model that can generalise using only a few examples is highly advantageous. Therefore, are few-shot machine learning techniques effective for low-data ligand-based virtual screening?
2. Do Prototypical (Snell et al., 2017) and Relation networks (Sung et al., 2018) offer better performance for ligand-based virtual screening than the Matching Network (Vinyals et al., 2016) component in the established state of the art (Altae-Tran et al., 2017)?

Additionally, the implementations of our molecular representation pipelines and few-shot learning models are open sourced (see Appendix A) to facilitate the reproduction of results and further research. The models are also in the process of being possibly integrated within the DeepChem project (Ramsundar et al., 2019).

1.3 | Approach

We start from a comprehensive review of the current state of the art approaches in the drug discovery domain for few-shot meta-learning. The state of the art, identified to be the work by Altae-Tran et al. (2017), set our initial trajectory. Following this, another review ensued in which we explored few-shot learning approaches in other domains, specifically the computer vision domain.

Following the literature review, we reproduce the work of Altae-Tran et al. (2017). While the authors open-sourced their primitives and machine-learning models, the code is highly dependent on the DeepChem library and at the time of writing, their implementation was not functional as it required outdated versions of both Tensorflow and DeepChem. Therefore, their work was reimplemented from scratch. We also attempt to draw inspiration from other advances in the few-shot meta-learning domain to further build on the work that currently exists in the drug discovery domain.

Our machine learning pipeline is composed of the following main components:

1. **Data Loading.** The datasets utilised for this study are the Toxicology in the 21st Century Program (Tox21) (NIH, 2014), Maximum Unbiased Validation (MUV) (Rohrer and Baumann, 2009), and the GPCR subset of Database of Useful (Docking) Decoys - Enhanced (DUD-E) (Mysinger et al., 2012). These are loaded as CSV files at the start of our machine learning pipeline.
2. **Conversion of SMILES to ECFP or Graphs.** We experiment with different molecular representations, namely ECFP and graph convolutions.
3. **Learning the molecular embedding.** A neural network learns the molecular embeddings to output an n -sized vector that is used as input further down the machine learning pipeline. In the case of graphs, we implement a graph convolutional network to learn the embedding.
4. **Training the few-shot learning model.** Each dataset is composed of a number of assays which are reserved for training. Training is conducted in episodes made up of 2-way K-shot classification tasks, where we experimented with varying amounts of samples for each class for the support set. The range of sampled values for each class making up the support set was from one example to 10 examples per class. The learned embeddings from the previous step are refined using LSTMs as proposed in the state of the art work (Altae-Tran et al., 2017). The implemented metric-based architectures are Siamese Networks, Matching Networks, Prototypical Networks and Relation Networks.
5. **Testing the model on new experimental assays.** A subset of assays in each dataset are reserved for testing, from which we randomly sample a support set, which the model uses to predict the activity of the remaining queries against the new, previously unseen target.

We finally compare and contrast the results obtained with the results reported in Altae-Tran et al. (2017). Additionally, we provide more reliable metrics for the problem domain to improve the interpretability of the machine learning models. We also conduct and present results on few-shot learning for the DUD-E dataset, for which to the best of our knowledge, no other similar attempt has been done prior to our study.

1.4 | Document Structure

The rest of this dissertation is composed of a total of five chapters. The four chapters to follow are outlined in the following list;

- In the **Background and Literature Overview** chapter, we provide a brief overview of the drug discovery process and dive deeper into virtual screening. Molecular datasets relevant to this study are presented. We present commonly used computational representations of molecules, such as fingerprints and graphs. These form the inputs of our machine learning networks, for which we lay the foundations through the introduction of feed-forward, convolutional, recurrent and graph neural networks. We finally delve into the crux of this study, few-shot learning. Existing literature for few-shot learning is presented, along with state of the art developments in the molecular domain.
- In the **Methodology** chapter, we present the approach taken to explore and build a relevant study on few-shot learning in the drug discovery domain. The chapter expands on the machine learning models and any resources used for the development of the presented architectures. Arguments supporting the choice of such architectures are also presented to add relevance to our study.
- In the **Results and Discussion** chapter, we present the conducted experiments and the respective results. We discuss results and compare and contrast our results with that of existing state of the art literature in the field. We also explain how our work builds on the state of the art literature by drawing inspiration from advances in few-shot learning for other domains, specifically the image domain.
- In the **Conclusion** chapter, we culminate the discussion by summarising the work carried out while highlighting our contributions to few-shot learning in the ligand-based virtual screening domain. We conclude by proposing relevant future work that can provide relevant contributions to the field by further building on this and other studies.

Background & Literature Overview

In this chapter, we expound on the theoretical aspects required for the understanding and appreciation of this study. This study revolves around cheminformatics, coined in 1998 (Brown), which is the application of computational techniques for chemistry. Cheminformatics involves the use of *in-silico* techniques to guide the drug discovery and development process through the utilisation of molecular databases (Prakash and Gareja, 2010). We focus mainly on virtual screening, which is used to computationally screen compound libraries to identify potential candidates likely to exhibit therapeutic activity against targets, and further optimisation on these leads. After introducing these concepts, the chapter introduces and expands on machine learning concepts required for this study, including few-shot learning techniques. Finally, we look at the state of the art work for few-shot learning for this problem domain.

2.1 | Drug Discovery Process

The drug discovery process is multifaceted, mainly involving the identification and development of a compound that can be used to treat and manage a disease condition. The process is time-consuming, resource intensive, and the probability of success for drug development is reported to be low (Hay et al., 2014; Wong et al., 2019). Moreover, attrition rates of compounds which make it to further stages of the drug discovery process is high, even in the final clinical phases. In fact, thousands of compounds are tested, and only one may eventually make it to market. From process inception to the launch of the medicinal product, the process can take up to 15 years and costs can exceed \$1 billion (Hughes et al., 2011).

Initial research generates data to formulate a hypothesis about the effect observed

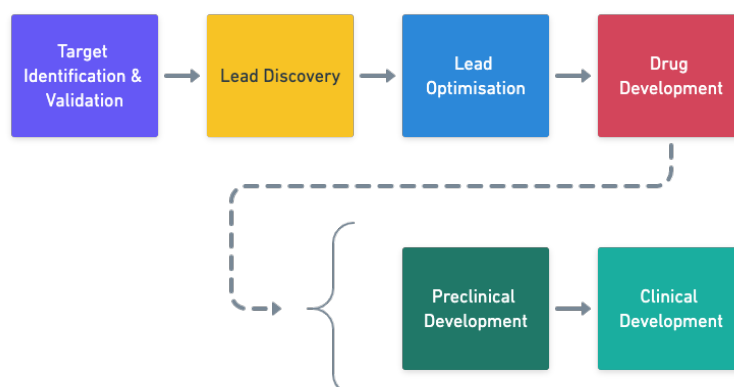


Figure 2.1: High-level schematic of the drug discovery process. Adapted from Hughes et al. (2011).

from activation or inhibition of a biological target. A target is a broad term that can encompass anything from molecular entities, such as genes, proteins or RNA, to biological functions such as molecular functions or pathways associated with a disease. The primary goal in the drug discovery process is to identify and develop ligands that exhibit the desired therapeutic effect against identified targets (see Figure 2.2).

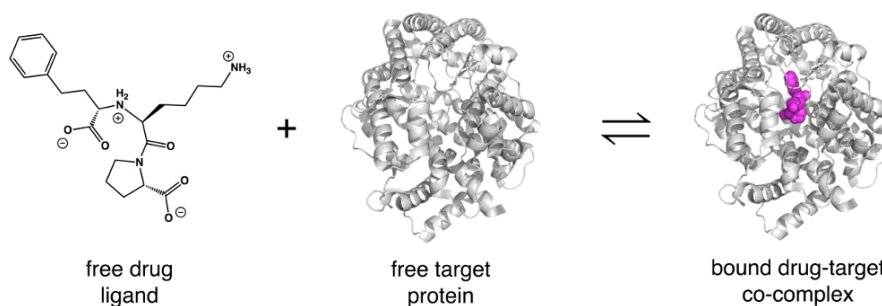


Figure 2.2: Ligand and target protein binding to create a complex. Reproduced Figure.¹

A thorough search for an active drug-like small molecule (i.e. the ligand), referred to as the development candidate, ensues. This is part of the lead discovery phase and involves both hit identification and lead optimisation. Large libraries of ligands that can interact with the target need to be screened to identify *hits*. A *hit* molecule refers to one that shows consistent activity in an assay. The search requires multiple *screening* approaches, involving the interplay between both experimental and computational approaches to effectively guide the search in a more productive way (Reddy et al., 2007).

¹Accessed from: <https://kambria.io/blog/modern-ai-drug-discovery/>. Last Accessed: 15 Nov 2021

High Throughput Screening (HTS) technology utilises automation and robotics to identify active molecules from thousands of chemicals against target assays. Assays used in HTS can be subdivided into two categories, namely *biochemical* assays and *cell-based* assays. Biochemical assays make use of homogeneous reactions to test for binding or affinity of compounds with the target of interest (Zang et al., 2012). However, such activity might not be representative of the activity in a cellular context, so *in vitro* cell-based assays, as the name implies, addresses this problem by making use of cultured cells for testing (An and Tolliday, 2010). A typical HTS process involves the testing to obtain early information on Absorption, Distribution, Metabolism, Excretion, and Toxicity (ADMET) of tested compounds. The results from such screening techniques can be used to build databases of molecular activity, however, this process is resource-intensive and expensive.

Computer-aided drug discovery and design approaches optimise the search, thus reducing both the exorbitant costs that come with the aforementioned experimental efforts and the time to market of developed drugs. Computational methods, such as Virtual Screening (VS), afford the prediction of certain properties in order to screen the most promising compounds (Van De Waterbeemd and Gifford, 2003), effectively filtering molecule databases for compounds that make it to later stage *in-vitro* processes.

Screening is used to identify hits and optimise leads, which progress further in the drug discovery pipeline to further optimisation, drug development and clinical phases (refer to Figure 2.1) (Hughes et al., 2011). This study is related to the lead discovery and lead optimisation part of the drug discovery process. Our study further focuses on ligand-based virtual screening (LBVS) which is explained in more detail in Section 2.2.2.

2.2 | Virtual Screening

VS is an *in-silico*, compound database searching process with the goal of finding compounds which have a high probability of exhibiting activity against a specific biological target. Virtual screening is subdivided into two main categories, namely structure-based (SBVS) and ligand-based (LBVS) (Lavecchia and Di Giovanni, 2013). The starting point for the former is the target protein itself, while for the latter category, it is a known ligand. This study focuses on LBVS predictive models, however, it is worth noting that the two categories may be combined, and most current virtual screening studies do (Ferreira et al., 2015). Both screening techniques can be used in conjunction if both the 3D structure of the target, affording the use of SBVS techniques, and a number of active compounds, affording the use of LBVS techniques, are known (refer to Figure 2.4).

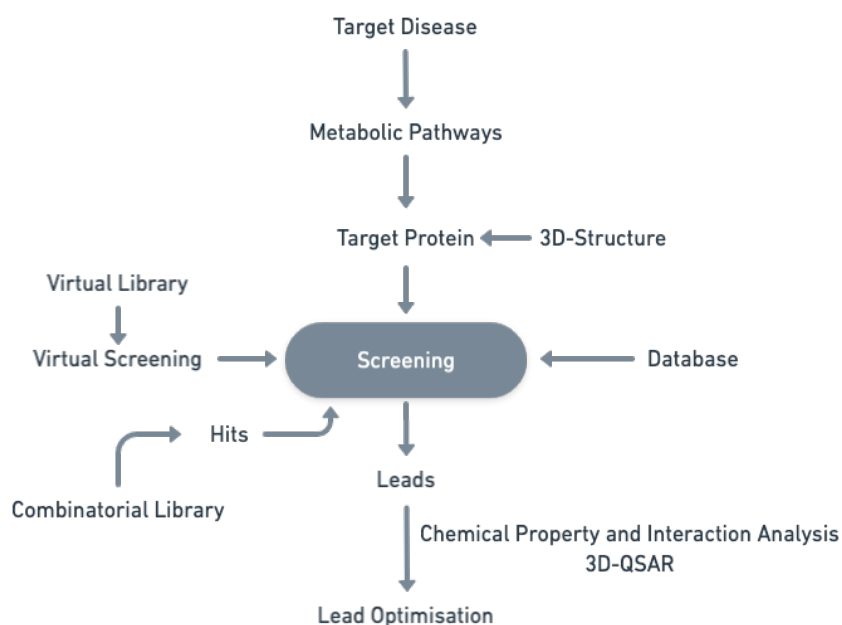


Figure 2.3: Screening methods involve the interplay of experimental and computational screening approaches to filter through databases in a rational way. Adapted from Reddy et al. (2007).

2.2.1 | Structure-based Virtual Screening (SBVS)

SBVS techniques require knowledge about the 3-dimensional (3D) structure of a biological target. Libraries of small molecules are screened against this target by docking into the active site of the target using computation techniques. The binding affinity between the molecule and the target is evaluated and ranked using a scoring function (Lavecchia and Di Giovanni, 2013). Molecules can take on multiple spatial arrangements, better referred to as *conformations*, due to the rotation around single bonds. As a result, scoring is not a trivial computational task, as molecular docking involves (i) the exploration of a large conformational space and (ii) the calculation of the energy associated with each conformation (Ferreira et al., 2015) to compute protein-ligand interactions.

2.2.2 | Ligand-based Virtual Screening (LBVS)

A ligand, etymologically meaning "to bind", is a molecule that can bind to a protein molecule, resulting in some biological activity. LBVS, also referred to as neighbourhood behaviour search, makes use of the information available on known active molecules to identify new ligands. One method is based on molecular descriptor exploration from

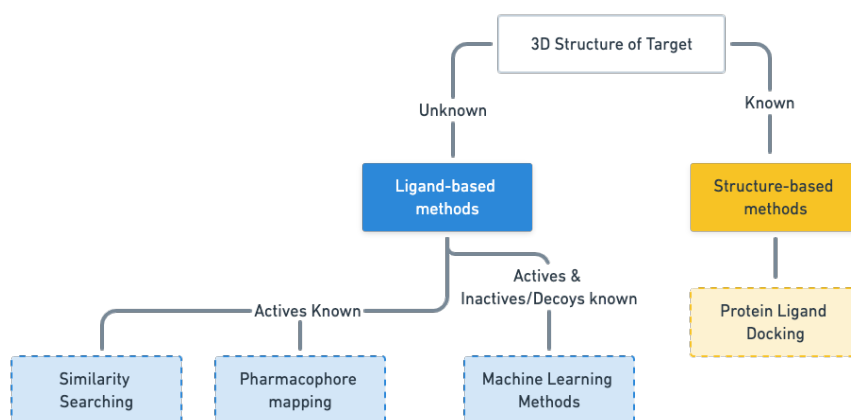


Figure 2.4: SBVS and LBVS Methods Criteria. Adapted from Pal (2016).

known active compounds, in which a set of mutual descriptors from known actives are used to filter out molecules from a database (Geppert et al., 2010). Another technique in LBVS is the use of structural features from known ligands to generate pharmacophore models (Spitzer et al., 2010). A pharmacophore is a substructure of a molecule that is responsible for specific interactions. Thus, these techniques are used to reduce the chemical space to be searched further down the drug-discovery pipeline. Databases of molecules are searched to identify either structurally similar molecules, or ones that possess a common pharmacophore with the known active (pharmacophore substructure search) (Reddy et al., 2007). The notion of LBVS techniques is in contrast with that of SBVS. The latter uses prior knowledge about the properties of the biological target, while LBVS uses prior knowledge on known ligands.

LBVS capitalises on structure-activity relationships (SAR)s, whereby new ligands are identified by evaluating the similarity between candidate ligands and known active ligands against a particular target (Hamza et al., 2012). One of the theories underpinning similarity searching techniques is the *Similar Property Principle (SPP)*, stating that molecules with close structural similarity often exhibit similar biological activity (Johnson and Maggiora, 1990). Figure 2.5 visualises the relationship between identified hits and the structural similarity of the molecule. Similarity-based methods are cornerstones of chemoinformatics. Based on the SPP, known active ligands can be used as *templates* for finding other molecules with a high probability of exhibiting affinity to the target.

However, Maggiora (2006) postulates that SARs are not always consistent with the SPP as in some cases, similar molecules may exhibit drastic changes in activity. This is referred to as the *activity cliff*, signifying the ratio between the similarity of two molecules and the difference in their activity. Such inconsistencies are a limitation of SARs and re-

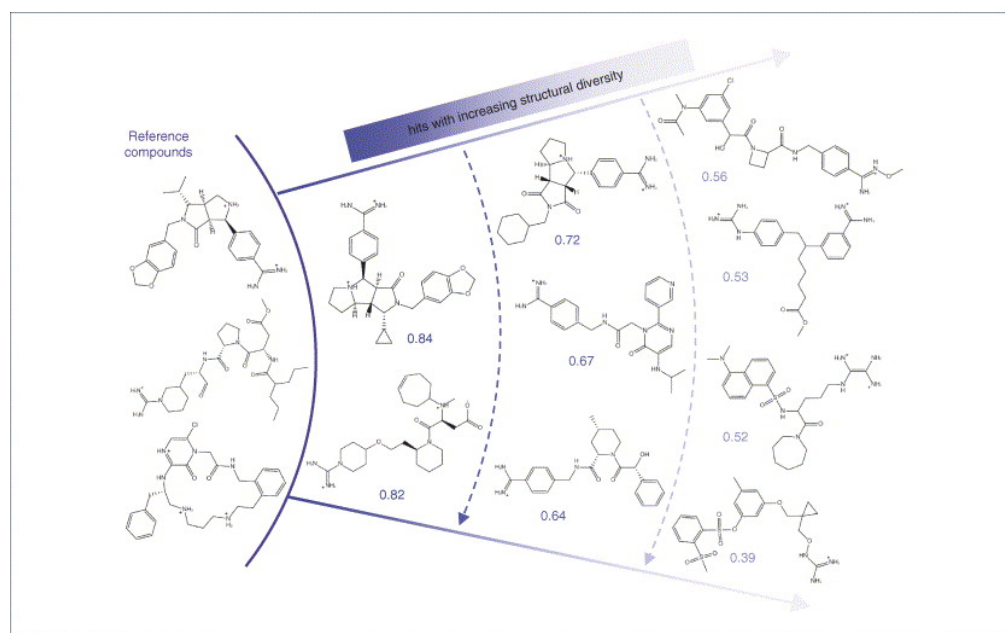


Figure 2.5: Structural spectrum of thrombin inhibitors, visualising hits arranged within layers of increasing structural diversity from top to bottom and left to right. Structural similarity is compared using the Tanimoto coefficient on molecular fingerprints based on 166 MACCS structural keys. Reproduced from Eckert and Bajorath (2007).

sult in outliers in the data. Maggiora (2006) concludes that advancements in computational techniques such as machine learning can ameliorate SAR models in such scenarios, however, the quality of data and the molecule representation defining the chemical space of the molecules is a limiting factor.

2.3 | Small-Molecule Databases

Small-molecule databases contain records consisting of a SMILES representation (see Section 2.6.1) and a Boolean value (0/1) for classification tasks indicating the molecule's activity in a specific assay. Regression tasks are also available for certain molecule databases, such as QM7 and Free-Solv (Wu et al., 2018). However, they are beyond the scope of this study. The datasets relevant for this study are made up of active and inactive or decoy compounds against a target in a specific assay. Typically, each dataset contains multiple assays for different pathways or target proteins. The data for active molecules is empirically obtained through *in-vitro* or *in-vivo* experimentation (Réau et al., 2018). On the other hand, the data for decoys is not always empirical. It

is also worth mentioning that such datasets are highly imbalanced, with the proportion of the number of inactive molecules being significantly greater than that of active molecules. For example, in clinical cases there is unavoidable higher data for the control groups than the treatment groups (Malin et al., 2013).

- **Actives.** Molecules which are empirically known to bind to a target, successfully active or inhibit a biological or chemical function. Molecules with a high confidence of bio-activity through virtual screening can also be regarded as *actives*.
- **Inactive.** True inactive molecules are ones that are confirmed to be inactive through experimental assays. Inactivity implies that when in contact with a target, no desired activation or inhibition is observed (Réau et al., 2018). Putative inactive molecules (also called *decoys*) are also used in datasets.
- **Decoy.** Given that data on inactive compounds is scarce in literature, putative inactive compounds are used instead. Decoy compounds are assumed to be inactive and not experimentally confirmed to be truly inactive. Therefore, false positives may arise. Structure-based virtual screening (SBVS) techniques such as protein-ligand docking and scoring are used to discriminate between active and inactive molecules, however several biases can influence this outcome (Verdonk et al., 2004). One example of this bias includes the difference in chemical space between active and decoys, leading to artificial overestimation of the molecule's activity (Bissantz et al., 2000). Databases such as the MUV (see Section 2.3.2) and DUD-E (see Section 2.3.3) were designed to minimise such biases.

For the purposes of this study, we explore three main datasets, namely the Toxicology in the 21st Century (Tox21) (Huang et al., 2016), the Maximum Unbiased Validation (MUV) (Rohrer and Baumann, 2009) and the Directory of Useful Decoys - Enhanced (DUD-E) (Mysinger et al., 2012). We expand on these datasets in the sections to follow.

2.3.1 | Toxicology in the 21st Century (Tox21)

The Tox21 library contains several thousand compounds, which have been tested through quantitative HTS against a number of nuclear receptors (NR) and stress response (SR) pathway assays. The National Center for Advancing Translational Sciences (NCATS) launched the Tox21 Data Challenge in 2014 to crowdsource data analysis on a subset of the Tox21 library, containing 10,000 compounds tested against a total of 12 NR and SR pathway assays (Huang et al., 2016). Tox21 is a toxicology dataset and is therefore used primarily in lead optimisation, to filter out leads which exhibit toxicity against a

number of biological targets. Toxicity is one component of the ADMET, which plays an important role in drug discovery.

The NR assays provide information for the activity against a number of targets, namely the Androgen Receptor (AR), Androgen Receptor Ligand Binding Domain (AR-LBD), Aryl Hydrocarbon Receptor (AhR), Estrogen Receptor (ER), Estrogen Receptor Alpha Ligand Binding Domain (ER-LBD) aromatase inhibitors, and the Peroxisome Proliferator Activated Receptor gamma (PPAR-gamma). The SR assays provide information about the activity against antioxidant response element (ARE), ATPase Family AAA Domain Containing 5 (ATAD5), heat shock response (HSE), Mitochondrial Membrane Potential (MMP) and Cellular tumor antigen (p53) assays (NIH, 2014).

2.3.2 | Maximum Unbiased Validation (MUV)

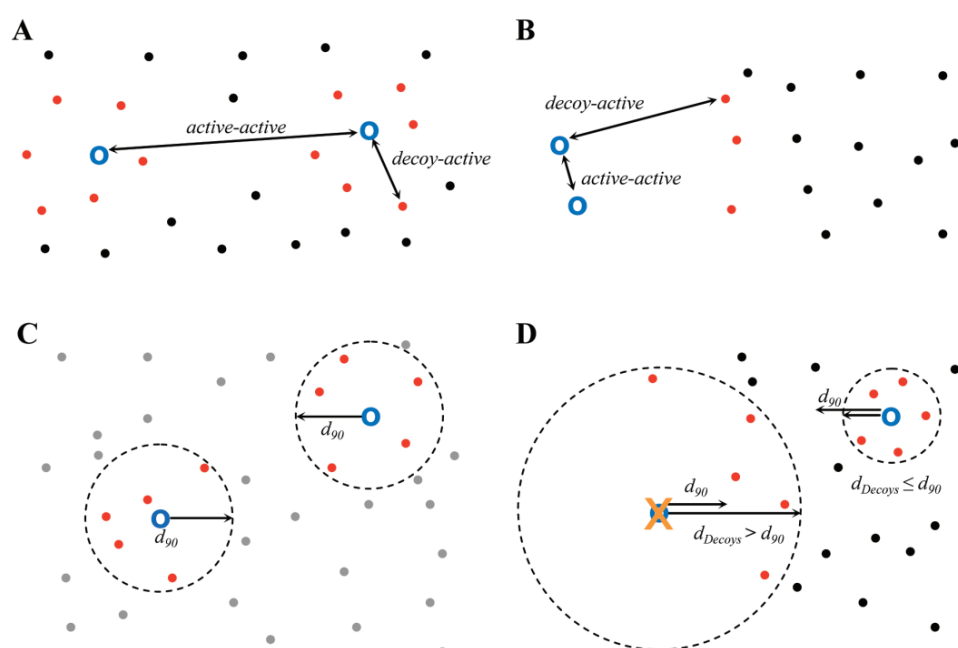


Figure 2.6: A) Artificial enrichment is avoided by having larger distances between actives when compared to distances between decoys and actives. B) If the opposite happens, artificial enrichment is the result. C) The 500th nearest neighbour distance is computed for every active ($k=5$ visualised for the sake of clarity). D) If an active is in a region that is significantly lacking in decoys, which is determined by the radius from the active being larger than the distance to the 100th nearest neighbour (with a 90% confidence boundary), the active is rejected. Reproduced from Rohrer and Baumann (2009).

The MUV dataset was designed to minimise biases in screening databases and is typically used to validate VS techniques. Refined nearest neighbour analysis is used to benchmark PubChem (Kim et al., 2021) BioAssay² data on bioactivity. Initially, compounds which exhibit potential for unspecific bio-activity are eliminated through an assay artifact filter. Active molecules which are not found within the same chemical space as decoys are subsequently eliminated to reduce an inherent chemical space bias through the chemical space embedding filter. The spread of actives within the chemical space is adjusted to a common level, eliminating actives outside this threshold. The distance between actives and decoys is also normalised to a common level to enforce spatial randomness. Figure 2.6 shows the process of active and decoy selection for the MUV dataset, which is composed of 500 decoys for each active. MUV provides a dataset of potential actives (PA) and potential decoys (PD) reducing inherent chemical space bias and artificial enrichment (Rohrer and Baumann, 2009).

2.3.3 | Directory of Useful Decoys - Enhanced (DUD-E)

The DUD-E dataset is an enhanced version of the Directory of Useful Decoys (DUD) (Huang et al., 2006), seeking to address the limitations associated with the latter, some of which include molecule net formal charge imbalance within the dataset, artificial enrichment, and false negatives (i.e. decoys binding to a target). DUD-E adds the net molecular charge to the physical properties matched to identify decoys. The DUD-E contains 22,886 active compounds and their affinities against 102 targets. The actives are experimentally verified, and each one is attributed 50 decoys which have similar physio-chemical properties but different 2D structures (Mysinger et al., 2012). The data available can be downloaded either separately by specific targets, (e.g. HIV protease) or by a number of specific subsets, namely G-protein coupled receptors (GPCR), protein kinases, nuclear hormone receptors, and proteases. For the purposes of our study, we make use of the GPCR subset from the DUD-E dataset.

2.4 | Machine Learning

Machine Learning (ML) has proven to be effective in addressing complex problems, such as computer vision (Voulodimos et al., 2018) and natural language processing (Young et al., 2018). Prior to the advent of neural networks, machine learning tasks relied on handcrafted feature engineering to extract feature sets, which necessitated

²Accessed from: <https://pubchemdocs.ncbi.nlm.nih.gov/bioassays>. Last Accessed: 03 Nov 2021

prior domain knowledge. However, this has recently been revolutionised with end-to-end deep learning paradigms such as convolutional neural networks (CNNs) (LeCun and Bengio, 1995) and recurrent neural networks (RNNs) (Hochreiter and Schmidhuber, 1997). The former have been widely used to advance image, video, speech and audio processing, while RNNs have enabled breakthroughs in sequential data such as speech or text (LeCun et al., 2015). We go into these paradigms in more depth in Sections 2.4.2 and 2.4.3. Deep Learning (DL) (LeCun et al., 2015) is built around the notion of composing computational models with multiple processing layers, affording multiple levels of abstraction in the data representation. One significant benefit of DL is that it requires little domain specific engineering by hand, instead taking advantage of the computational power and volume of data available (LeCun et al., 2015). Images, text, and videos are naturally represented in Euclidean space, from which patterns and representations can be extracted through the use of deep learning. Data such as molecules in chemoinformatics, can be naturally represented as graphs. In Section 2.4.4 we go into graph machine learning in more detail.

There are three main paradigms in ML; (a) supervised $p(y|x)$, (b) unsupervised $p(x)$, and (c) reinforcement learning (Kotsiantis et al., 2007). Additionally, semi-supervised learning is a learning paradigm which incorporates two of the three main paradigms with the goal of incorporating the combination of labelled and unlabelled data to train a model (Zhu and Goldberg, 2009). When the examples used for learning are labelled, the learning is said to be supervised. For the purposes of this study, we are mainly concerned with the supervised learning paradigm. We make use of data on compounds which are known to be active or inactive/decoys against a specific biological target.

Mitchell (1997) describes machine learning as learning a set of tasks from experience by improving a designated performance measure. In a supervised learning scenario, the experience refers to the set of labelled data points $D = X, y$, where the machine learns over the data points. Processing of the data points depends on the task at hand, which could include a classification task $y = f(x)$ or a regression task $f : \mathbb{R}^n \rightarrow \mathbb{R}$, among others. The performance measure, which we elaborate more on in Section 2.4.5, is a measure of how well the model produces the expected output. In conventional supervised learning scenarios, datasets are split into training, validation and testing splits to evaluate the performance. The validation dataset is used to evaluate the performance at training time, while the test set is used to evaluate the final performance of the machine learning model. The goal here is to develop a generalisation ability, whereby the model can perform well on previously unseen data at the inference stage. In statistical learning, the independent and identically distributed (IID) assumption stipulates that the training and test sets are identically distributed and are collected from the same probability

distribution. Therefore, in theory, the expected test set error should be identical to the training set error. However, in practice, this is not always the case and underfitting or overfitting can occur. Underfitting occurs when the model is not capable of capturing the relationship between the inputs and the target values, generating an inadequate error rate. On the other hand, overfitting occurs when the model generalises too close to the training data, rendering it unable to generalise well to unseen data in the test set. Regularisation can be used to reduce overfitting by reducing the complexity of the model. It is defined as any modification to the learning approach to reduce the generalisation error, but not the training error (Goodfellow et al., 2016). Figure 2.7 visualises underfitting and overfitting intuitively for some data points.

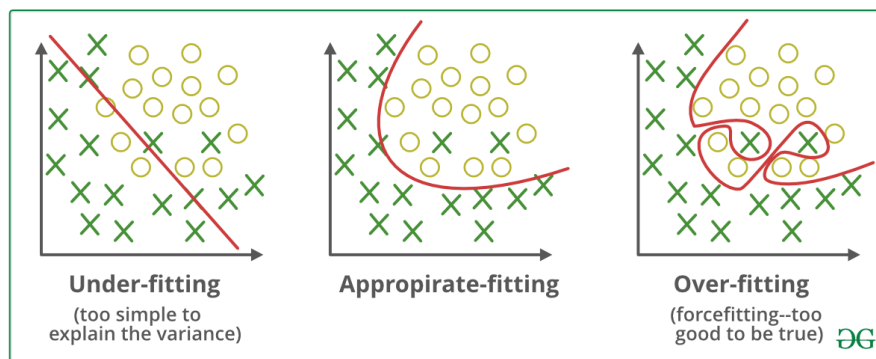


Figure 2.7: A visualisation of underfitting, overfitting and appropriately fitting a model. The circles and crosses represent two classes in the training data. Reproduced Figure.³

2.4.1 | Feed-forward Neural Networks

Inspired by the neurons in the human brain, neural networks are the cornerstone of deep learning. Neural networks consist of an input layer, hidden layers, and an output layer. Each hidden layer can consist of multiple units or nodes, also referred to as *perceptrons*. Neural networks take as input a vector of variables $x = (x_1, x_2, \dots, x_p)$ to predict a response Y by building a nonlinear function $f(X)$. In Figure 2.8, the neural network takes four features x_1, \dots, x_4 , which are connected to the hidden layer, and subsequently connected to the node in the output layer. A neural network can be defined by Equation 2.1. More hidden layers can be stacked, increasing the number of parameters and making the network deeper.

³Accessed From: <https://bit.ly/3ARNdoA> - Last Accessed: 18 October 2021.

$$f(X) = \beta_0 + \sum_{k=1}^K \beta_k A_k \quad (2.1)$$

where:

β_k = bias

K = number of nodes in the hidden layer

g = function for given hidden layer node

$f(X)$ = nonlinear function output

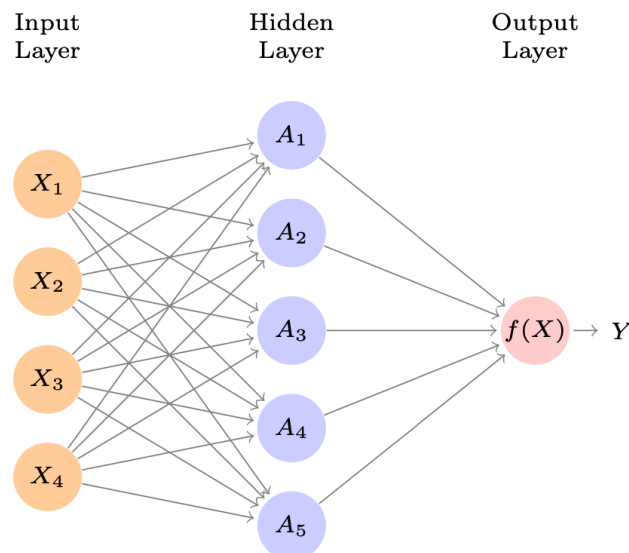


Figure 2.8: Neural network with one hidden layer and one output layer Y computed through $f(X)$. $f(X)$ takes as input the outputs from A_k . A_k is learned during training and is a nonlinear transformation of inputs from the input layer. Reproduced from James et al. (2021).

The outputs of the perceptrons in the neural network are passed through a differentiable, non-linear activation function. Four commonly used activation functions are the rectified linear activation (ReLU), leaky ReLU, Sigmoid, or hyperbolic tangent (tanh) functions (see Figure 2.9). The sigmoid function squashes values between zero and one, while the TanH function squashes values between negative one and one. The ReLU function sets any negative value to zero, and positive values between 0 and 1. In contrast, the leaky ReLU has a small slope for negative values rather than setting them all to zero. The leaky ReLU minimises the risk of a vanishing gradient, as negative val-

ues are not all set to zero. The default recommendation in machine learning for neural networks is the ReLU activation function (Glorot et al., 2011; Goodfellow et al., 2016).

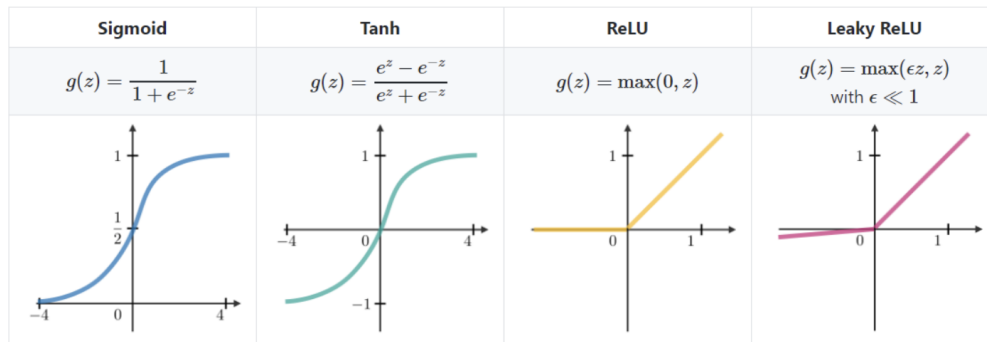


Figure 2.9: Activation function visualisations for sigmoid, tanh, ReLU and leaky ReLU functions. Reproduced Figure.⁴

Feed-forward neural networks are called as such because information flows from the input, through the hidden layers and finally to the output y (Sarle, 1994). Such networks learn through gradient descent and back propagation (Rumelhart et al., 1986). After producing an output from the forward pass through the network, a cost function calculates the loss. A commonly used function in classification tasks is the cross-entropy loss function. In a multi-class classification task, the outputs of the neural network are passed through a softmax function to represent the outputs as class probabilities. The softmax activation function ensures that the outputs are non-negative and sum to one, with the highest value signifying the highest probability of the predicted class. The discrepancy between the predicted values and the target labels is computed through the loss function. The loss is propagated back to the network to adjust the weights and guide learning towards a local minimum of the differentiable function. The goal is to minimise the cost function as much as possible while avoiding overfitting (Goodfellow et al., 2016).

2.4.2 | Convolutional Neural Networks

In this section, we provide a brief overview into convolutional neural networks (CNNs). This will provide the theoretical foundation for subsequent sections, which will dive into convolutional neural networks on graphs. LeCun and Bengio (1995) pioneered convolutional neural networks (CNNs), a kind of feed-forward neural networks, which have led to breakthroughs in computer vision (He et al., 2016; Krizhevsky et al., 2012).

⁴Accessed From: <https://bit.ly/3peskIC> - Last Accessed: 18 October 2021.

A CNN model is composed of two main concepts, namely; the convolution and the pooling operator. A kernel, which is represented from a matrix of numbers, is defined, which is iterated across the whole image using the stride parameter. The lower the stride, the denser the resulting feature maps. Figure 2.10 visualises a stride of 2 for the convolutional kernel of size 3x3 pixels. The kernel is convoluted with the range of pixel values to formulate the feature maps, which is pivotal in feature extraction. The convolution is a type of matrix operation, which can be defined by Equation 2.2 (Goodfellow et al., 2016).

$$(I * K)(i, j) = \sum_{k=0}^m \sum_{l=0}^n I(i+k-1, j+l-1)K(k, l) \quad (2.2)$$

where:

I = Input image

K = kernel

O = output

m = Rows of kernel

n = Columns of kernel

i = Row index of output

j = Column index of output

The pooling operation is applied to the resulting feature maps after the convolution operation. In Figure 2.10, max pooling is utilised which takes the maximum value from a defined range of values. Pooling effectively down-samples the feature maps to reduce overfitting (Li et al., 2021).

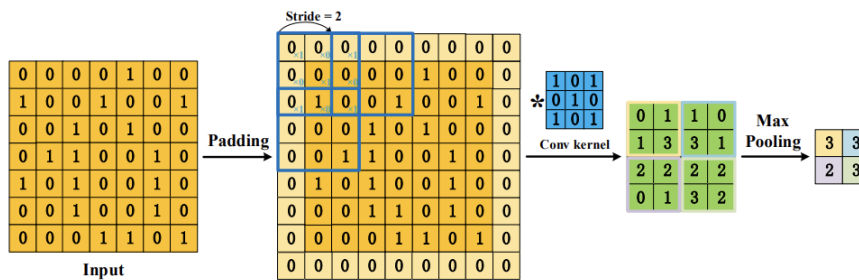


Figure 2.10: Procedure of a CNN for a 2D input, visualising the convolutional kernel with a stride of 2 pixels, and the pooling operation. Reproduced from Li et al. (2021).

2.4.3 | Recurrent Neural Networks

Recurrent neural networks (RNNs) are an extension to feedforward neural networks for processing sequential data. RNNs integrate feedback connections so the outputs of each time step have recurrent connections (Goodfellow et al., 2016). Rather than taking as input a fixed-sized input, RNNs allow us to learn over sequences of inputs, where each input is not just influenced by itself, but by the history of inputs that are fed to the network. RNNs have an internal hidden state that is updated with every step and is propagated for the following input, thus acting as memory for the network (Schuster and Paliwal, 1997). Figure 2.11 illustrates this flow, where each step, or module, is a neural network.

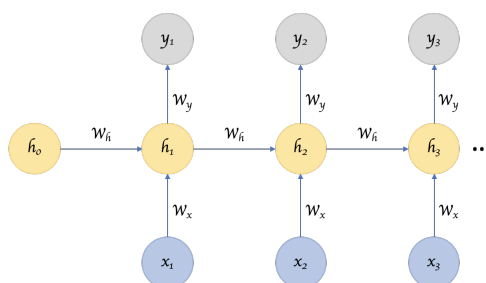


Figure 2.11: Recurrent Neural Network with hidden state, persisting and carrying relevant information from one step to the other. Reproduced Figure.⁵

An innovation in RNNs is the Long Short-Term Memory (LSTM) networks, pioneered by Hochreiter and Schmidhuber (1997). LSTMs, as the name implies, are able to handle and learn long-term dependencies. LSTMs, similar to RNNs, have repeating modules, but each module has four main elements rather than one neural network (refer to Figure 2.12). LSTM units are composed of modules with corresponding input gates, output gates and forget gates, which regulate data flowing in and out of the cell while updating the hidden state. The *forget gate layer* removes information from the cell state. The next step in the LSTM module is the *input gate layer*, which updates the state. The final step is the computation of the output layer, which is based on the cell state. LSTMs are of significant importance for this study, as the state of the art work for few-shot learning in drug discovery makes use of LSTM architectures to process the molecular embeddings.

⁵Accessed From: <https://bit.ly/3j96sUY> - Last Accessed: 18 Oct 2021.

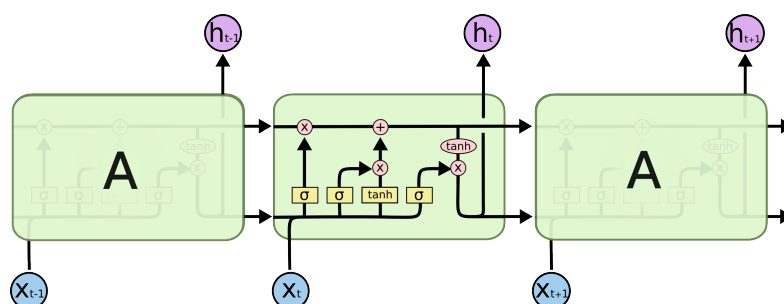


Figure 2.12: Repeating modules in LSTMs, containing four neural networks (represented by yellow boxes). Reproduced Figure.⁶

2.4.3.1 | Attention

RNNs can memorise sequences, however this might not be done in a way that benefits learning. Attention, initially proposed by Graves et al. (2014), focuses on a subset of the provided information. Attention is differentiable, allowing the model to learn what to focus on, constraining read and write operations to a small portion of the memory, rather than considering all of it. Attention has been used successfully in applications such as machine translation (Bahdanau et al., 2014) and voice recognition (Chan et al., 2016).

2.4.4 | Graph Neural Networks (GNNs)

As information for several applications can be naturally represented as graphs, the necessity to extend deep learning techniques for graph data lead to the emergence of Graph Neural Networks (GNNs). Some examples of data that can be naturally represented as graphs include social networks, citation networks, knowledge graphs, and molecular data. We make use of GNNs in our study to embed the molecule representations (explained in further detail in Section 2.6.1) in latent space, before processing further by the few-shot learning architectures, which will be presented in Section 2.5. GNNs can be used to predict either **node**, **edge**, or **graph** level outputs (Wu et al., 2020). For the purposes of this study, we will be working with graph-level classification, as we are concerned with the molecular activity of molecules in an experimental assay. Each molecule is represented as a single graph, and its class label describes the molecular activity associated with it (see Section 2.6.1.3). We go into further detail about graphs in the next section. Wu et al. (2020) propose the following taxonomy for GNNs, categorising them into four groups, namely:

⁶ Accessed From: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/> - Last Accessed: 18 October 2021.

- **Recurrent Graph Neural Networks (RecGNNs)** were first proposed by Gori et al. (2005), which extend upon Recurrent Neural Networks (RNNs) and fall into the Recurrent GNNs category. Node representations are learned through an iterative process by propagating neighbouring information until an equilibrium is reached, however this is computationally expensive.
- **Convolutional GNNs.** Through the use of convolutional operations, each node's representation is generated through aggregation of its own features, and its neighbouring node features. In contrast to RecGNNs, these neural networks stack multiple graph convolutional layers to achieve a high level node representation. As our study is primarily concerned with this category of neural networks, we go into further detail in subsequent sections.
- **Graph Autoencoders** are unsupervised learning frameworks, encoding nodes or graphs into latent vector space and subsequently reconstructing the graphs.
- **Spatial-temporal GNNs**, as the name implies, consider both spatial and temporal data. This is particularly useful in applications where the temporal and spatial dependencies are correlated, such as in human action recognition (Yan et al., 2018).

2.4.4.1 | Graphs

A graph is formally represented as a set of nodes and edges, as shown in Equation 2.3. Graphs are named as such as they can be conveniently represented graphically as 2D data structures (Bondy and Murty, 1976). Graphs can be either homogeneous or heterogeneous. In the former, the nodes are all of the same type, which is in contrast to heterogeneous graphs which can include nodes and edges of different entity types. For this study, we are concerned with homogeneous graphs, as nodes representing the molecular graph are of the same atom type.

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}) \quad (2.3)$$

where:

V = set of vertices/nodes v

E = set of edges e_{ij} , where each edge e connects pairs of nodes (v_i, v_j) in V

The neighbourhood of a node $N(v)$ is the set of nodes connected to a node v_i , and can be defined as $N(v) = u \in V | (v, u) \in E$. In graph processing, the adjacency matrix A is a $n \times n$ matrix describing node connectivity. n is the number of nodes in a graph.

If nodes v_i and v_j are connected such that $e_{ij} \in E$, $A_{ij} = 1$, and $A_{ij} = 0$ if $e_{ij} \notin E$. The adjacency matrix is symmetric, such that $A = A^T$, if the graph is undirected, meaning that there is no explicit direction between two connected nodes resulting in an inverse direction of the pair of edges between them. Each node can have a feature vector x_v of size d , which makes up the graph feature matrix X . Similarly, each edge can have edge attributes, represented in the edge feature matrix X^e (Wu et al., 2020).

2.4.4.2 | Elements of Graph Deep Learning

If the graph object is our input signal, we can apply a set of operators for the function we are attempting to learn. Bronstein et al. (2021) propose four key building blocks for deep learning on graphs, which include linear set equivariant layers, non-linear functions, local pooling layers and set invariant layers (see Figure 2.13). For graphs, the nodes v are found on a domain Ω such that $v \in \Omega$. The nodes in Ω are stored in a feature space C , such that $C = \mathbb{R}^k$. Using a set of feature functions $X(\Omega, C)$, we can transform the feature space of the nodes in our domain.

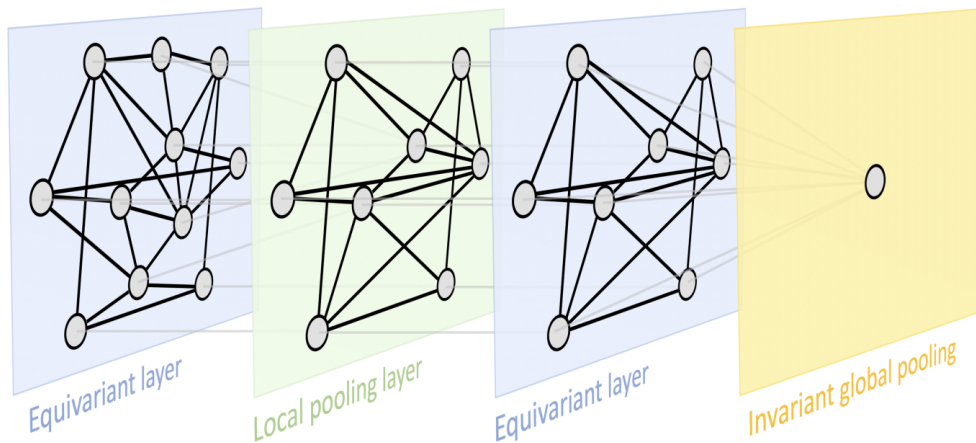


Figure 2.13: The main components in graph deep learning, including permutation equivariant, local pooling, and permutation invariant global pooling layers. Non-linearity is not visualised, which is also a key component. Reproduced from Bronstein et al. (2021).

In the equivariant layer B , we can take the nodes in our domain and apply a function that transforms the features of the nodes such that $X(\Omega, C) \rightarrow X(\Omega', C')$. Equivariance allows for a function g to be applied before or after this layer, such that $B(g.x) = g.B(x)$. The non-linear activation functions can be applied element-wise on the features of the nodes in a graph, such that $(\sigma(x))(v) = \sigma(x(v))$. Local pooling layers can be used to apply coarsening to the graph such that $X(\Omega, C) \rightarrow X(\Omega', C)$, in which we can reduce

the number of nodes in our domain such that $\Omega' \subseteq \Omega$. Finally, we have the invariant layer Z , which can also be referred to as a global pooling layer, in which $X(\Omega, C) \rightarrow y$, which satisfies the invariant condition such that $Z(g.x) = Z(x)$ (Bronstein et al., 2021).

2.4.4.3 | Graph Convolutional Networks

Inheriting ideas of message passing, graph convolutional networks (GCNs) are closely related to RecGNNs, but effectively address the graph mutual dependencies problem (Micheli, 2009) through the use of layers with different weights. Figure 2.14 shows how images can be represented as graphs, where each pixel is connected to nearby pixels. A 3×3 filter is used to convolve the central node's representation with neighbouring nodes captured within the filter. Node information is propagated along edges, based on the notion of message passing neural networks (Wu et al., 2020). In contrast with images, molecular graph representations are irregular and nodes do not have an explicit order.

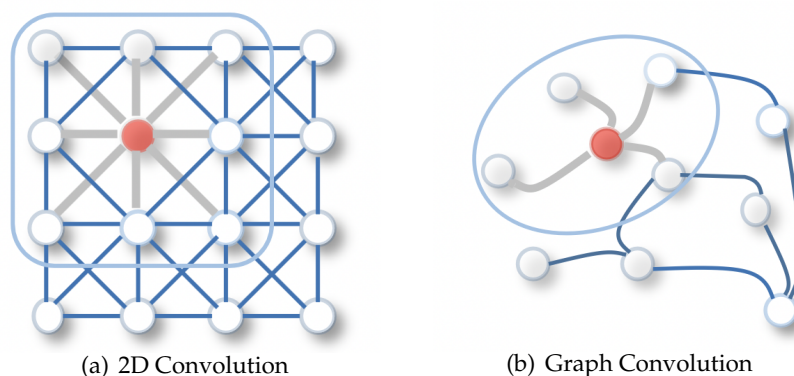


Figure 2.14: 2D vs Graph Convolutions. (a) illustrates that in 2D convolutions, each pixel is analogous to a graph with neighbouring pixels determined by the filter size. The neighbours are ordered and are of a fixed size. (b) In contrast, node neighbours in graphs are irregular as they can vary in size and the order is arbitrary. Reproduced from Wu et al. (2020).

The goal of these models is to learn a function for a graph G , using the feature matrix C and the adjacency matrix A to produce a node level output. This function is equivalent to the equivariant layers explained in the previous section and are synonymous with graph convolutional operators (see Figure 2.15). Local pooling operations (see Section 2.4.4.4) are used to reduce dimensionality and global pooling operations, also referred to as readout functions, are used to compute graph level outputs (see Section 2.4.4.5).

Bruna et al. (2013) first introduced the concept of spectral GNNs, combining concepts from graph signal processing (Sandryhaila and Moura, 2013) such as spectral anal-

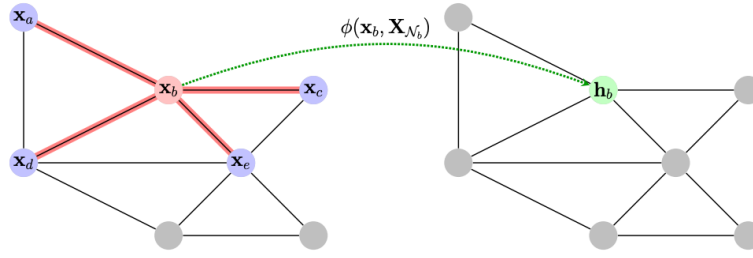


Figure 2.15: Equivariant function in a graph, synonymous with a graph convolutional operator over neighbouring nodes. Reproduced from Bronstein et al. (2021).

ysis with Convolutional Neural Networks (CNNs). A signal x can be represented as a vector $x \in \mathbb{R}^n$. The graph Laplacian (Chung and Graham, 1997) is an essential operator in spectral graph analysis. Equation 2.4 is the definition of the normalised version of the graph Laplacian matrix L , which is composed of a set of orthonormal eigenvectors u_l (graph Fourier modes) and the associated real non-negative eigenvalues λ (graph frequencies).

$$L = I_n - D^{-1/2}AD^{-1/2} \quad (2.4)$$

where:

D = diagonal matrix of node degrees

A = graph adjacency matrix

I = identity matrix

The normalised Laplacian matrix L is formalised by Equation 2.5.

$$L = U\Lambda U^T \quad (2.5)$$

where:

U = Matrix of eigenvectors ordered by eigenvalues $U = [u_0, \dots, u_{n-1}] \in \mathbb{R}^{n \times n}$

Λ = Diagonal matrix of eigenvalues (spectrum) $\Lambda = [\lambda_0, \dots, \lambda_{n-1}] \in \mathbb{R}^{n \times n}$

I = identity matrix

The eigenvectors of this matrix form orthonormal space $U^T U = I$. The graph Fourier transform of signal x , which in graph signal processing represents the node in a graph through feature vectors, is defined as $\mathcal{F}(x) = U^T x \in \mathbb{R}^n$ (Shuman et al., 2013). The inverse of the graph Fourier transform is $\mathcal{F}^{-1}(\hat{x}) = U\hat{x}$. The graph Fourier transform projects x to the orthonormal space. A spectral graph convolution is in essence the

multiplication of a signal x with a filter $g \in \mathbb{R}^n$ in the Fourier space of a graph, which is formalised in Equation 2.6

$$x *_G g = \mathcal{F}^{-1}(\mathcal{F}(x) \odot \mathcal{F}(g)) \quad (2.6)$$

$$x *_G g = U g_\theta U^T x \quad (2.7)$$

Substituting the values in the definition of the graph Fourier transform we get $U(U^T x \odot U^T g)$. We can denote the filter g as $g_\theta = \text{diag}(U^T g)$. The spectral graph convolution can thus be formalised in Equation 2.7. The difference between different graph convolutional networks is the filter g_θ , but they all follow this same definition (Wu et al., 2020). In Bruna et al. (2013), it is assumed that the filter g is a set of learnable parameters.

The work by Bruna et al. (2013) is domain dependent, meaning the filters cannot be transferred to other graphs if these have different structures. Additionally, any changes in the graph require a recomputation of the eigenbasis, and the eigendecomposition is computationally expensive, with complexity of $O(n^3)$. As a result, Defferrard et al. (2016) propose ChebNet, in an attempt to find a compromise between the slow and more structured spectral approach, with faster heuristics by approximating smooth filters in the spectral domain. The computation complexity is reduced to $O(m)$ by utilising localised filters to extract local features independently of graph size. Kipf and Welling (2016) propose a first-order approximation of ChebNet. The graph convolution is formulated using the propagation rule which is formalised in Equation 2.8.

$$f(H^{(l)}, A) = \sigma \left(\hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{-\frac{1}{2}} H^{(l)} W^{(l)} \right) \quad (2.8)$$

where

- A = Adjacency matrix
- \hat{A} = $A + I$, in which I is the identity matrix to include self-loops
- \hat{D} = diagonal node degree matrix of \hat{A}
- $\sigma(\cdot)$ = non-linear activation function
- W = weight matrix for the l th layer
- H = neural network layer
- f = non-linear function

In Equation 2.8, the current node is included in the aggregation of neighbouring nodes through the addition of the identity matrix to the adjacency matrix to form \hat{A} . The diagonal node degree matrix D is used to normalise A to preserve the scale of feature vectors, such that $D^{-1}A$, which is equivalent to averaging neighbouring node features. This is further enhanced through symmetric normalisation of A through $D^{-1/2}AD^{-1/2}$,

which gives us the propagation rule in Equation 2.8 (Kipf and Welling, 2016). The convolutional layer can finally be mathematically defined through Equation 2.9.

$$h_i^{(l+1)} = \sigma(b^{(l)} + \sum_{j \in \mathcal{N}(i)} \frac{1}{c_{ji}} h_j^{(l)} W^{(l)}) \quad (2.9)$$

where

h_j = feature set of nodes

N_i = set of neighbouring nodes i

c_{ji} = product of the square root of node degrees

b = learnable bias

σ = non-linear activation function

From a message-passing perspective, the above can be summarised into the following steps for every node feature space u ;

1. Aggregating the neighbouring representations h_v , producing an intermediate representation \hat{h}_u .
2. Transforming \hat{h}_u through a linear projection and finally through a non-linearity function such that $h_u = f(W_u \hat{h}_u)$ (Kipf and Welling, 2016).

In our study, we make use of the convolutional operator from Kipf and Welling (2016) to process our graphs. Figure 2.16 is a graphical representation of an example of a graph convolution network. The node in blue is the one on which the operation is being performed.

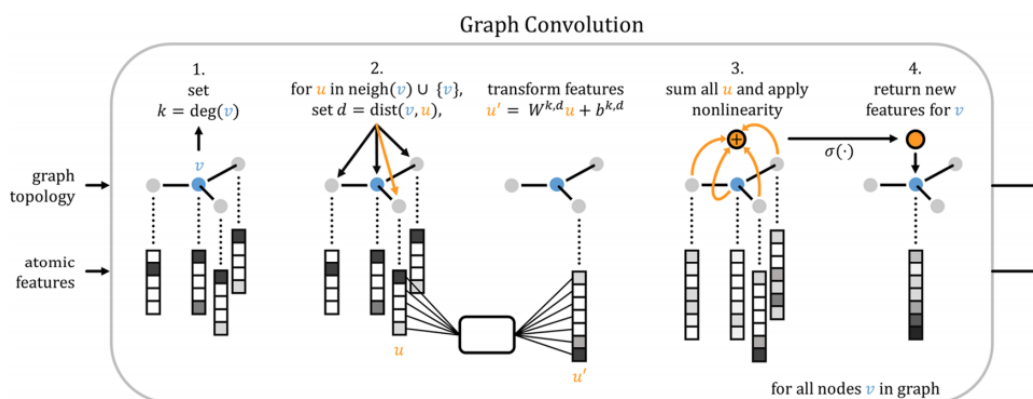


Figure 2.16: Graph Convolutions. Reproduced from Altae-Tran et al. (2017).

2.4.4.4 | Local Pooling Layer

Once node features are created using GNNs, local pooling can be used to reduce the dimensionality of the graph node features, reducing the probability of overfitting and computational complexity issues (Wu et al., 2020). Pooling essentially coarsens the graph by creating smaller node representations or by reducing the number of nodes in our domain as explained in Section 2.4.4.2. Commonly used operations are mean, max or sum pooling as these are fast operations (Equation 2.10). Henaff et al. (2015) highlight the importance of performing a simple max or mean pooling operation at the beginning of the network to attenuate the computational cost of graph Fourier transforms. In Figure 2.17, we visualise a pooling operator that coarsens the graphs. In our study, we do not coarsen the graph using local pooling layers. Between graph convolution layers, we simply apply a linear max function to aggregate the maximum node features for the current node from its neighbouring nodes.

$$r^{(i)} = \text{mean}/\text{max}/\text{sum}(h_n^{(i)}) \quad (2.10)$$

2.4.4.5 | Global Pooling Layer

As explained in Section 2.4.4.2, we can use a permutation-invariant global pooling layer in a GNN. This layer is also referred to as a readout layer in literature (Wu et al., 2020). The readout layer is used to generate graph-level representations after the node features have been transformed using convolutional and local pooling layers. The readout layer is typically used at the end of the GNN pipeline as the node representations are collapsed into a vector representing the graph (Wu et al., 2020). This vector can be further processed by multilayer perceptrons (MLP) to classify the final output as shown in Figure 2.17.

In their review, (Wu et al., 2020) point out that methods to improve pooling operation's effectiveness and computational complexity is still an open research question.

2.4.4.6 | Challenges in Graph Machine Learning

Graphs pose a challenge for machine learning algorithms due to their complexity as the number of neighbouring nodes per node can differ and have no particular order. This differs from other domains such as image processing, where each pixel has a fixed number of ordered neighbouring pixels. In fact, images are analogous to graphs as they are made up of pixels connected to adjacent pixels. However, graph data can be irregular with a differing number of neighbouring nodes per node, in no particular order. A fun-

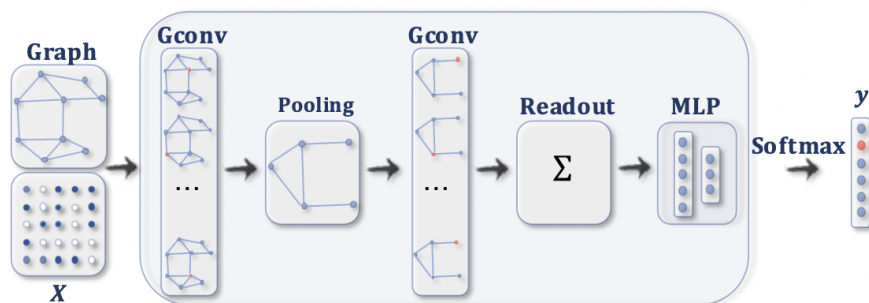


Figure 2.17: Convolutional GNN with pooling, readout, Multilayer perceptron (MLP) and softmax layer for graph classification. The pooling layer coarsens the graph, while the readout layer reduces the graph representation into a vector. Reproduced from Wu et al. (2020).

damental assumption in machine learning algorithms is the independence of distinct data from each other, such as different images. However, given the interconnection of nodes in a graph network, this assumption no longer holds. However, this assumption does not pose a challenge to our study as molecules are represented as distinct graphs, rather than one large graph.

2.4.5 | Evaluation Metrics

The most commonly used basic way to evaluate the performance of a classifier is through the accuracy metric (Equation 2.11). However, the accuracy metric is commonly not representative of the true performance, as it does not provide reliable insight into the performance in multi-class problems or when using imbalanced datasets. A confusion matrix is thus used to summarise the output from a classifier. As the name implies, it shows how the classifier ‘confuses’ predictions. The confusion matrix provides the true positives and negatives, and the false positives and negatives.

In this study, the data is highly imbalanced (refer to Section 2.3). If we rely purely on accuracy, the model can simply predict the most common class, which in this case is the negative class (i.e. the inactive/decoy class). As we are more interested in the positive class (i.e. the active class), a more robust representation is required. Sensitivity (recall) (Equation 2.12) is equivalent to the true positive rate, and specificity (Equation 2.13) is equivalent to the $1 - FalsePositiveRate$. The precision is the positive predictive power, meaning how well the positive class is predicted.

$$accuracy = \frac{TP + TN}{TP + TN + FN + FP} \quad (2.11)$$

		ACTUAL VALUES	
		Positive	Negative
PREDICTED VALUES	Positive	TP	FP
	Negative	FN	TN

Figure 2.18: Confusion Matrix

$$\text{sensitivity} = \frac{TP}{TP + FN} \quad (2.12)$$

$$\text{specificity} = \frac{TN}{TN + FP} \quad (2.13)$$

$$\text{precision} = \frac{TP}{TP + FP} \quad (2.14)$$

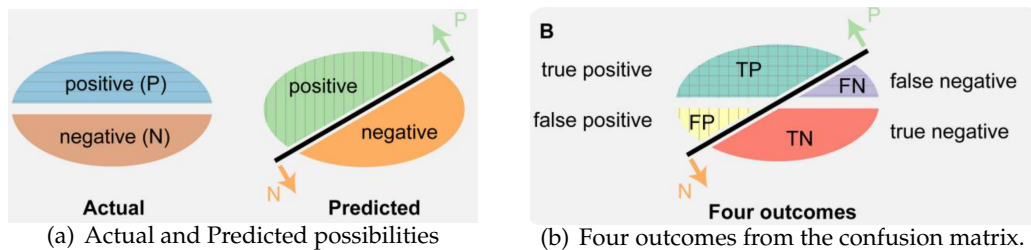


Figure 2.19: Confusion Matrix Visualisation. (a) illustrates the actual and predicted values. Outcomes from predicted values can take on four values, represented via a confusion matrix. Reproduced from Saito and Rehmsmeier (2015).

These measures are single-threshold measures, which are based off cutoffs for a classifier into positive and negative predictions. Threshold-free measures such as the Receiver Operator Characteristic (ROC) and Precision-Recall Curve (PRC) plots are more powerful as they produce scores that are used to divide datasets into positively and negatively predicted classes, rather than provide a static division. The ROC plot shows the relationship between sensitivity and specificity. The ROC plot for random classifiers is a diagonal line from the origin (0,0) to (1, 1), which is the baseline for the ROC curve. The ROC Area under the Curve (AUC) score is 1.0 for perfect classifiers and 0.5 for random

classifiers. The ROC is a well-renowned measure, however, the PRC is better suited for imbalanced datasets (Fawcett, 2006).

Fawcett (2006) explore the performance of the PRC evaluation method against the ROC in imbalanced bioinformatics datasets, and the authors strongly recommend the use of PRC in such scenarios as it provides a more accurate and intuitive interpretation. It is also reported that despite these findings, many studies still use the ROC to evaluate performance on imbalanced datasets. The PRC plot visualises the relationship between precision and recall. The baseline, unlike the ROC plot, is not fixed but moves accordingly with class distribution. Therefore, the PRC AUC achieves better evaluations for predicting the positive class.

2.5 | Learning with Low Data

Humans have an innate capacity to learn new concepts from just a few examples (Lake et al., 2011). For instance, a person seeing a Segway for the first time is able to differentiate future encounters of Segways from other vehicles (Lake et al., 2015). In contrast, conventional supervised end-to-end machine learning is data hungry, and cannot generalise using a few examples.

Inspired by human learning (Lake et al., 2015), few-shot learning makes use of data from similar tasks to compensate for the lack of data for the task at hand. Several successful research programs have exploited this paradigm (Koch et al., 2015; Snell et al., 2017; Sung et al., 2018; Vinyals et al., 2016), in which a model learns a similarity measure from image embeddings from tasks which are similar, but not identical to the task at the inference stage. Thus, few-shot learning (Fei-Fei et al., 2006) is a machine learning paradigm that aims to bridge this gap between this human learning ability and machine learning. Learning to learn (Thrun and Pratt, 2012) is referred to as meta-learning (Finn et al., 2017). Being able to learn from only a few examples is important as certain domains do not have access to the plethora of data that we see in other domains such as computer vision. This inaccessibility could be due to privacy, safety, or ethical issues. For instance, data acquisition can be problematic in the drug discovery domain due to possible toxicity, low activity or solubility in clinical candidates. Learning with less data leads to less expensive data gathering and computational cost for learning (Wang et al., 2020).

2.5.1 | Problem Definition

This section will introduce commonly used terms and nomenclature in the few-shot learning paradigm. Few-shot learning can be characterised as metric-learning or non-parametric meta-learning (Hospedales et al., 2020). Metric learning refers to learning a distance function over data samples to differentiate between data samples. For instance, in the image domain we can train a CNN to embed an image as a vector, which is subsequently compared to other embeddings to infer a class prediction.

The meta-learning paradigm differs from conventional supervised end-to-end learning, where in the latter data points from k classes are used to train a model that can identify unseen objects that belong to the same classes. In the meta learning paradigm, we have a meta-training and meta-testing stage.

2.5.1.1 | Meta-learning

Figure 2.20 illustrates a typical meta-learning scenario in the image domain. The different stacks of images represent different classes. A subset of classes are reserved for training, and the rest for testing. Training consists of a number of epochs, each composed of a sequence of episodes. Each episode consists of a **k-way n-shot** scenario. The k represents the number of classes to sample, while n is the number of data points to sample per class. For example, a 3-way 1-shot meta-learning scenario consists of 3 different classes, where each episode consists of 1 example per class. During training, the sampling of these data points make up the support set $S = x_i, y_i$. A set Q of q query data points are sampled from the remaining data points in the sampled task. The model learns a differentiable function to classify the query data points using only the support set. This simulates the conditions at the testing stage, where the classification of a new unseen class is *learned* using only n data points from the new class as the support set. For example, we can learn to classify images of animals in a few-shot scenario, and then correctly classify photos of a new unseen class during training (e.g. an image of a lion), using only a few (e.g. five images). The model therefore *learns* to classify images of lions using only five examples. This capability is beneficial in domains where it is difficult to obtain data for new classes, but for which we already have ample data for other classes or tasks. While training simulates the few-shot conditions at test time, training still necessitates a lot of data from classes for which we already have data about.

In the chemoinformatics domain, instead of different classes we have different tasks for an assay. For example the Tox21 dataset (refer to Section 2.3.1) contains 12 different tasks. Each task consists of a binary classification problem where molecules are classi-

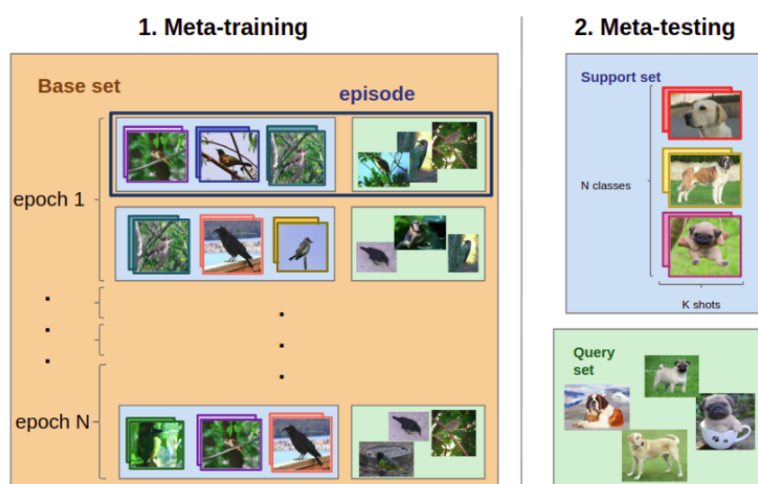


Figure 2.20: Meta-learning consisting of meta-training and meta-testing. The classes/tasks in the meta-training and meta-testing phases are non-overlapping. The light blue boxes represent the support sets, while the green boxes represent the query sets. Reproduced from Bennequin (2019).

fied as either active molecules or inactive. Therefore, the **way** value is always *two* for this study as we only have the active, and the inactive class.

A number of successful research undertakings (Finn et al., 2017; Koch et al., 2015; Lake et al., 2015; Snell et al., 2017; Sung et al., 2018; Vinyals et al., 2016) have exploited the meta-learning paradigm for the image domain. One-shot learning was first introduced by Fei-Fei et al. (2006). It is worthwhile to note that the undertaken studies for the state of the art research papers in few-shot learning are focused on the image domain, for which the authors could make use of image augmentation to artificially increase the size of the datasets through the use of affine transformations such as skewing, rotating or stretching the images. These techniques help reduce overfitting and allow the model to be more generalisable to unseen data. However, this cannot be done for molecular data as small changes to the molecule can lead to completely different properties. This study will focus mainly on Siamese Networks (Koch et al., 2015) and Matching Networks (Vinyals et al., 2016) in line with related work in the drug discovery domain (Altae-Tran et al., 2017), which we develop further by applying prototypical (Snell et al., 2017) and relation (Sung et al., 2018) networks to the problem domain.

2.5.2 | Siamese Networks

Siamese Networks were first proposed by Bromley et al. (1993) to solve a document signature verification problem. A Siamese neural network is composed of twin neural

networks with shared weights. Such a network takes as input a pair of inputs, which are embedded separately through each neural network. As the neural networks share weights, the feature extraction is maintained to the same feature space for both inputs. These identical sub-networks are finally connected in a final layer that acts as a distance function for the two outputs. If the distance falls within a pre-determined threshold, the signature passes as a match to the original one, and if not, marked as a fake. Siamese Networks are always fed in pairs of inputs to compute the distance metric that identifies how similar the samples are to each other. In fact, Siamese Networks do not learn in the conventional end-to-end supervised learning approach. Instead, the model is trained to differentiate between examples and will identify if two inputs belong to the same class. The neural networks are trained to lift meaningful features to the feature space.

Siamese Networks have managed to achieve close to human accuracy on the Omniglot dataset (Koch et al., 2015). The Omniglot dataset is made up of images of characters from alphabets from all around the world. One character from each alphabet was extracted for training. A common practice when training with images is to apply affine transformations to augment the dataset and increase the training samples. However, results without these distortions are also presented by Koch et al. (2015) and the accuracy merely drops from 93.42% to 91.63%. These results confirm that there is potential for ML in low-data scenarios. Figure 2.21 illustrates a schematic of how Siamese Networks can be used for molecular data, starting from the molecular representation generation, passing this to a neural network and then finally a distance function.

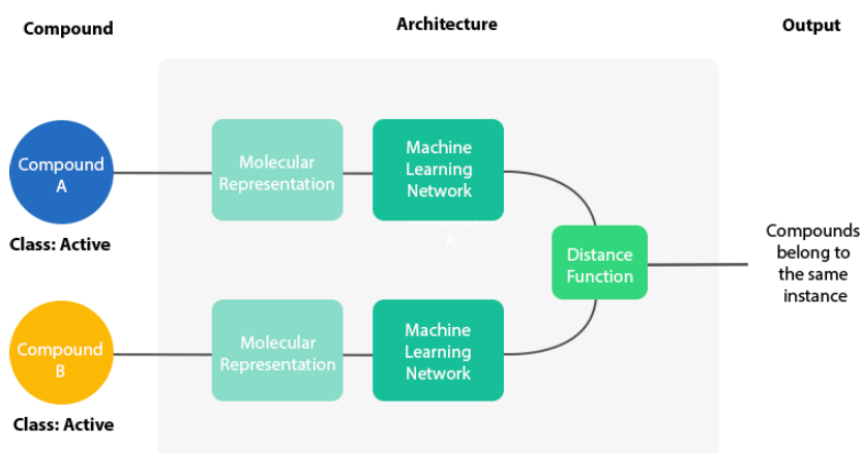


Figure 2.21: High level schematic of a Siamese network for molecular network.

2.5.3 | Matching Networks

Vinyals et al. (2016) propose Matching Networks, a non-parametric approach that draws inspiration from attention (Bahdanau et al., 2014) and memory networks (Lake et al., 2015). This technique is applied to ImageNet (Deng et al., 2009) datasets for classifying images, and the Penn Treebank (Santorini, 1990) dataset for one-shot language modelling. In their research, the authors stipulate a training precept where training and testing conditions must match. This implies that training for rapid learning must emulate the same conditions as those during testing time. The authors remark that one-shot learning is more effective if you train to perform one-shot learning, rather than train on more samples to generalise the model further. This training method is achieved through the use of *episodes* during training, where each training episode emulates the test conditions by sub-sampling classes and the data points within each class to create a mini-batch. The model is presented with n examples from k classes that will be used to infer the query's class. Matching Networks are able to predict new, unobserved classes during training without any changes to the network by training a classifier that can learn from a few number of examples. Using a support set S containing n labelled examples, the goal is to estimate the probability that a query example \hat{x} belongs to a given class by taking the argmax $\operatorname{argmax}_y P(\hat{y}|\hat{x}, S)$. The attention mechanism in Equation 2.15 specifies how similar \hat{x} is to each example x in S .

$$\hat{y} = \sum_{i=1}^n a(\hat{x}, x_i) y_i \quad (2.15)$$

where

\hat{y} = output label

a = attention mechanism

\hat{x} = query example

Figure 2.5.3 illustrates the Matching Nets architecture. Embedding functions f and g are CNNs, potentially being identical to each other, which lift the inputs to the feature space. The authors also propose full context embedding functions, which take as input the whole support set with the element x_i , thus resulting in $g(x_i, S)$. Full context embeddings effectively modify how the element is embedded with respect to the whole support set S . A bidirectional Long-Short Term Memory (LSTM) is used to encode x_i in the context of the support set. The attention mechanism a , at the end of the pipeline, is the classifier. This mechanism takes a softmax over the cosine distance of the embeddings. The matching nets outperformed memory augmented neural network and convolutional Siamese Networks in all instances.

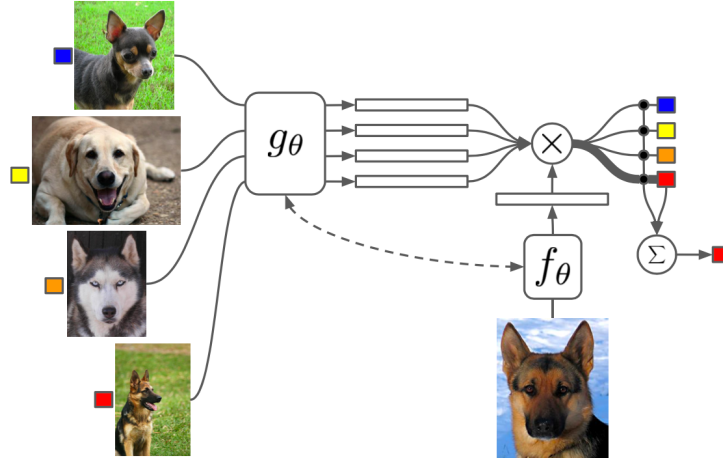


Figure 1: Matching Networks architecture

Figure 2.22: Matching Networks Architecture. Reproduced from Vinyals et al. (2016).

2.5.4 | Prototypical Networks

Snell et al. (2017) build upon the work of Vinyals et al. (2016), proposing Prototypical Networks to address the few-shot classification problem. The authors show that Prototypical Networks afford the same level of performance as Matching Networks, but with simpler design choices. Prototypical Networks learn a metric space, where classification is performed by computing the distances between *prototype* representations of each given class and finding the closest one. This method reflects a simpler inductive bias, which is reported to be beneficial in the limited-data regime.

Prototypical Networks are based on the notion that an embedding exists in which points cluster around a prototypical representation for each provided class. Data points are embedded using a neural network, and the prototype c_k is defined by taking a mean of the support set in the resulting embedding space as shown in Equation 2.16.

$$c_k = \frac{1}{|S_k|} \sum_{(x_i, y_i) \in S_k} f_\phi(x_i) \quad (2.16)$$

where

f_ϕ = embedding function with learnable parameters ϕ

S_k = support set with k examples

In a one-shot learning scenario, Prototypical Networks are equivalent to Matching Networks. However, in a few-shot scenario there are two main design differences. The

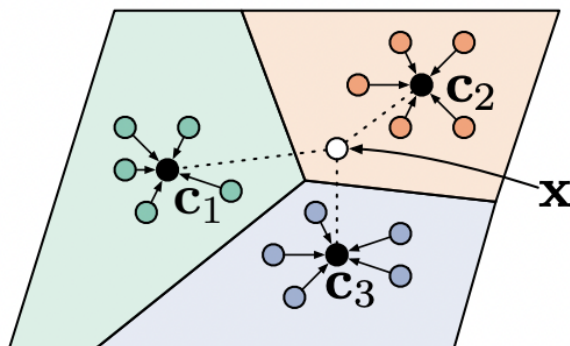


Figure 2.23: Few-shot learning scenario in Prototypical Networks where prototypes c_k are taken as the mean of embedded support examples for each class. Reproduced from Snell et al. (2017).

squared Euclidean distance is used instead of the cosine distance used in Matching Networks. Snell et al. (2017) find that it is beneficial to train with a higher number of classes than will be used to infer the queries at test time. This does not apply for the number of samples selected for each class, as it is observed that these should match the test conditions. Therefore, this episodic adaptation applies only to the *way* number. However, this adaptation is not applicable for this study as we are always concerned with a 2-way scenario as our problem is formulated as a binary classification problem. Using these adaptations, Prototypical Networks achieve better results in the reported experiments in the literature than Matching Networks (Vinyals et al., 2016).

2.5.5 | Relation Network

Sung et al. (2018) present the Relation Network, a framework for few-shot learning, which could also be extended to zero-shot learning. The Relation Network learns a non-linear distance metric to compare support and query examples. As opposed to siamese, matching, and Prototypical Networks, this network uses a feed-forward neural network to calculate the distance in feature space. After embedding the support and query examples through an embedding function, each query example is concatenated with each of the feature maps as illustrated in Figure 2.24. The resulting feature map concatenations are processed using a convolutional neural network to output a relation score vector, from which the class can be inferred. The authors utilise a mean-squared error (MSE) objective function, regressing the relation score to the target label. Despite the problem being a binary classification one, where the target label is a label from the

set 0,1, the authors utilise an MSE function rather than cross entropy. They postulate that predicting the relation score can be considered a regression problem.

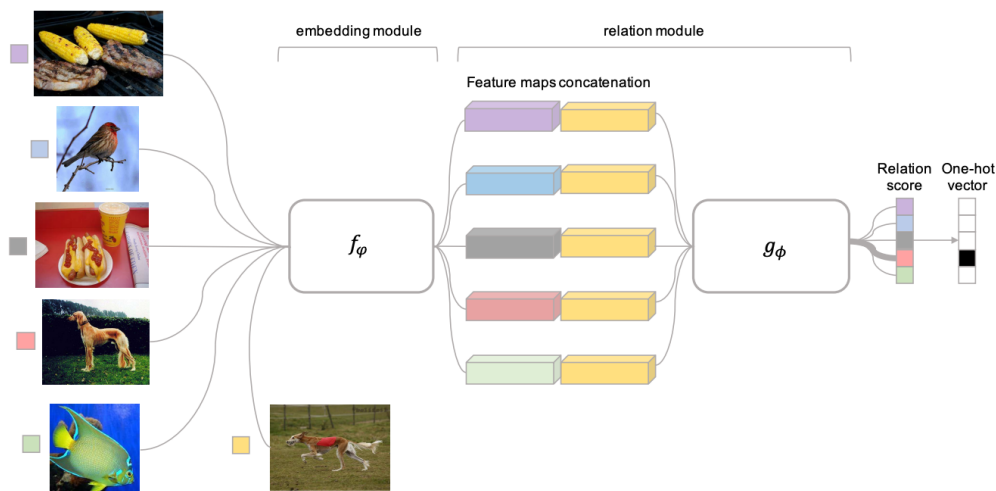


Figure 2.24: Few-shot learning scenario in Relation Networks for a 5-way 1-shot learning task with one query as an example. Reproduced from Sung et al. (2018).

2.6 | Molecular Machine Learning

Wu et al. (2018) remark that developing machine learning models for molecular data comes with a number of issues, namely the limited amount of data, the large heterogeneity in molecular structures, and choosing appropriate learning algorithms. The authors propose MoleculeNet, a benchmark for machine learning models on molecular data. Wu et al. (2018) experiment with a number of molecular representations and machine learning models. We go in further detail into molecular representations in subsequent sections. Their machine learning models include graph convolutional networks, gradient boosted trees, random forests, kernel ridge regression and message passing neural networks. However, they conclude that graph-based algorithms perform best overall, with the exception of highly complex tasks in which data is scarce, or in datasets in which there is heavy data imbalance. In such scenarios, conventional machine learning algorithms such as support vector machines or random forests perform better than graph-based models. For the Tox21 dataset, Wu et al. (2018) report that the best model is the graph convolutional one with a ROC-AUC score of 0.829. Meanwhile, for the MUV dataset, a multitask model obtained the best results with a ROC-AUC value of 0.184.

Mayr et al. (2018) compare the performance of a number of machine learning methods across different molecular descriptors. The authors considered Support Vector Machines (SVMs), Random Forests (RFs), Naive bayes (NB) statistics, similarity ensemble approaches (SEA), and K-nearest-neighbour (KNN), along with three types of deep learning methods: (i) feed-forward neural networks, (ii) graph-based neural networks (convolutional based), and (iii) recurrent neural networks. Experiments were conducted on the ChemBL dataset (Gaulton et al., 2017), which is a large benchmark dataset containing around 500,000 compounds and more than 1,000 assays. This particular dataset was chosen as the authors report that most studies comprise only a single or a few assays or targets. Having a limited number of assays or targets in a dataset restricts the choice of the method employed to that particular subset of chemical space. In this study, target prediction was modelled as a binary classification problem, indicating whether a certain compound inhibits a pathway, binds to a receptor or induces toxic effects. The authors also demonstrate that the application of deep learning to activity against target prediction is comparable, and can outperform, *in vitro* assays. The authors observe that feed-forward neural networks outperform other methods, followed by SVMs. It is highlighted that these two methods outperform graph convolution networks in this particular study.

2.6.1 | Molecular Representation

Before the molecules in our dataset can be processed using the aforementioned machine learning models, we need to represent the molecules in computational space. A number of representations have been studied, including extended-connectivity fingerprints (ECFP) (Rogers and Hahn, 2010), Coulomb matrices (Rupp et al., 2012), graph convolutions (Duvenaud et al., 2015) and Weave featurisation (Kearnes et al., 2016). However, our study will focus mainly on ECFPs and graph representations of molecules.

Molecules are complex structures, consisting of a combination of different atoms, bonds, and conformations. One challenge in the field of small-molecule design is the representation of the chemical composition of molecules for computer processing. The classical notation of chemicals is the empirical formula, which is a form of the Hill notation. The formula of alanine, $C_3H_7NO_2$, provides information about the atoms present in the molecule. However, there is no information on how atoms are linked together and as a result, the same empirical formula can refer to sarcosine and lactamide (David et al., 2020).

The choice of molecular representations presupposes the choice of ML architecture to be employed. Graph representations of molecules must be processed using graph

neural networks (GNNs) due to the nature of the data. On the other hand, given that fingerprints are representations in vector forms, these can be processed using conventional feed-forward neural networks. While it is difficult to have one single descriptor that works best for all machine learning models, more studies are required as it is still not clear which descriptors work best for different small-molecular design problems (Vamathevan et al., 2019).

2.6.1.1 | SMILES Notation

Weininger (1988) initiated the SMILES notation, that represents the chemical structure in a simple, text-based syntax. In the datasets discussed in Section 2.3, molecules are supplied in the SMILES format. SMILES supports all the elements found in the periodic table. Upper case letters denote non-aromatic atoms, while lower case letters refer to aromatic ones. Bonds are single bonds by default and are not written in the SMILES notation. Double bonds, triple bonds, aromatic and disconnected bonds are represented by =, #, *, and . respectively. SMILES notation is hydrogen-suppressed, meaning that hydrogens (H) are excluded by default. However, they can also be explicitly defined. Branches in the molecule are defined in SMILES by placing the SMILES symbols between parenthesis. Ring structures are represented through the use of a number to pinpoint the opening and closing of ring atoms. In Cyclohexane (C1CCCCC1), the first carbon, followed by a 1, is connected by a single bond to the last carbon, which is also followed by a 1. Table 2.1 illustrates 3 examples of SMILES representations, along with their SMILES-based molecular construction using RDKit (refer to Section 2.6.2.1). Figure 2.25 illustrates the generation of SMILES notation in canonical order of Aspirin.

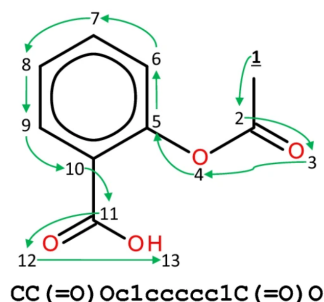


Figure 2.25: SMILES generation in canonical order of Aspirin. Reproduced from Arús-Pous et al. (2019).

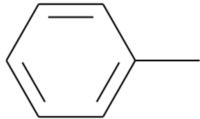
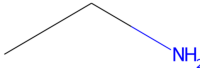
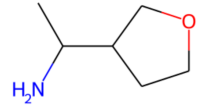
Molecule	SMILES	Structure Diagram
Toluene	<chem>CC1CCCCC1</chem>	
Ethylamine	<chem>NCC</chem>	
3-(1-aminoethyl)tetrahydrofuran	<chem>C1OCC(C(N)C)C1</chem>	

Table 2.1: Examples of SMILES representations. Structure diagrams are generated using RDKit (refer to Section 2.6.2.1).

2.6.1.2 | Extended Connectivity Fingerprints (ECFP)

Molecular fingerprints are fixed vectors which represent chemical molecules, historically utilised for substructure and similarity searching. ECFP are explicitly designed to capture the relationship between the molecular features and the molecular activity. The features captured by ECFPs describe both the presence (i.e., *positive* information), and also the absence (i.e., *negative* information) of structural information. They are widely used in LBVS, and can be used to distinguish between active and inactive molecules using similarity methods. The generation of the fixed-length ECFP string is a three-step process that can be executed efficiently. The generation of ECFP can be used to represent any novel molecules, as the process is not *a priori*.

1. **Initial assignment.** Each non-hydrogen atom is assigned a 32-bit integer identifier, capturing atomic information. ECFP uses Daylight atomic invariants rules, which capture six atom properties. These include (i) number of immediate non-hydrogen neighbouring atoms, (ii) the valence excluding hydrogens, (iii) atomic number, (iv) atomic mass, (v) atomic charge, and (vi) number of attached hydrogens. An additional property capturing whether the atom is aromatic is included.
2. **Iterative updating.** Each atom identifier is updated to reflect the identifier of the neighbours within the *diameter* set. The diameter is the number of bonds that are included in each iteration, starting from one. It is a crucial parameter in ECFP generation, and Rogers and Hahn (2010) suggest that diameters greater than three or four are ideal for capturing molecular activity. Each iteration captures larger

connects two atoms together. As the bonds in a molecule are not directed, molecular graph representations are undirected. Converting an abstract concept such as a graph to a computer readable representation involves the use of matrices and arrays. Atom connectivity in a molecule is represented through an adjacency matrix A (also referred to as a connectivity matrix). The adjacency matrix contains the edges joining v_i and v_j (Bondy and Murty, 1976). A consists of a_{ij} Boolean elements, where a one signifies a bond between two atoms, and a zero the absence of one. Atom features are represented in a node features matrix X . Each row x_i of X , is the node feature vector of that atom. Selected properties such as atomic number, atom type, charge, valences and other properties can be encoded in the node feature vector. These are typically one-hot encoded in machine learning applications. Bond information is represented in the edge feature matrix E , whose rows correspond to the edge feature vector e_{ij} , which maps edge information between two atoms $e_{ij} = (v_i, v_j)$. E consists of the possible bond types between two atoms, namely single, double, triple, or aromatic bonds (David et al., 2020). These three elements (the adjacency matrix, the node feature matrix, and the edge feature matrix) are illustrated in Figure 2.27

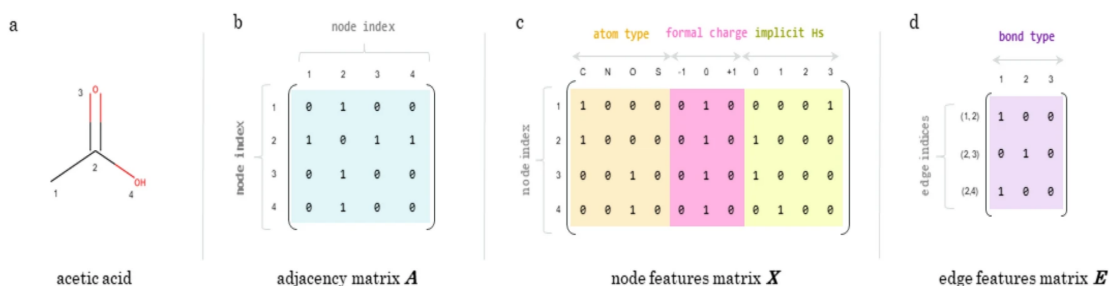


Figure 2.27: Graph representation of acetic acid. (a) Structural diagram. (b) Adjacency Matrix. (c) One-hot encoded atom features. (d) One-hot encoded edge feature matrix. Reproduced from David et al. (2020).

As graphs are 2D data structures, molecular information such as delocalised bonds, polycentric, ionic or metal-metal bonds cannot be represented well using pairwise relationships between atoms in adjacency matrices. In real scenarios, molecules are constantly changing in 3D space, and pairwise bonds can break and reform, essentially rearranging the molecule (David et al., 2020). In this study, we assume a single molecular graph representation without changing conformations. Wu et al. (2018) show that graph convolutions are the best performing models as graph-based models outperform conventional methods on 11/17 datasets.

2.6.1.4 | Graph Learned Embeddings

Graph neural networks can be used to learn molecular representations by applying convolutional operators and pooling layers on the molecular graphs (Jiang et al., 2021). Embeddings learned through neural networks afford the construction of automated features, rather than fixed fingerprints. Duvenaud et al. (2015) propose a method to compute neural graph embeddings, which they report is an improvement over circular fingerprints. A single layer neural network (refer to Section 2.4.1) is used to compute the molecule vector by taking as input the molecule represented as a graph. Graph neural networks are effective in transforming small molecules into real-valued vector representations, which has been found to be a productive way of processing small molecules within deep neural networks (Gómez-Bombarelli et al., 2018). By using a differentiable method, Duvenaud et al. (2015) report that collisions of substructures are reduced and the fingerprint can be optimised to contain only relevant features. Activity and similarity of substructures is also captured, increasing the fingerprint's interpretability. The combination of atom and neighbourhood atoms through hashing in ECFP is replaced by a layer of a neural network in neural fingerprint generation. Instead of encoding the output vector through indexing, a *softmax* is used, essentially classifying each atom into a category. The final fingerprint is produced from the sum of these classification label vectors, which is analogous to the pooling operation in convolutional networks (refer to Section 2.4.2).

When tested against ECFP, Duvenaud et al. (2015) report that learned embeddings matched or outperformed ECFP in all experiments, which included solubility, drug efficacy and organic photo-voltaic efficiency. However, this comes at a computational cost as training is computationally more expensive than ECFP generation. Which approach is deemed superior is still an unanswered question as the literature is conflicting. Wu et al. (2018) demonstrate that convolution based models outperform fingerprint based models, while Mayr et al. (2018) report otherwise. Wu et al. (2018) report that graph-based models outperformed conventional machine learning models on most datasets, except for those which have scarce data, or are heavily imbalanced. Under high imbalance, graph-based models are reported to not be robust in controlling false positives. This improvement in graph-based methods across the majority of datasets suggests that a learned embedding is advantageous over other molecular representations. In our study, we explore this further by comparing the performance of our best performing few-shot learning architecture with both ECFP inputs and graph objects.

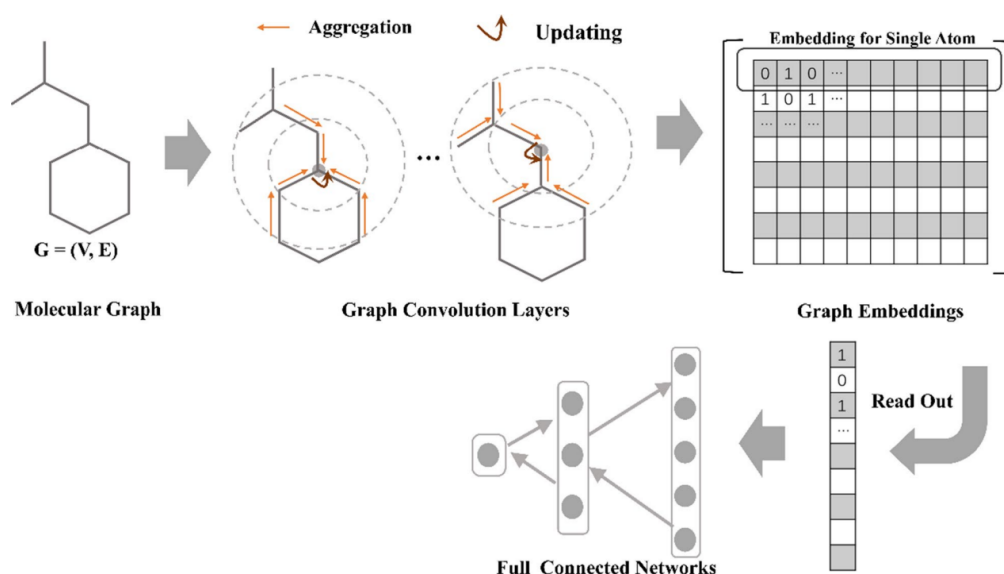


Figure 2.28: A typical pipeline for representing molecules using a learned embedding function, which can be processed further using feed-forward neural networks as shown. Reproduced from Jiang et al. (2021).

2.6.2 | Open-source Libraries

We make use of a number of open-source libraries in our study, however, the following libraries are specifically developed for cheminformatics. The following sections provide a brief overview of these libraries.

2.6.2.1 | RDKit

RDKit (RDKit, 2012) is an open-source cheminformatics software that can be used to generate descriptors and fingerprints for machine learning. Molecular visualisations can also be generated using this library. This library is frequently used in cheminformatics libraries. We mainly make use of RDKit to add atom descriptors to the graph objects representing the molecules. In our molecular representation creation pipeline, we make use of two other libraries, namely DGL-LifeSci (Mufei et al., 2021) and the ChEMBL Structure Pipeline (Bento et al., 2020). The former library is used to create the atom descriptors, while the latter is used to standardise the input molecule. The functions used from these libraries are based on RDKit, but are conveniently provided through less complex APIs. We delve into more detail for our implementation in Chapter 3.

2.6.2.2 | DeepChem

Ramsundar et al. (2019) initiated the DeepChem project, an open-source machine learning library to democratise deep learning for drug discovery, material science, quantum chemistry, and biology. DeepChem allows users to integrate implementations with popular ML libraries such as Keras and Pytorch. For our study, the DeepChem library is used to create ECFP molecular representations.

MoleculeNet

The datasets discussed in Section 2.3 are all publicly available. However, MoleculeNet (Wu et al., 2018) aggregated multiple public databases to provide a full collection of over 700,000 compounds. MoleculeNet is based on the DeepChem library, which was introduced in the previous section. Figure 2.29 illustrates the different categories of data found in MoleculeNet.

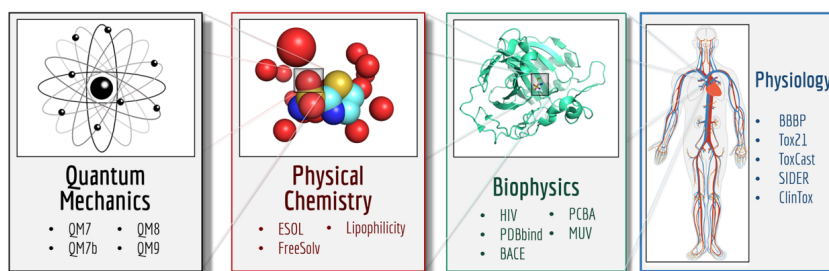


Figure 2.29: Different categories of datasets in MoleculeNet. Reproduced from Wu et al. (2018).

In this study, we will focus on Biophysical and Physiological classification-based datasets. While the Tox21 and MUV datasets are conveniently available through MoleculeNet, the DUD-E dataset is not. However, the addition of the DUD-E dataset to MoleculeNet is a work in progress by the authors of this study. The data contained in MoleculeNet for these three datasets is highlighted in the following list.

- **Tox21.** Data on 8,014 compounds against 12 different NR and SR pathways is available.
- **MUV.** Data on circa 90,000 compounds for 17 different tasks.

MoleculeNet also includes data splitting methods namely random, scaffold, stratified, and time-based splitting. Random splitting separates data randomly across training, validation and test groups. Scaffolding splits the data based on the RDKit based 2D

structural representation of the molecule. Stratified splitting sorts data points in order of their label value in the case of continuously labelled datasets, such that each subgroup contains the whole range of labels. Finally, time-based splitting divides data based on the time, so training is performed on older data and tested on newer data.

MoleculeNet also provides featurisation functions to convert SMILES representations into better suited representations for machine learning. These include ECFP, Coulomb Matrices, Grid Featurisation, Symmetry Functions, Graph Convolutions and Weave featurisation. For the purposes of this study, only ECFP and Graph Convolutions will be considered.

Finally, MoleculeNet tests the performance of machine learning models including logistic regression, support vector classification, kernel ridge regression, random forests, gradient boosting, multitask networks, bypass networks, influence relevance voting, graph convolutional models, weave models, directed acyclic graph models, deep tensor neural networks, ANI-1 and message passing neural networks. These methods are implemented as part of the DeepChem library.

2.6.2.3 | Deep Graph Library

Wang et al. (2019) propose a platform agnostic library called Deep Graph Library (DGL), which easily integrates with tensor oriented frameworks such as PyTorch (Paszke et al., 2019) to allow deep learning on graphs. DGL provides flexible and efficient APIs to allow arbitrary message passing computations over graphs. This library conveniently provides ready-made convolutional operations for GNNs implementations such as the aforementioned Graph Convolutional Network (GCN), pooling layers, and readout functions. This library is used for our graph-learned molecular embeddings. We load the data, and process it using APIs provided from this library, but this is further explained in Chapter 3.

2.7 | Related Work

While the aforementioned few-shot learning is primarily studied for the computer vision domain to recognise images in previously unseen classes using only a small support set, in drug discovery, we aim to predict the activity of a molecule in a new experimental assay. In the case of drug discovery, we have limited information on the new molecule assay, but we can make use of similar molecule assays (prior knowledge) (Wang et al., 2020).

There have not been many studies around few-shot learning for low-data problems in drug discovery. However, Altae-Tran et al. (2017) proposed a state of the art approach for few-shot learning in this problem domain. The authors apply Siamese Networks (Koch et al., 2015) for molecular datasets and further build on the concepts of Matching Networks (Vinyals et al., 2016) to propose the Iterative-Refinement LSTM (IterRefLSTM) networks.

Altae-Tran et al. (2017) build on meta-learning concepts, where they train a machine learning model on molecular data from a set of targets reserved for training. The model is then used to generalise for the activity of molecules in new, previously unseen experimental assays using only a small support set from the new assay. These test assays are related, but not identical, to the ones reserved for training. The support set is a small set of molecules sampled from an experimental assay to train a machine learning model. The number of molecules sampled for each class in the support set ranges from one, to a maximum of ten molecules. In their work, the support and query molecules are embedded using a graph convolutional network as shown in Figure 2.30. Bond information and distinction between bond types was not considered in this study. The GCN architecture used by the authors is tabulated in Table 2.2. We note that the *pool* layers tabulated do not coarsen the graphs as explained in Section 2.4.4.2, but simply apply a max function over neighbouring nodes. The *gather* layer is equivalent to a global pooling or readout layer discussed in Section 2.4.4.5.

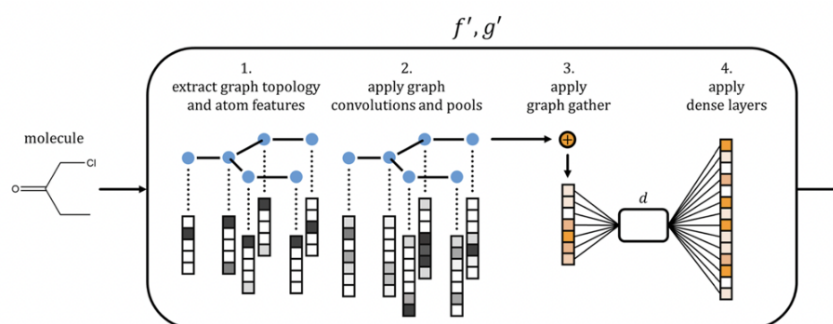


Figure 2.30: Embedding functions f' and g' , used to embed molecular graphs from the support and query sets in latent space. Reproduced from Altae-Tran et al. (2017).

To further process the resulting embeddings from the GCN, Altae-Tran et al. (2017) utilise a bidirectional LSTM and an attention mechanism to eliminate the order dependence of the LSTM. This model is referred to as the attentional LSTM (attnLSTM) in the original work and utilises the support set to enhance the query embeddings. How-

layer	conv	pool	conv	pool	conv	pool	dense	gather
dimension	64		128		64		128	
nonlinearity	relu		relu		relu		tanh	tanh

Table 2.2: Graph Convolutional Network Architecture in Altae-Tran et al. (2017).

ever, the model that provided the best results is the proposed iterative refinement LSTM model (IterRefLSTM). In this implementation, the two embedding functions $f(\cdot|S)$ and $g(\cdot|S)$ are developed simultaneously. Therefore, the embedding of the query is built iteratively with that of the support set, using information from both sets to enhance the support and query embeddings. The iterative refinement process is visualised in Figure 2.31.

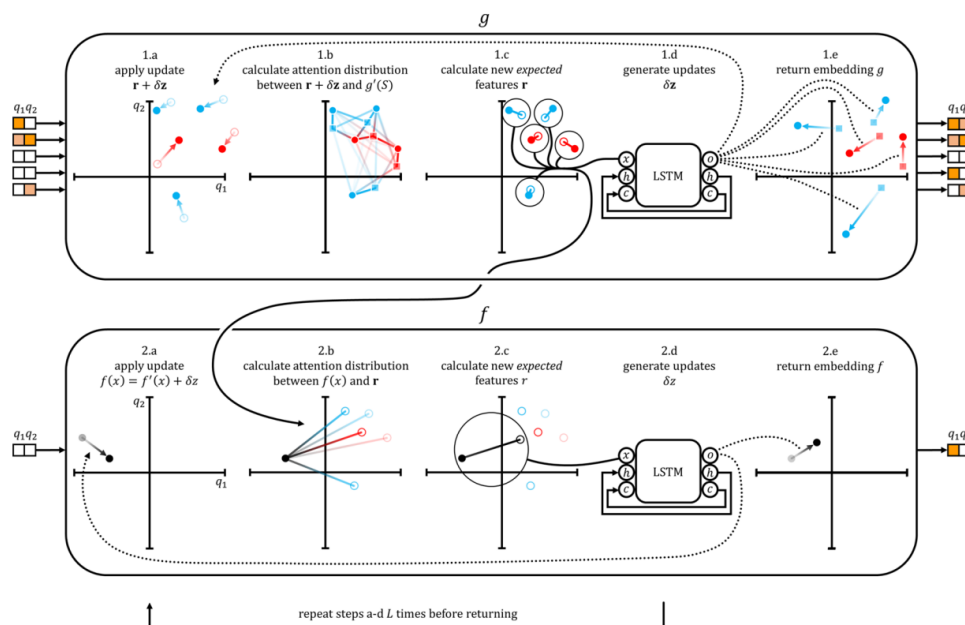


Figure 2.31: Iterative refinement of embeddings using an LSTM network. The red and blue points depict the active and inactive/decoy class respectively. The squares depict the original embedding from g' . Reproduced from Altae-Tran et al. (2017).

Once the embeddings have been iteratively refined, the authors apply a metric-based function to classify the queries using the support set embeddings. To emulate the Matching Networks, the authors make use of the cosine distance to achieve this.

Figure 2.32 illustrates a one-shot learning scenario encapsulating the aforementioned concepts.

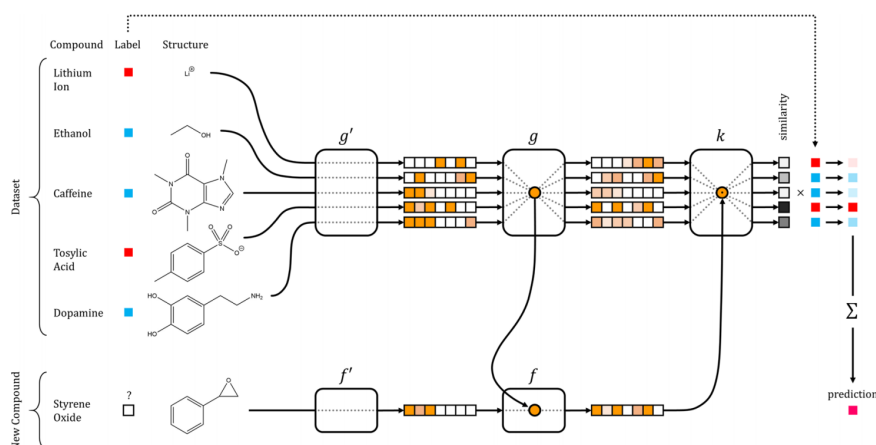


Figure 2.32: Schematic of one-shot learning in drug discovery based on the Matching Network (Vinyals et al., 2016) architecture. Reproduced from Altae-Tran et al. (2017).

Their work is tested on the Tox21, the Side Effect Resource (SIDER) (Kuhn et al., 2016), and MUV datasets. We provide an overview of the Tox2 and MUV datasets in Section 2.3 as we make use of these datasets in our study. The SIDER dataset contains information on side effects for marketed drugs, and is primarily used in the lead optimisation part of the drug-discovery pipeline (refer to Section 2.1). For every dataset, a subset of the targets is reserved for training and the rest for testing. Training is carried out as explained in the Matching Networks paper, in which training conditions match those at test time (Vinyals et al., 2016). The authors make use of a random forest with 100 decision trees as a machine learning baseline model. They also utilise a conventional GCN as an additional baseline model, which is trained using only a small support set from the test targets. They then experiment with Siamese Networks (Koch et al., 2015), Matching Networks (Vinyals et al., 2016), and an adaptation of the Matching Networks by applying the iterative refinement concepts explained above.

The authors report ROC-AUC scores to report the performance of the models. Considering the extreme imbalance of the data in the utilised datasets, we note that the PR-AUC score would be more appropriate. Section 2.4.5 explains the difference between these two metrics in further detail, however, PR-AUC is based on the relationship between precision and recall. Therefore, this evaluation metric provides a clearer picture into how the model performs when predicting the *positive* (active) class in the data. Predicting the active class correctly is of significant importance in virtual screening.

On the Tox21 and SIDER datasets, the proposed machine learning architecture achieves good ROC-AUC performance. The mean score for 10-shot learning on the median held-out task on Tox21 achieves a score of 0.823 ± 0.002 , while for one-shot learning the model achieves a mean score of 0.827 ± 0.001 . The reasons why one-shot learning achieved better performance than 10-shot learning is uncertain, as we expect the model to perform better with larger support sets. However, this might be attributed to variance in the data between experiments. On MUV data, the baseline machine learning models outperformed few-shot learning. The authors report that this is due to MUV data being maximally informative, and therefore structural similarity cannot be utilised to generalise for activity prediction.

The authors open-sourced the models developed in the DeepChem library. However, the implementations are now outdated and not executable with the DeepChem library, which makes reproduction of results difficult. However, we study the open-sourced implementation along with the implementation details in the original literature to successfully reproduce this work. In Chapter 3, we provide further detail into how we reproduced this work in order to reproduce results and develop the work further. This study builds on the work of Altae-Tran et al. (2017) to further explore few-shot learning for virtual screening.

2.8 | Summary

In this chapter, we provided a comprehensive overview of the relevant topics and a theoretical foundation for the machine learning concepts required for this study. We first introduced the various components of the drug discovery process, followed by an overview of the three datasets to be used for this study. We introduce the relevant machine learning concepts, and we build upon them by reviewing the literature on few-shot machine learning. The chapter then moves to how molecules can be represented in computational space, after which we introduce the relevant work in the field of few-shot learning for low-data drug discovery.

Methodology

In this chapter, we elaborate on the methodology used to achieve the established aims and objectives. We first introduce a schematic of the different components of our pipeline, expanding upon each one to allow reproduction and further development of our work. Where needed, we will present concise details on theoretical concepts, and link back to sections in the *Background and Literature Overview* chapter to aid the reader. Section 3.8 provides an overview of the hardware specifications and library versions used throughout this project.

3.1 | Overview

The machine learning pipeline for this study consists of seven main parts, which are illustrated in Figure 3.1. We will elaborate on each part in subsequent sections, however, the following list provides an overview of the whole process. For convenience and efficiency, the output of the first four steps is loaded into a Pandas DataFrame and saved to a Pickle file to avoid repeating the generation of molecular features and graphs before every experiment. These remain unchanged throughout all experiments.

1. **Data Acquisition.** We utilise three main publicly available datasets for this study, namely, the Tox21, MUV and the GPCR subset of the DUD-E. The data is provided as SMILES strings with a Boolean value for the experimental assays in the dataset recording whether the molecule is active or an inactive/decoy.
2. **Standardise molecule.** The SMILES strings are first standardised in order to transform all molecular representations according to a set of well-defined and consistent rules and conventions to ensure validity and uniformity.

3. **Molecular features generation.** The molecular graph generated from the standardised SMILES representation is enriched with atom descriptors to add information to the molecular representation.
4. **Molecular graphs generation.** The molecular representations we have so far are transformed into graph objects, consisting of nodes and edges representing atoms and bonds respectively. The connectivity between atoms is represented via an adjacency matrix.
5. **Episode generation.** Effective few-shot learning necessitates that conditions at training match those at testing (Vinyals et al., 2016). Therefore, support sets and queries are randomly sampled to form a series of episodes for training. The support sets are composed of a number of examples per class. The number of examples per class ranges from just one example, up to 10 examples.
6. **Learning a molecular embedding.** The sampled molecules in the episode are used to learn a molecular embedding using a graph neural network.
7. **Few-Shot Learning.** The learned embeddings are processed using four different meta-learning architectures namely Siamese, Matching, Prototypical and Relation Networks. Iterative Refinement LSTMs (Altae-Tran et al., 2017) are used to enrich the few-shot learning. A subset of experimental assays in each dataset is reserved for training, while the rest are reserved for testing.
8. **Testing.** The trained models are subsequently used to test on new experimental assays, unseen during training, to gauge the generalising capability of a model trained for a low-data scenario. Support sets are randomly sampled and trained on the remaining molecules in the dataset for 20 rounds, for which the mean and standard deviation of the areas under the curve for PRC and ROC curves are calculated to quantify performance.
9. **Evaluation.** Finally, we evaluate the results based on the Receiver Operator Characteristic (ROC) and Precision Recall Curves (PRC) scores from the 20 test rounds. We apply statistical analysis for results obtained across different experiments to determine the best performing techniques for each support set composition. Confusion matrices, ROC and PRC graphs for the experiment with the median ROC score from the 20 rounds are generated after test completion.

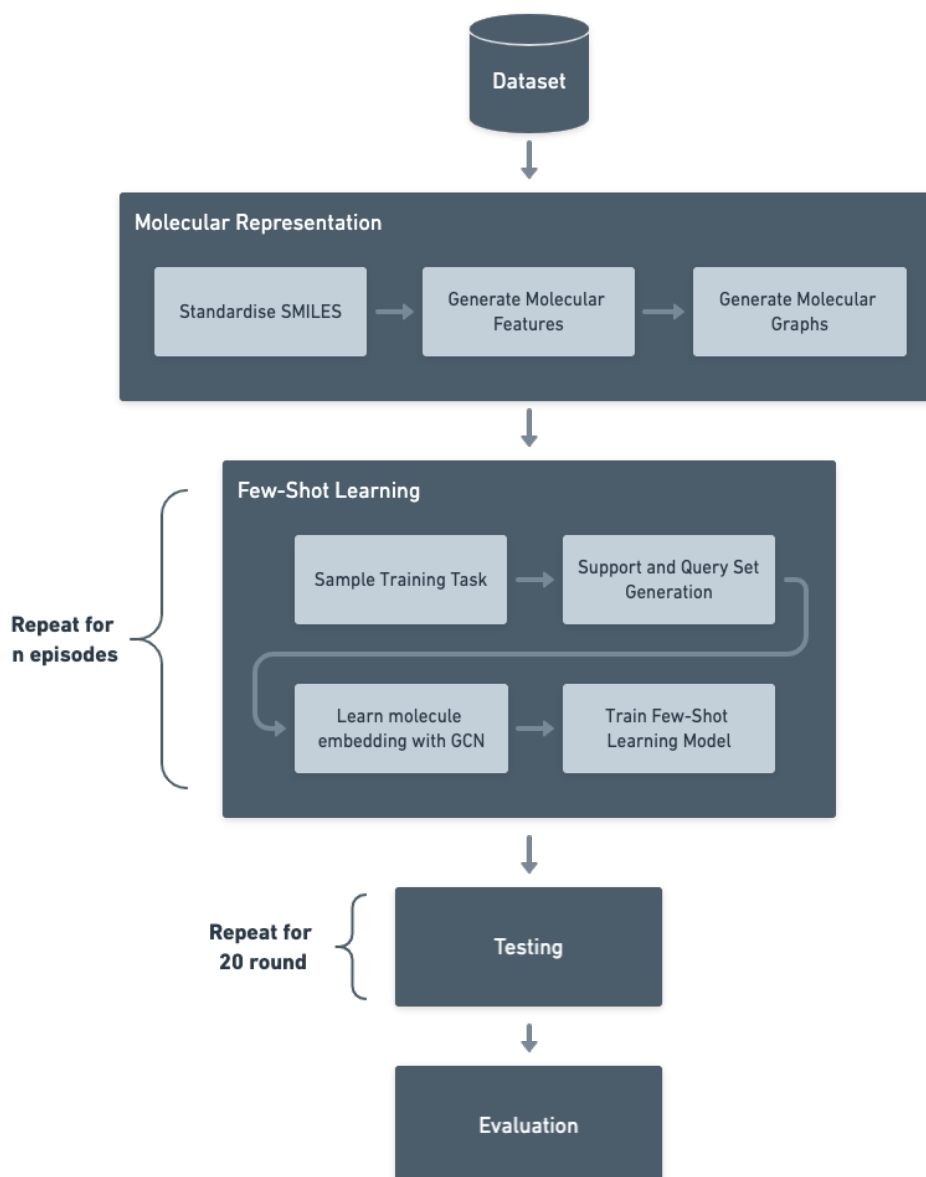


Figure 3.1: Schematic of the machine learning pipeline designed for this study. The changes across few-shot learning architectures lies in the 'Train Few-Shot Learning Model' component, otherwise, all other modules remain identical.

3.2 | Data Acquisition

For this study, we have used three main public datasets, namely, Tox21, the Maximum Unbiased Validation (MUV), and the Database of Useful (Docking) Decoys - Enhanced (DUD-E) dataset. The data sources are listed in the following list.

- **Tox21.** The dataset was obtained from the DeepChem AWS bucket¹ in CSV format.
- **Maximum Unbiased Validation (MUV).** The dataset was obtained from the DeepChem AWS bucket² in CSV format.
- **Database of Useful (Docking) Decoys — Enhanced (DUD-E).** The data for the GPCR subset was obtained directly from the DUD-E website.³ The actives and decoys for the targets within the DUD-E subsets are provided as separate SMILES files. These files are loaded using the Pandas library and aggregated in a CSV file per subset. The script for performing this operation is provided in the GitHub repository for this project (see Appendix A).

The targets used, along with the number of active and inactive/decoy molecules and whether these were used for training or testing can be found in Table 3.1 and Table 3.2 for the Tox21 and MUV datasets respectively. For the DUD-E dataset, target classes for two particular subsets, grouped by the data provider are used. These include seven transmembrane helix receptors (GPCR), and nuclear hormone receptors (Nuclear). The composition of the data for both these subsets can be found in Table 3.3. For the purposes of this study, the rest of the targets in the DUD-E dataset were omitted. It is important to highlight the inherent imbalance of the datasets used, in which the ratio of actives to inactives is very low, having tens or hundreds of actives in contrast to the thousands available inactives. Within Tables 3.1 - 3.3, the split column shows which targets are reserved for training and which are used for testing. The goal in this study is to train a model on experimental assays by emulating the low-data conditions during test time, and then using a small support set from unseen experimental assays to generalise for the remaining data for the unseen experimental assay.

The format of the CSV files is identical across the three different datasets used. The `smiles` column contains the molecules in SMILES notation. The rest of the columns are

¹ Accessed from: <https://deepchemdata.s3-us-west-1.amazonaws.com/datasets/tox21.csv.gz>. Last Accessed: 08 Nov 2021

² Accessed from: <https://deepchemdata.s3-us-west-1.amazonaws.com/datasets/muv.csv.gz>. Last Accessed: 08 Nov 2021

³ Accessed from: <http://dude.docking.org/subsets>. Last Accessed: 08 Nov 2021

Target	Split	Inactives	Actives
NR-AR	Training	6,956	309
NR-AR-LBD	Training	6,521	237
NR-AhR	Training	5,781	768
NR-Aromatase	Training	5,521	300
NR-ER	Training	5,400	793
NR-ER-LBD	Training	6,605	350
NR-PPAR-gamma	Training	6,264	186
SR-ARE	Training	4,890	942
SR-ATAD5	Training	6,808	264
SR-HSE	Testing	6,095	372
SR-MMP	Testing	4,892	918
SR-p53	Testing	6,351	423
<i>Total Training</i>		54,746	4,149
<i>Total Testing</i>		17,338	1,713

Table 3.1: Tox21 Dataset Composition

named as the targets within the specific dataset or subset, in the case of the DUD-E. The target columns contain boolean values $\{0, 1\}$, denoting whether the molecule is an active or inactive/decoy against the target. Molecules for which no data exists about its activity against a specific target are left blank. The Tox21 and MUV dataset also contain a `mol_id` column, however, this can be omitted for the purpose of this study.

3.3 | Generating the Molecular Representation

The datasets used provide molecules as a string in SMILES notation. This string alone provides limited information, but it can be converted to a molecular graph, which in turn can be augmented with descriptors for its atoms. RDKit is an open source toolkit for cheminformatics, and will be directly used to generate the molecule object from the SMILES string. The molecule object is created using the `MolFromSmiles()` function from the RDKit Chem module. A visualisation of an RDKit molecule can be seen in Figure 3.3.

Target	Split	Decoys	Active	Target	Split	Decoys	Actives
MUV-466	Training	14,814	27	MUV-548	Training	14,705	29
MUV-600	Training	14,698	30	MUV-644	Training	14,593	30
MUV-652	Training	14,873	29	MUV-689	Training	14,572	29
MUV-692	Training	14,614	30	MUV-712	Training	14,383	28
MUV-713	Training	14,807	29	MUV-733	Training	14,654	28
MUV-737	Training	14,662	29	MUV-810	Training	14,615	29
MUV-832	Testing	14,637	30	MUV-846	Testing	14,681	30
MUV-852	Testing	14,622	29	MUV-858	Testing	14,745	29
MUV-859	Testing	14,722	24				
<i>Total Training</i>		175,990	347	<i>Total Testing</i>		73,407	142

Table 3.2: MUV Composition from data provisioned by DeepChem.

Target	Split	Decoys	Actives
aa2ar	Training	31,550	482
drd3	Training	34,050	480
adrb1	Training	15,850	247
adrb2	Testing	15,000	231
cxcr4	Testing	3,406	40
<i>Total Training GPCR Subset</i>		99,856	347
<i>Total Testing GPCR Subset</i>		3,406	142

Table 3.3: Composition of the GPCR subset from the DUD-E dataset used for this study.

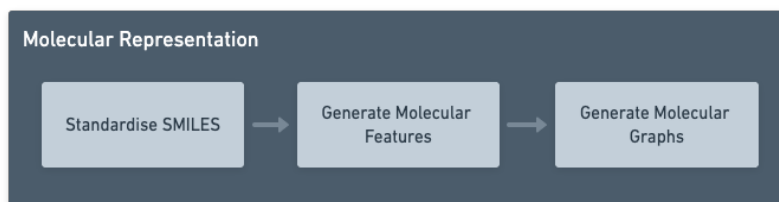


Figure 3.2: Schematic of the steps in the molecular representation module.

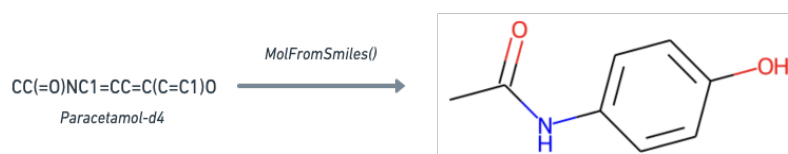


Figure 3.3: Illustration of the conversion from an example SMILES string to an RDKit Molecule, created using the `MolFromSmiles()` function and visualised inside a Colab Notebook using the RDKit IPythonConsole from the `Chem.Draw` module.

3.3.1 | Standardise SMILES Molecules

Before generating any features or molecular representation, the molecule from the SMILES string must first be standardised. Standardisation of compounds according to a set of well-defined and consistent rules and conventions is of utmost importance to maintain uniformity and integrity across the data being used. Bento et al. (2020) propose an open source chemical structure curation pipeline based on RDKit for validating and standardising chemical structures, which follow FDA/IUPAC guidelines (Brecher, 2006; Food and Administration, 2007). The authors report that the mere loading of data can also result in subtle changes in structural information, highlighting the importance of standardisation in a cheminformatics pipeline. Their work is packaged in the ChEMBL Structure Pipeline package⁴, which provides a *Standardizer* component for standardising molecules. The following list highlight the processes involved in the standardizer with the utilised version (see Section 3.8 for full list of package versions).

1. Exclude organometallic and molecules with more than seven boron atoms from the standardisation process.
2. Standardise unfamiliar stereochemistry.
3. Assign double bonds to the molecular graph using the delocalisation subgraph as a guide. This process is referred to as kekulization.
4. Remove explicit hydrogen atoms except for instances where (i) an isotope is set, (ii) the hydrogens have a dashed or wedged bond, (iii) they are chiral hydrogens, (iv) they are bonded to atoms with three or more ring bonds, and (v) they are bonded to atoms which have a charge of +1 and a valence one higher than the default.

⁴Accessed from: https://github.com/chembl/ChEMBL_Structure_Pipeline. Last Accessed: 09 Nov 2021

Object	Type	Format	Size
Graph	Node	Float32 Tensor	177

Table 3.4: Feature vector size for molecular graphs

5. Normalise the structure

- a) Fixing of hypervalent nitro groups
- b) Ensure halogens with no neighbours, trivalent (valence of three) Oxygens and trivalent Sulfurs are charged.
- c) Diazonium Nitrogens are standardised to N+
- d) Sulphoxides are standardised to charge separated form.
- e) Correct amides
- f) Standardise KO to K+ O- and NaO to Na+ O-
- g) Ensure quaternary Nitrogens are charged.

6. Neutralise the molecule by adding or removing hydrogen atoms to neutralise the formal charge of the atom.

3.3.2 | Generate Molecular Features

So far, we have access to a molecule object with atoms and bonds, but we can enrich this information by adding various descriptors of the atoms and bonds within each molecule. Most of our experiments are run using only atom features, while omitting bond information and regarding all bonds as identical bonds, as was carried out in the state of the art by Altae-Tran et al. (2017).

All atom descriptors are obtained and one-hot encoded using the molecule featurisation utility functions in the DGL LifeSci⁵ library, which uses RDKit under the hood to obtain atom and bond descriptors. We loop over each atom in a molecule and featurise each atom accordingly. The various descriptors are concatenated into a single vector of size 177 bits using the DGL LifeSci ConcatFeaturizer as shown in Table 3.4. Each atom's concatenated feature vector is added to a PyTorch Tensor.

We consider the following descriptors for every atom. Each atom descriptor in the list that follows is accompanied by the DGL LifeSci function used to generate the feature

⁵Accessed from: <https://lifesci.dgl.ai/>. Last Accessed: 09 Nov 2021

vector in snake case. These functions are found in the `dgllife.utils` module. To avoid redundant statements in the list, all returned values are one-hot encoded.

- **Atom Type** - `atom_type_one_hot`. Returns the type of atom from the default set of atoms which include {C, N, O, S, F, Si, P, Cl, Br, Mg, Na, Ca, Fe, As, Al, I, B, V, K, Tl, Yb, Sb, Sn, Ag, Pd, Co, Se, Ti, Zn, H, Li, Ge, Cu, Au, Ni, Cd, In, Mn, Zr, Cr, Pt, Hg, Pb}.
- **Atomic Number** - `atomic_number_one_hot`. Returns the number of protons found in the nucleus of the atom.
- **Atom Degree** - `atom_degree_one_hot`. Returns the number of directly bonded neighbours to the current atom.
- **Explicit Valence** - `atom_explicit_valence_one_hot`. Returns the explicit valence of the atom. The valence relates to electrons involved in forming bonds. For example, an oxygen atom has a valence of two, while hydrogen has a valence of one. Two hydrogen atoms can bond with oxygen to form water (H_2O) to create a compound where all atoms have a full outer electron shell.
- **Hybridisation** - `atom_hybridization_one_hot`. Returns the atom hybridisation from a default set which includes SP, SP2, SP3, SP3D and SP3D2.
- **Formal Charge** - `atom_formal_charge_one_hot`. Returns the formal charge of the atom.
- **Radical Electrons** - `atom_num_radical_electrons_one_hot`. Returns the number of radical (i.e. unpaired) electrons in the atom.
- **Aromatic** - `atom_is_aromatic_one_hot`. Returns a Boolean value showing if the atom is aromatic.

3.3.3 | Molecular Graph Generation

Following the findings in Wu et al. (2018) and Altae-Tran et al. (2017), we choose to mainly focus on graph representations of molecules for our few-shot learning architectures. In order to learn a molecular embedding from molecular graphs, we need to generate a graph object with nodes (representing atoms) and edges (representing bonds). The graph entities are created using Deep Graph Library's (DGL) `DGLGraph` data structure. As we utilise the PyTorch backend for DGL, node and edge features are stored as PyTorch tensors, in a `ndata` interface for node data. While nodes in a molecular graph

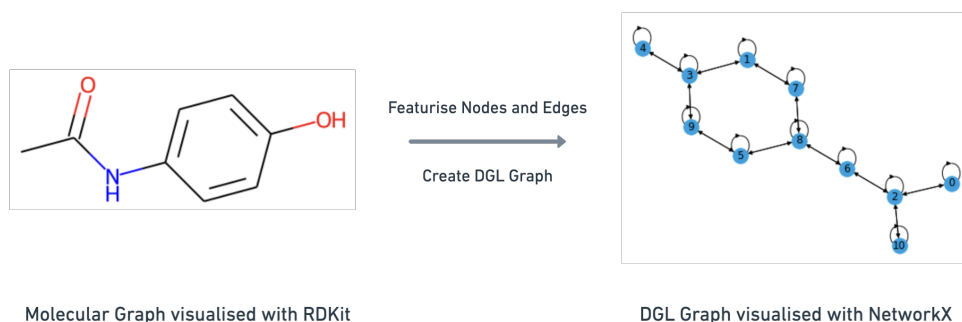


Figure 3.4: DGL Graph visualised with NetworkX using the Kamada Kawai Layout. The graph shows how bonds are bidirectional and contain self loops to aggregate information to the node itself when performing message-passing. Each node contains a feature vector with various atom descriptors.

can contain distinct data, they are of the same entity type, making them homogeneous graphs. In the molecular graphs we create, the edges have no distinct direction as graph information can flow in both directions. Unless explicitly defined, hydrogen atoms are not added as nodes in the graph. The DGL Graph data structure is created using the DGL LifeSci library using the `mol_to_bigraph()` function. This function creates a bi-directed DGL Graph from an RDKit molecule object. The featurisers are passed into this function to featurise the nodes and edges where applicable. Self loops are added to every node in the generated graph so aggregation functions during message passing consider the features of the node itself. The order of the atoms follows the canonical order of the atoms assigned through RDKit. As the structure and number of atoms varies between different molecules, the graph objects created come in different sizes. This non-uniformity needs to be taken into consideration as the resulting adjacency matrices and the atom features vary in size. Therefore, batching graphs is not a trivial task, but it can be addressed using block-diagonal matrices, which is explained further in Section 3.4.2.

3.4 | Few-Shot Machine Learning

In this study, we utilise metric-based few-shot machine learning to address the low-data problem for drug discovery. To achieve this, the low-data conditions during test time are replicated repetitively during training through a series of episodes. The few-shot learning model is trained by sampling a small support set (one to ten per class) and predicting the probability of query examples as a function over the support set samples. Figure 3.5 illustrates the main steps involved in the few-shot learning pipeline, which

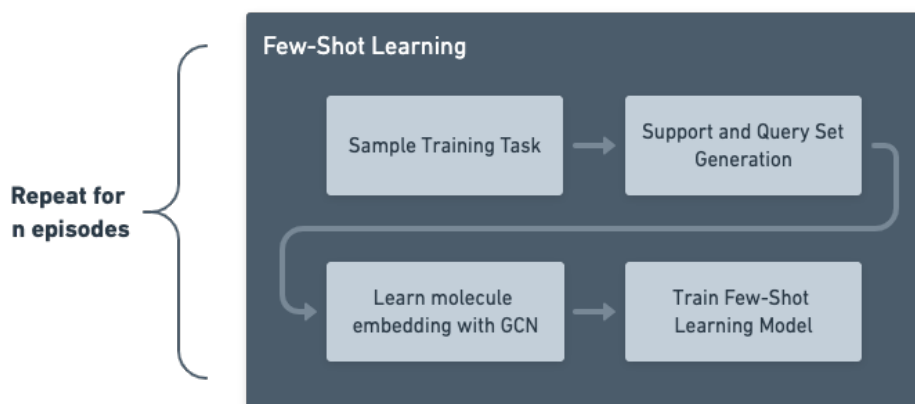


Figure 3.5: Schematic for episodic learning for our few-shot machine learning pipeline.

will be discussed in further detail in the following sections.

3.4.1 | Episodic Learning

Episodic learning is used to train a few-shot machine learning model. Vinyals et al. (2016) suggest that conditions during training must match those during testing. The conditions refer to the composition of the support set. Training consists of a sequence of learning problems where the model is supplied with a *support* set and a corresponding *query* set. The support set consists of a few molecules sampled from each class, in our case representing the active molecules and the inactives/decoys. The aim during training is to train a model how to classify the molecules within the query set using only the few data points in the support set. We consider N -way K -shot classification tasks, where the support set contains N classes and K labelled molecules. In our scenario, N is always assigned a value of two as we are attempting to solve a binary classification problem, whereby the model tries to classify the query molecules as active or inactive in a specific experimental assay. We experimented with a varying number of molecules for the support sets, however the minimum limit was set to one compound per class, while the maximum was set to 10 compounds per class. The 2 -way N -shot formulation is what the model is presented with at test time. As an example, in which $N = 10$, the model is presented with 10 active and 10 inactive compounds as the support set for an *unseen* target in an experimental assay. Using this support set, the model classifies the presented queries.

The value for K is not always the same for both classes as we also experiment with

Actives	Inactives/Decoys	Support Set Size
10	10	20
5	10	15
1	10	11
1	5	6
1	1	2

Table 3.5: Support set composition

using a different number of molecules sampled from each class to make up the support set. Table 3.5 contains the composition of the support sets used for our experiments. For each episode, we sample a total of 128 query molecules, which is composed of a balanced combination of molecules from each class. If the active class for a specific target contains less than 64 molecules, the active molecules are over-sampled such that each query set contains 64 actives.

Episodes are generated using the developed `create_episode` module, which is included in the Jupyter notebooks (see Appendix A). First, a task from the list of reserved tasks for training is sampled at random. The support and query sets for this task are sampled as explained previously, and training initiates as visualised in Figure 3.5.

Parameters for `create_episode` Module

- `n_support_pos` - [Integer] - Specifies the number of active labelled compounds to include in the support set.
- `n_support_neg` - [Integer] - Specifies the number of inactive/decoy labelled compounds to include in the support set.
- `n_query` - [Integer] - Specifies the number of queries per class to include in each episode during training.
- `data` - [Pandas DataFrame] - The dataset including the labels and the generated molecule representations
- `test` - [Boolean] - Specifies whether the episode is for training or testing. If the value is `True`, the module creates a support set, and includes all remaining molecules as queries.

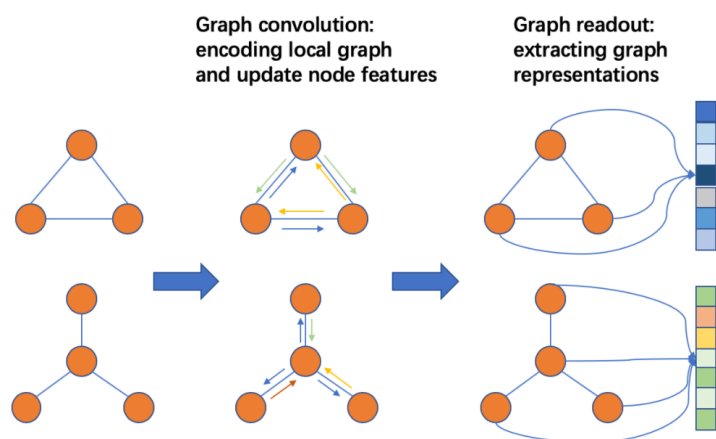


Figure 3.6: Schematic for the main components in learning an embedding from graphs using a graph neural network. Reproduced Figure.⁶

- `train_balanced` - [Boolean] - If set to True, the queries returned will be balanced, containing the same amount of actives and inactives. However, if this is set to False, a minimum of one compound per class will be generated, and the rest sampled at random. In this case, the queries will contain a higher ratio of inactives/decoys due to the significantly imbalanced nature of the data.

3.4.2 | Learning a Molecular Embedding

A Graph Neural Network (GNN) is used to learn to embed the support and query molecular graphs into latent space. The GNN is developed using the DGL libraries, which provides modules based on PyTorch. The architecture is presented in Table 3.6. Figure 3.6 shows an overview of the main components of a GNN.

1. **Batch Graphs.** As the end-goal in our few-shot learning models is to predict the activity of a particular molecule in an experimental assay, training needs to be done on graphs representing the molecules used for training. Processing individual graphs is inefficient, so the graphs used in each episode can be batched together to form a single batched graph as illustrated in Figure 3.7 using the DGL `GraphDataLoader` function to batch graphs. This data loader inherits from the PyTorch dataloader. We set the `pin_memory` flag to True, which enables faster data transfer to CUDA enabled GPUs. The structure and number of atoms varies between molecules. This non-uniformity therefore results in varying inputs for the

⁶Accessed from: <https://docs.dgl.ai/guide/training-graph.html>. Last Accessed: 09 Nov, 2021

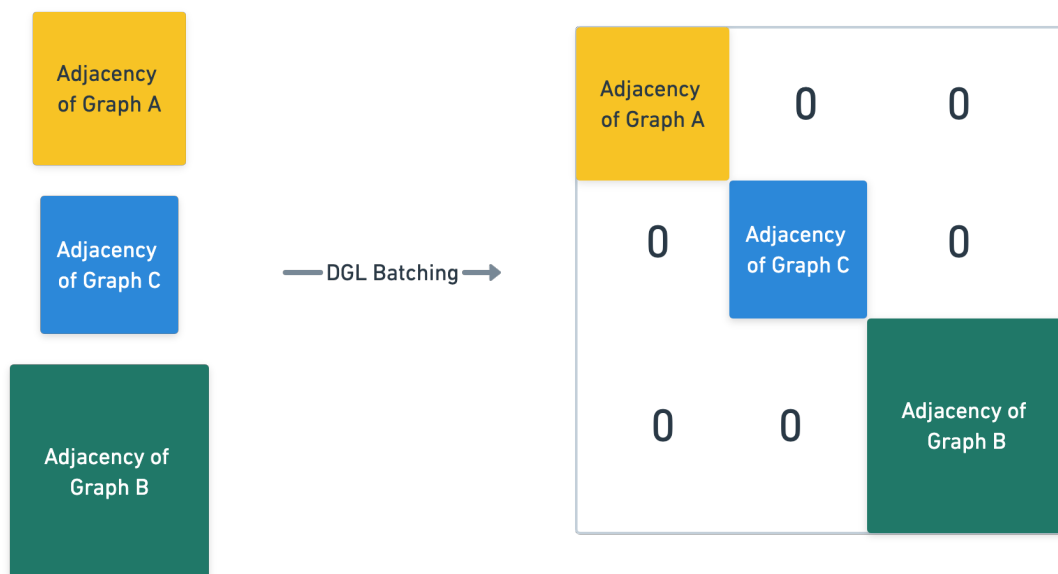


Figure 3.7: Batching of graphs to represent them as a single large graph for more efficient training.

neural networks. Figure 3.7 also illustrates three adjacency matrices which represent the connectivity of three different graphs of different sizes.

2. **Message Passing to update node features.** The next step is to update the features for nodes using message passing. We follow the work of Altae-Tran et al. (2017) and propose a similar architecture using the DGL Library. Table 3.6 illustrates the architecture used. The two main components used for message passing are the graph convolution (Kipf and Welling, 2016) layer, followed by a maximum function aggregating the node features with the maximum value of its neighbours and the node itself. Each graph convolution layer aggregates the neighbouring features and transforms this aggregated representation using a linear projection, over which we apply a non-linear function. The maximum pooling layer is applied after every convolutional layer, and we highlight that this is different from the final read-out function. The maximum aggregation layer merely updates node information through message passing, and does not apply any coarsening to our graphs.
3. **Readout function to aggregate node/edge features into graph-level representation.** Sum pooling is applied as the read-out function, which sums over the node features of the graph.

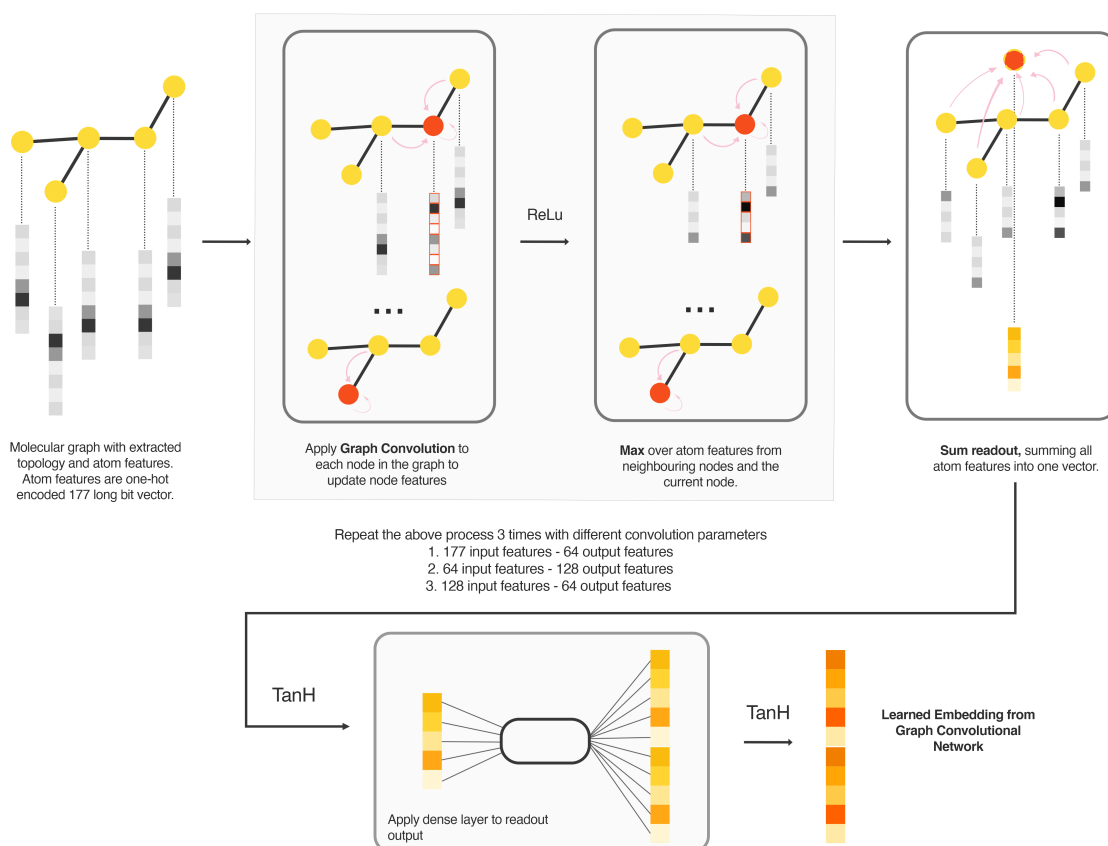


Figure 3.8: Learning an embedding through a Graph Convolutional Network (GCN). The molecule, represented as a graph object with nodes, edges and atom features is processed using graph convolutions. Each convolution layer is followed by a max message passing function over the current and neighbouring nodes. The convolution layers are tabulated in Table 3.6. After this process, a sum readout is applied to aggregate all atom features into one vector. A TanH function is applied to this vector, and the output is processed using a dense linear layer. The resulting embedding is once again applied a non-linear TanH function to yield the final learned molecular embedding.

$$r = \sum_{n=1}^{N_i} x_n \quad (3.1)$$

- Linear Layer.** A linear transformation is applied to the output from the read-out layer, followed by a non-linear activation function, for which we use a hyperbolic tangent function (Tanh), outputting the final molecule embedding.

Layer Type	Input Dimension	Output Dimension	Non-Linearity
GraphConv	177	64	ReLU
Max Pooling	64	64	
GraphConv	64	128	ReLU
Max Pooling	128	128	
GraphConv	128	64	ReLU
Max Pooling	64	64	
SumPool Readout	64	64	TanH
Linear	64	128	TanH

Table 3.6: Graph Convolution Network Architecture

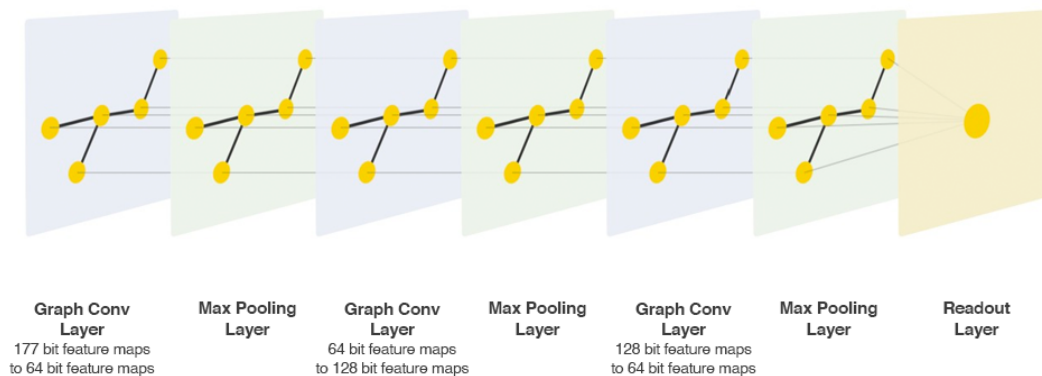


Figure 3.9: Graph processing layers in our GCN implementation. The graph convolution layers apply operations on each individual node’s feature maps based on neighbouring nodes. The ReLU function is applied after each convolutional layer, and the TanH function is applied after the final readout layer. The resulting vector from the readout layer is further processed using a neural network as shown in Figure 3.8.

3.4.3 | Training a Few-shot Machine Learning Model

The generated molecular embeddings from the GNN are passed to the few-shot learning architectures introduced in Chapter 2. The success of a few-shot learning model for metric-based meta-learning is dependent on the effectiveness of the kernel k_θ , which measures the similarity between data samples (see Equation 3.2) using a metric or distance function. The models discussed in this section, excluding the benchmark model, use the embeddings generated from the GNN, presented in the support and query sets, to learn the kernel function.

$$P_\theta(y|\mathbf{x}, S) = \sum_{(\mathbf{x}_i, y_i) \in S} k_\theta(\mathbf{x}, \mathbf{x}_i) y_i \quad (3.2)$$

where

P_θ = Probability over a set of labels y

S = Support set

k_θ = Kernel function

x_i = Data sample from S

y_i = Data label from S

x = Query data sample

To be able to compare the model's effectiveness objectively, the GNN architecture presented in Section 3.4.2 remains unchanged in all experiments, including the benchmark. Training hyperparameters and optimisations are presented in Section 3.5.

3.4.3.1 | Benchmark Models

For the benchmark models, we make use of a random forest model and a graph convolutional network trained through conventional supervised learning. The episodic learning explained previously is not used, but instead we sample a support set directly from the test target and train a model using only the few molecules in the support set. The trained model is then used to predict the activity of the remaining molecules against the target being tested. The random forest model is trained on ECFPs with a size of 2048 bits, as opposed to graph representations, using an ensemble of 100 decision trees. The Scikit-Learn library is used to build the random forest model. The architecture designed for the graph convolution network is outlined in Table 3.7.

Layer	Input Dimension	Output Dimension	Non-Linearity
GraphConv	177	64	Relu
Max Pooling	64	64	
GraphConv	64	128	Relu
Max Pooling	64	64	
GraphConv	128	64	Relu
Max Pooling	64	64	
Sum Pooling	64	64	TanH
Linear	64	128	TanH
Linear	128	1	Sigmoid

Table 3.7: Benchmark Neural Network for Few-Shot Learning

3.4.3.2 | Siamese Networks

Siamese networks (Koch et al., 2015) are composed of two identical networks, with shared weights and parameters, taking in a pair of data samples as inputs. The outputs from the networks are compared to learn the relationship between them. For the Siamese Networks, the molecular embeddings in the support and query sets are not created as explained previously with the GraphDataLoader as these need to be learned in pairs. The following is the process employed for learning a classifier using Siamese Networks. The process is repeated for all training tasks.

1. Generate a list of all possible pairs between training data. If both data samples in the pair have the same target, the pair's label is set to 1, and 0 if otherwise.
2. Create a twin network using the GNN architecture to embed two molecular graph inputs into latent space.
3. Calculate the L1 distance between the molecule embeddings. This is achieved by calculating the absolute difference between the embeddings $|mol_i - mol_j|$.
4. The distance between the two embeddings is passed through a linear feedforward layer with 128 nodes and an output of 2, followed by a sigmoid function to output the probability.
5. As we are dealing with binary classification, the binary cross entropy loss is calculated and back-propagated to the network.

In our implementation, the final sigmoid layer is not defined explicitly during training because this is applied automatically within the PyTorch `BCEWithLogitsLoss` function to compute the loss. Therefore, during testing, the sigmoid function is added explicitly to compute the probabilities of the outputs from the few-shot learning networks.

As a note on the three architectures that follow, training is done using the episodic learning explained in Section 3.4.1, using the support set compositions in Table 3.5.

3.4.3.3 | Matching Networks

Matching Networks builds on Siamese Networks, but instead of learning a metric function over pairs of data, the classifier learns how to define a probability distribution of output labels from query/test examples using a support set S . The classifier outputs a sum of attention weighted labels from the support set to predict the similarity between the test example and the samples from the support set. We use the same embedding function for the support and query sets to compute the molecular embeddings. Subsequently, the cosine similarity between pairs of data points between the support and query sets is computed, which is then normalised by a softmax function. As proposed in Vinyals et al. (2016), Fully Contextual Embeddings (FCE) are used in our implementation. Taking single data points to learn an embedding function limits the ability of embedding the molecules effectively into latent space. Therefore, a bidirectional long-short term memory (LSTM) is used, taking as input the whole support set to adjust the embedding based on the other support samples.

Two functions are defined. $g_\theta(x_i, S)$ encodes x_i , a data sample from the support set, in the context of the whole support set S , using a bidirectional LSTM. The LSTM transforms our support set embeddings by adding the forward and backward activations to the original support image embeddings. Subsequently, $f_\theta(x, S)$, encodes the query sample x and trains the LSTM with read attention over the support set. The hidden state is updated over 10 processing "read" steps, until eventually the hidden state is equivalent to the aforementioned $f_\theta(x, S)$. Throughout each iteration, the hidden state and the output from the attention function are added together. To train the network using stochastic gradient descent, the cross entropy loss is computed for each query prediction.

3.4.3.4 | Prototypical Networks

Prototypical Networks (Snell et al., 2017) have similarities to the Matching Networks described above, but instead of considering the individual support set embeddings, the mean vector of the embeddings for each class within the support set is taken. This mean

vector for each class is referred to as the *prototype*. Another improvement the authors of the original paper make over Matching Networks (Vinyals et al., 2016), is the use of Euclidean distance rather than the Cosine distance. In order to classify query data samples, the softmax of the inverse of the euclidean distances between each query and each prototype is taken. To train the network through stochastic gradient descent, the negative log likelihood loss is used.

3.4.3.5 | Relation Networks

For the Relation Networks (Sung et al., 2018), the classification of query samples is not done directly in latent space from the embeddings. The embeddings for the support and query samples are generated using the GNN. Following this step, the feature maps concatenations are created by concatenating the query samples with each data sample within the support set. The feature map concatenation therefore has a size double the length of the embedding generated by the GNN.

The relationship between the queries and the different classes within the support set is captured by passing these feature map concatenations through a feed forward neural network $g_{\theta}([x_i, x_j])$ to predict a relation score. $[,]$ is the concatenation between each support set data sample x_i and the query data samples x_j . The architecture of this function is defined in Table 3.8. The loss function used is Mean Squared Error (MSE) loss as proposed in the original paper.

Layer Type	Input Dimension	Output Dimension	Non-Linearity
Linear	256	128	ReLU
Linear	128	64	ReLU
Linear	64	8	ReLU
Linear	8	2	Sigmoid

Table 3.8: The architecture for generating the relation score using function g_{θ} .

3.4.4 | ECFPs vs GCNs Learned Embeddings Experiments

We also carry out a number of experiments to determine whether learned embeddings through GCNs are superior to Extended-connectivity fingerprints (ECFP) (Rogers and Hahn, 2010). ECFPs are generated from the SMILES strings using the DeepChem CircularFingerprint featuriser, with a radius of 2 and a size of 2048 bits. The featuriser

discards chirality information and considers bond order in the fingerprint generation. The non-linear activation functions used emulate those used in the GCN architecture. In fact, the last layer is also activated using a hyperbolic tangent function. The neural network components used to process the ECFPs is tabulated in Table 3.9. We choose the best performing few-shot learning technique and compare performance based on ROC and PRC scores between the few-shot learning models when trained using ECFP embeddings versus when trained on learned embeddings through a GCN. For the set of ECFP experiments, we switch the *Learn molecule embedding with GCN* component in Figure 3.5 with the neural network in Table 3.9.

Layer Type	Input Dimension	Output Dimension	Non-Linearity
Linear	2,048	1,000	ReLU
Linear	1,000	500	ReLU
Linear	500	128	TanH

Table 3.9: Neural Network Architecture for ECFPs.

3.5 | Training Process and Hyper-parameters

This section outlines the parameters defined for training, along with optimisation techniques to improve the training process. A list of parameters is available in Table 3.10.

Adam (Kingma and Ba, 2014) is used as an optimisation algorithm for stochastic gradient descent. This optimiser combines properties from AdaGrad (Duchi et al., 2011) and RMSProp (Hinton et al., 2012) algorithms, providing an algorithm which works well with sparse gradients and does not require stationary objectives.

When learning stagnates, learning benefits from reducing the assigned learning rate. For this, we make use of PyTorch’s `ReduceLRonPlateau`, which reduces the learning rate when the loss value stops improving over a number of patience number of episodes. The patience parameter is set to 200, which means that if the loss does not decrease after 200 episodes, the learning rate is reduced by half.

3.5.1 | Performance Monitoring

All training is run in Jupyter Notebooks within Colab (see Appendix A). Monitoring of epochs is carried out using the TQDM library, which can be configured to visualise updates to the metrics during training. The progress bars are configured to showcase

Network	Type	Value
Siamese Nets	Learning Rate	0.001
Siamese Nets	Loss Function	Binary Cross Entropy Loss
Matching Nets	Learning Rate	0.01
Matching Nets	Loss Function	Binary Cross Entropy Loss
Prototypical Nets	Learning Rate	0.001
Prototypical Nets	Loss Function	Negative likelihood Loss
Relation Nets	Learning Rate	0.0001
Relation Nets	Loss Function	Binary Cross Entropy
All	Optimizer	Adam
All	Scheduler	ReduceLROnPlateau
All	Max Episodes	10,000
All	Query Samples	128
All	N-Way	2
All	ReduceLROnPlateau Patience	200
All	ReduceLROnPlateau Factor	0.5

Table 3.10: Hyperparameters and Optimisation

the episode number, the loss value, the accuracy and the current learning rate. The loss functions used to train through gradient descent are the same used in the respective work for Siamese, Matching, Prototypical and Relation Networks. Tensorboard⁷ is a Tensorflow visualisation toolkit, however, it can be extended to be used in PyTorch too. This toolkit provides visualisation for ML experiments, allowing us to track and visualise metrics generated during training such as the loss improvement, highlighting convergence points during training. The SummaryWriter module provided for PyTorch by the Tensorboard library allows the creation of logs, which are consumed and visualised by Tensorboard. Tensorboard was used to monitor loss graphs during training.

3.6 | Testing

All generated models are saved to a PyTorch state dictionary file (.pt). As is emphasised by Vinyals et al. (2016) and Snell et al. (2017), training and testing conditions should match when doing few-shot learning. Therefore, the same support set composition used

⁷<https://www.tensorflow.org/tensorboard>. Last Accessed: 11 Nov 2021.

to train the model is used during test time. For example, if during training we do 10-shot learning, testing is carried out with 10-shot support sets. We remind the reader that testing is carried out on a new, previously unseen target. Tables 3.1, 3.2, and 3.3 show the targets reserved for testing. After the support set has been sampled, the rest of the data for the target being tested is used as query/test data. This process is repeated 20 times, and the mean and standard deviation of the ROC and PRC scores from these 20 rounds are reported as the final classification result.

3.7 | Evaluation

The evaluation metrics used are the Receiver Operating Characteristic (ROC) curve, and the Precision-Recall Curve (PRC). To determine the predictive power of our classifier, we make use of the ROC Area under the curve (AUC) (ROC-AUC) as this provides a clearer picture of the relationship between the true positive and the false positive rate. The ROC-AUC affords a more nuanced approach than accuracy as it provides visibility into thresholds one can utilise to ameliorate predictions. While (Altae-Tran et al., 2017) only report the ROC scores in their paper, we take into consideration the highly imbalanced nature of the datasets used and also introduce the PRC metric. The PRC is more robust at determining the effectiveness of the positive (active) class in a classifier, as explained in Chapter 2 in Section 2.4.5. We make use of the Sci-kit Learn library to compute the ROC-AUC, PR-AUC and overall accuracy using the `roc_auc_score`, `average_precision_score` and `accuracy_score` modules. Confusion matrices, ROC and PRC plots are generated for each experiment. These are generated using the results from the round which obtained the median ROC score from the 20 test rounds. Confusion matrices are beneficial as they provide insight into true or false positives and true or false negatives.

We apply statistical analysis on the ROC and PRC scores from the 20 test rounds for each experiment to establish whether there are significant differences between the few-shot learning models. The scores are compared against those of the model that obtained the best result for the same conditions. Comparing the results between two models is carried out using the Mann-Whitney U-test, also referred to as the Wilcoxon rank sum test (Mann and Whitney, 1947). This non-parametric technique is used to test for differences between two independently sampled groups. Unlike the parametric t-test, non-parametric tests assume no particular distribution (McKnight and Najab, 2010). The Mann-Whitney U-test tests for the null hypothesis H_0 , stating that there is equal probability for the values in X being greater than those in Y and the probability

Package	Version	Description
PyTorch	1.9.0	Machine learning framework
Scikit-Learn	1.0.1	Machine Learning Library
Deep Graph Library (DGL)	0.7.2	Deep learning on graphs
DGL-LifeSci	0.2.8	Cheminformatics graph functions
RDKit	2021.09.2	Cheminformatics Toolkit
DeepChem	2.6.0.dev	Cheminformatics Machine Learning
Pandas	1.1.5	Data manipulation and preparation
Numpy	1.19.5	Adds support for multi-dimensional arrays
ChemBL Structure Pipeline	1.0.0	Used to standardise molecules
NetworkX	2.6.3	Used to visualise graphs
TQDM	4.59	Progress bars library
SciPy	1.7.1	Statistical Analysis

Table 3.11: Python libraries utilised for this project.

for the values in Y to be greater than X . X and Y in our problem scenario represent the ROC or PRC values obtained from two different few-shot learning models for a specific support-set composition experiment. The SciPy library is used to compute the Mann-Whitney U-tests. We set a significance level (α) to 0.05, so the corresponding confidence level is 95%. If the resulting p-value is less than α , we reject the null hypothesis and accept the alternate hypothesis, indicating that there is significant difference between the two distributions. On the other hand, if the p-value is greater than α , we accept the null hypothesis, indicating that there is no statistical significant difference between the two distributions.

3.8 | System and Software Specifications

This research project was developed using Python 3.7. Most packages were installed using Pip 21.0.1, however, Conda 4.10.3 was also used to install packages not found on the Python Package Index (PyPi)⁸. Pip and Conda are package management systems for Python, allowing users to conveniently install and run packages and their dependencies.

⁸Accessed from: <https://pypi.org/>. Last Accessed: 07 Nov 2021

Type	Model	Details
CPU	Intel (R) Xeon	2.20Ghz 4 Cores
GPU	Nvidia Tesla P100	16GB using Cuda 11.1
RAM	N/A	25GB

Table 3.12: Hardware provisioned in Google Colab.

All the experiments were run on Google Colaboratory⁹, *Colab* in short. Colab is a hosted Jupyter notebook¹⁰ service, providing access to computational resources including CPUs and GPUs to run Python code. These Colab notebooks can be accessed and run in web-browsers such as Chrome, Firefox and Safari. For this study, we upgraded from the free tier to Colab Pro to get access to faster GPUs, more memory and longer runtimes without disconnecting.

3.9 | Summary

In this chapter, we explained in detail the methodology employed for this study. First, we provided an overview of our few-shot machine learning architecture. This overview was followed by an in-depth report of each component, allowing the reader to understand the whole few-shot learning process. Our pipeline first starts from loading molecular data, standardising SMILES strings and generating the corresponding molecular graph or ECFP representations. The molecular representations are sampled to create a series of support and query sets, which are used to recreate the conditions of few-shot learning during training. The support and query sets are used to learn a molecular embedding using a graph convolutional network (GCN). The embeddings are then processed using the Iterative-Refinement LSTM (IterRefLSTM) from the state of the art work by Altae-Tran et al. (2017). The resulting embeddings are finally processed using the few-shot learning technique of choice. We implemented Siamese, Matching, Prototypical and Relation Networks for this component of our few-shot learning pipeline. Testing is then carried out on new, unseen experimental assays, using only a small support set to generalise for the new target. Evaluation is carried out using ROC and PRC scores, which are finally evaluated using the Mann-Whitney U-test. Results are presented and discussed in the following chapter.

⁹Accessed from: <https://colab.research.google.com/>. Last Accessed: 07 Nov 2021

¹⁰Accessed from: <https://jupyter.org/>. Last Accessed: 07 Nov 2021

Results & Evaluation

In this chapter, we present and discuss the results in accordance with the defined aims and objectives around which this study is designed. First, we revisit the aims and objectives to set the scene for the results to follow. We then present the results for the developed machine learning models for the three datasets, which include the following.

1. **Tox21**. Dataset mainly used for lead optimisation, containing toxicity data for 12 targets (NIH, 2014).
2. **Maximum Unbiased Validation (MUV)**. Dataset based on PubChem BioAssays, used for validating virtual screening techniques against 17 different targets (Rohrer and Baumann, 2009).
3. **Directory of Useful Decoys (Enhanced) (DUD-E)**. This dataset is used to benchmark virtual screening techniques by introducing a number of active compounds against specific targets. For each active, a number of *decoys* with similar physical properties but different topologies are made available. For this research study, we made use of the GPCR subset of the DUD-E dataset rather than the whole dataset (Mysinger et al., 2012).

All these datasets are highly imbalanced, where the inactive/decoys greatly outnumber the number of actives. This defining feature of these datasets presents a challenging problem, but is also further evidence that low-data machine learning is highly beneficial in this domain. We first present the work we reproduced from Altae-Tran et al. (2017), which we also test on a subset of the DUD-E dataset, which was not explored in the original study. The reproduced work includes Siamese Networks (Koch et al., 2015) and the Matching Networks (Vinyals et al., 2016) with the Iterative Refinement LSTM (IterRefLSTM), which obtained the best results in Altae-Tran et al. (2017).

This is followed by presentation and discussion of the results for two newly proposed machine learning models in this domain, which are based on work of Vinyals et al. (2016) for Matching Networks. These machine learning models include the Prototypical (Snell et al., 2017) and Relation (Sung et al., 2018) Networks. Finally, we evaluate the results with the state of the art, which is identified to be the work of Altae-Tran et al. (2017).

4.1 | Revisiting aims and objectives

In Chapter 1, we presented the main aim of the study, which is to determine the efficacy of low-data machine learning for LBVS for hit identification and lead optimisation in the drug discovery process. The goal is to train a machine learning model that can generalise well enough to classify the activity of molecules in a new experimental assay using only a few training examples as the support set. We reiterate that this is not simply an application of few-shot learning methods to molecular data. The analogous learning task for such techniques would be to learn a classifier in a fixed experimental assay that can predict the activity of molecules in a new molecular scaffold given only a small support set from this scaffold. Altae-Tran et al. (2017) compare the analogous learning challenge in few-shot learning for molecular data to training a computer vision model to perform object recognition, and then performing object localisation at test time given only a small amount of data. This method of training is also different from the conventional supervised machine classification approach, where we have ample data points for each class/target, which are used for training in order to correctly classify unseen data points for the classes/targets we trained on. In this study, we attempt to classify the activity of molecules in a *new* experimental assay, using only a small support set from the previously unseen experimental assay. Training for few-shot learning is carried out in a series of episodes, framed as *N-way K-shot* classification tasks. These classification tasks match the conditions at training time with those during testing, as proposed by Vinyals et al. (2016). The tasks in this research are binary classification tasks, therefore *N* is always set to two to represent the active and the inactive/decoy class respectively. Experiments with varying values of *K* are carried out to generate the support sets, with a minimum of one data point, to a maximum of 10 data points per class. The combinations for *K* active and inactive/decoy classes are not exhaustive, but we follow the support set composition used in Altae-Tran et al. (2017) to directly compare results with this study. The rest of the data is sampled as query data points. In episode generation, 128 balanced queries (i.e. 64 actives and 64 inactive/decoys) are

sampled for training.

The rest of the chapter will introduce the results obtained for the developed machine learning models, which are highlighted once again in Section 4.3. As per our objectives, we also run a number of experiments using ECFPs rather than embeddings created from graph convolutions to identify whether either one can ameliorate the generalisation of few-shot learning models for the prediction of molecular activity in new, unseen experimental assays.

Table 4.1 shows the excessive imbalance of the data used, highlighting the scarceness of data on active compounds in this domain. Hence, the accuracy metric is disregarded for the evaluation of results as this metric is not reliable when evaluating highly imbalanced datasets. In imbalanced scenarios, the model can easily classify the majority class, thus achieving high levels of accuracy. However, looking deeper into the confusion matrix usually uncovers the ineffectiveness of such models. Hence, we evaluate results based on the confusion matrices, the Receiver Operator Characteristic (ROC) Area Under the Curve (AUC) (ROC-AUC) score, and the Precision-Recall Curves (PRC) AUC (PR-AUC) scores. We go into further detail in Section 2.4.5 about the advantages and limitations of using the ROC-AUC and PR-AUC scores.

Dataset	Actives	Inactives/Decoys
Tox21	4,149 (7.04%)	54,746 (92.96%)
MUV	347 (0.20%)	175,990 (99.80%)
DUD-E (GPCR)	1,249 (1.45%)	84,856 (98.55%)

Table 4.1: Number of actives and inactives/decoys across all targets in the datasets used. Figures in parentheses show the percentage of the total compounds in the dataset. These figures highlight the excessive imbalance of the data.

4.2 | Benchmark Machine Learning Models

Following the methodology of Altae-Tran et al. (2017), we made use of a Random Forest model and a Graph Convolutional Network (GCN) to build a baseline to benchmark the purpose-built few-shot learning models. For the random forest model, ECFP representations of the molecules of size 2048 bits are used for the classification task. Meanwhile, the same GCN architecture used for the few-shot learning models is used for the benchmark. The only addition to the architecture is a final linear layer that takes as input 128

features, which is the size of the embedding used for the experiments to follow, and outputs a feature of size one, onto which we apply a non-linear function, in this case a Sigmoid function, to output the probabilities for a Boolean target (0, 1).

These two models are only trained on a small support set, sampled from the targets assigned for testing. For example, the notation $K+/K-$ used in the tables to follow signify the sampling of N actives and N inactives/decoys, which are in turn used for training. Testing is carried out on the remaining data for the designated target. For the benchmark models, this process of sampling, training, and testing is repeated 20 times. The means and standard deviation from these 20 runs are reported accordingly.

4.3 | Few-shot Machine Learning Results

In this section, we present the results obtained for the four machine learning models tailored for few-shot learning. The following list provides a brief overview of each machine learning model used, however, refer to Section 2.5 for a more in-depth overview. The first step in our research was the replication of the results of Altae-Tran et al. (2017) through the implementation of the Matching Networks with Iterative Refinement LSTM (IterRefLSTM), which was established to be the state of the art model in this domain. We also reproduce the Siamese Networks, although the state of the art model affords superior performance as per the literature (Altae-Tran et al., 2017). After successful replication of the state of the art, we apply Prototypical Networks and Relation Networks, previously applied for computer vision in the literature, for this problem domain. The same IterRefLSTM is also applied to the embeddings in these two architectures, so we are just changing the metric-based component in our architecture. We recapitulate the few-shot learning architectures utilised in our experiments in the following list.

- **Siamese Networks** (Koch et al., 2015). A network consisting of twin GCNs whose outputs are connected and the distance between them is calculated to classify the inputs. These networks are always fed in pairs of data points.
- **Matching Networks** (Vinyals et al., 2016). Reproduction of the work by Vinyals et al. (2016) and Altae-Tran et al. (2017). The former study proposes the Matching Networks, while the latter proposes the established state of the art for low-data drug discovery which is an improvement on the Matching Networks through the introduction of IterRefLSTM. In Matching Networks, two embedding functions embed the support and query data points. The LSTM uses softmax attention to further enhance the support and query embeddings in the context of each other.

The cosine distance between the support and query data points is taken to classify the queries.

- **Prototypical Networks** (Snell et al., 2017). This type of network is similar to the Matching Networks, but instead of comparing the query support to every support data point, a *prototype* is calculated, which takes all the support data points per class and creates an embedding by averaging over the embeddings. The euclidean distance between the query data points and the prototypes is calculated for classification.
- **Relation Networks** (Sung et al., 2018). A GCN creates embeddings for the support data points and the query data points. The query embeddings are then concatenated to the support embeddings, and these feature map concatenations are passed through a neural network to compute a relation score, which is used to classify the queries.

While the work of Altae-Tran et al. (2017) is open-sourced and made available in the DeepChem library, at the time of writing of this study, all the code was deprecated and proved problematic to run. In addition to being dependent on older versions of TensorFlow, the relevant code is also coupled with other components of the DeepChem library, which were in turn written for newer versions of libraries such as TensorFlow. As a result, rewriting the code to run it again directly in DeepChem proved to be difficult. Therefore, we reproduced the work to the best of our abilities without tightly coupling our work with the DeepChem library. We also joined DeepChem weekly developer meetings in order to gain a better understanding of DeepChem and the outdated low-data learning code.

4.3.1 | Evaluation Overview

Evaluation is carried out using ROC-AUC and Precision Recall Curve AUC (PR-AUC) metrics. The original work of Altae-Tran et al. (2017) only reports ROC results, however, this metric alone does not fully encompass the nature of the performance of the machine learning models in this problem domain due to the imbalanced nature of the data. In virtual screening, the detection of rare events (equivalent to our minority active class) holds significant importance, as active compounds against a specific target should be identified from the compound database. However, we do not disregard the importance of correct classification of the majority inactive/decoy class as this is also important for filtering out thousands of screened compounds. Table 4.1 highlights the imbalance of

the data used for this study. As the active class is the minority class, PRC are used to evaluate how well the model can classify the active class. A high area under the PRC indicates high recall and high precision.

- **High Precision** is attributed by a low false positive rate, meaning compounds classified as active when in fact they are inactive/decoys.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

- **High Recall** is related to a low false negative rate, meaning active compounds incorrectly classified as inactive/decoys.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

The ideal scenario for predicting the minority active class is thus one where we achieve high recall and high precision. As our data contains a lot of negative examples, there is a higher chance of these being predicted as false positives. On the other hand, we have much fewer active examples which could be predicted as false negatives. Given that the active class is in such a minority, even a small false positive rate could result in high numbers of false positives, due to the high number of the negative class examples. In this scenario, the precision will be low as we are predicting a lot of false positives when compared to true positives. We can also have a scenario of high recall with low precision. In this scenario, we have a high number of incorrect predictions as the model returns a lot of false positives, but it correctly predicts most of the active class as it has high recall. On the other hand, if we achieve high precision with low recall, most of the predictions are correct as we have a high number of true positives when compared to false positives.

Each model is evaluated 20 times per target, with a randomly sampled support set from a specific test target per round. The rest of the data for the target is used for testing. The tabulated results are the mean values from these 20 rounds, along with the standard deviation for each mean. Where applicable, graphs and confusion matrices are presented for the median round based on the ROC-AUC metric from the 20 testing rounds. The non-parametric Mann-Whitney U rank test, also referred to as the Mann-Whitney Wilcoxon Test, is used to determine whether the results from the test rounds between different machine learning models are significantly different. We report whether there is a significant difference between the scores obtained between two different machine learning models. The null hypothesis when comparing the results from two machine

learning models is that there is no significant difference between the two distributions. We choose a significance level α of 0.05, which gives us a confidence level of 95% when rejecting the null hypothesis and accept the alternative hypothesis, which resolves to the distributions being significantly different.

4.3.2 | ROC-AUC and PR-AUC Scores

To evaluate the performance of our machine learning models, we make use of the ROC-AUC and Average PR-AUC (labelled as PRC) for five different support set compositions, tested with the two benchmark models, and four few-shot machine learning models. We reiterate that for the support set composition (the number of examples per class), we follow the same ones utilised by Altae-Tran et al. (2017). One limitation our results have is for Siamese Networks. The nature of the architecture dictates that the model takes a pair of samples as inputs, and outputs a Boolean value if the classes belong to the same class or not. When combining queries with the data points in the support set, this leads to a three class problem, rather than a binary classification one as per the rest of our study. We have the following combinations (i) both inactive/decoys, (ii) one inactive and one active or vice versa, and (iii) both active compounds. The literature (Altae-Tran et al., 2017; Snell et al., 2017; Sung et al., 2018; Vinyals et al., 2016) suggests that the other proposed architectures are superior to Siamese Networks. Adding to the fact that we assign more importance to classifying the active class, we frame the Siamese Network data inputs for binary classification. Hence, if the input pairs are both inactive, they are still assigned a value of zero, and the positive/True class is only assigned in the case where both molecules are active. As a result, when comparing and evaluating results against Siamese Networks, we mainly consider the PRC scores. However the other few-shot learning models in our experiments outperform the Siamese Networks, as supported by literature. In the tables to follow, the best performing metric is highlighted in bold, excluding the ROC score for the Siamese Networks due to the aforementioned reasons.

4.3.2.1 | Few-Shot Learning on Tox21

For the Tox21 dataset, three targets were reserved for testing, namely SR-HSE, SR-MMP and SR-p53. The rest of the targets (refer back to Chapter 3 Section 3.2 Table 3.1) are used for training. The comparisons reported below are between Matching Networks (MN), Prototypical Networks (PN) and Relation Networks (RN), as these three outperform the Random Forest (RF), Graph Convolutional Networks (GCN), and Siamese Networks (SN) significantly in ROC and PRC scores, taking into consideration our previous observation about ROC scores for Siamese Networks. A discussion on each target ensues

in the following sub-sections. We reiterate the number of actives and inactives for each target, and then present the results for each support set composition. The support set composition is denoted by K^+/K^- , where the K stands for the number of actives and the number of inactives respectively. Figures 4.1 and 4.2 show the box plots for results across targets for the $10^+/10^-$ support set composition for all machine learning models.

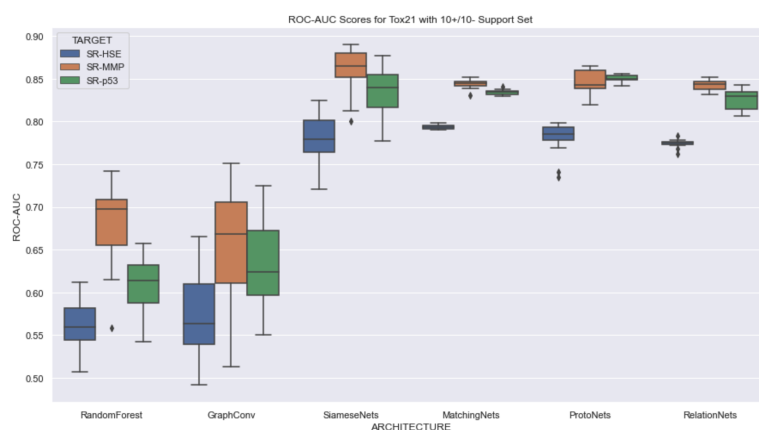


Figure 4.1: Box plots for Tox21 ROC scores for $10^+/10^-$ Support Set, showing clear separation between benchmark models and few-shot learning models.

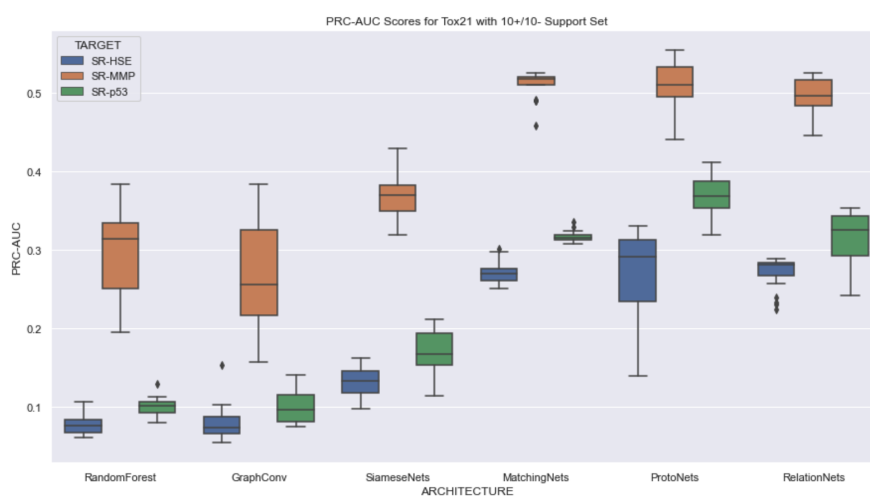


Figure 4.2: Box plots for Tox21 PRC scores for $10^+/10^-$ Support Set, showing clear separation between benchmark models and SNs when compared to MNs, PRs, and RNs.

SR-HSE - 372 Actives - 6,095 Inactives

Table 4.2 contains the tabulated results for the SR-HSE target. From the statistical analysis below, it is evident that the PN dominates in the PRC scores, followed by RN and

finally the MN. For ROC scores, the results are less clean-cut. For the largest support set, the Matching Networks obtain the best results, while for one-shot learning, Prototypical Networks outperform the other networks.

- **10+/10-. Best ROC - MN. Best PRC - MN/PN/RN.** The best ROC and PRC metrics are obtained by the MN model. The ROC metric for MN is established to be significantly different from both the PNs ($U=297.00$ and $p=0.009$) and RNs ($U=400.00$ and $p=0.000$).

However, there is no significant difference between PRC scores from both PN ($U=165.00$ and $p=0.351$) and RN ($U=167.00$ and $p=0.380$).

- **5+/10-. Best ROC - RN. Best PRC - PN/RN.** The best ROC and PRC metrics are obtained by the RN model. The ROC metric for RN is established to be significantly different from both the MN ($U=400.00$ and $p=0.000$) and PN ($U=399.00$ and $p=0.000$).

There is also significant difference between PRC scores from MN ($U=307.00$ and $p=0.004$), but none from the PN ($U=180.00$ and $p=0.598$).

- **1+/10-. Best ROC - MN. Best PRC - PN.** The ROC score for MN is significantly better than the score for PN ($U=396.0$ and $p=0.000$) and RN ($U=400.0$ and $p=0.000$), however the PRC score for PN is significantly better than MN ($U=400.0$ and $p=0.000$) and RN ($U=391.0$ and $p=0.000$).

- **1+/5-. Best ROC - RN. Best PRC - PN.** RN ROC score is significantly better than both MN ($U=400.0$ and $p=0.000$) and PN ($U=400.0$ and $p=0.000$). However, the PRC for PN is significantly better than both MN ($U=381.0$ and $p=0.000$) and RN ($U=382.0$ and $p=0.000$).

- **1+/1-. Best ROC - PN. Best PRC - PN/RN.** For one-shot learning, PNs obtain significantly better ROC scores than MN ($U=369.0$ and $p=0.000$) and RN ($U=377.0$ and $p=0.000$). On the other hand, while RN obtained the best PRC score, there is no significant difference than those obtained through PN ($U=217.0$ and $p=0.655$).

SR-MMP - 918 Actives - 4,892 Inactives

Table 4.3 contains the tabulated results for the SR-MMP target. From the statistical analysis below, PNs are consistently PRC performant in all experiments. Meanwhile MNs are ROC performant in most experiments, excluding the 1+/10- where PNs offer better ROC performance.

TOX21 SR-HSE	Metric	RF	Graph Conv	SiameseNet	MatchingNet	ProtoNet	RelationNet
10+/10-	ROC	0.563 ± 0.030	0.572 ± 0.049	0.780 ± 0.028	0.793 ± 0.002	0.782 ± 0.017	0.774 ± 0.004
	PRC	0.078 ± 0.014	0.080 ± 0.022	0.133 ± 0.020	0.272 ± 0.015	0.271 ± 0.055	0.271 ± 0.021
5+/10-	ROC	0.546 ± 0.035	0.565 ± 0.047	0.777 ± 0.067	0.778 ± 0.002	0.772 ± 0.013	0.791 ± 0.003
	PRC	0.074 ± 0.013	0.085 ± 0.020	0.117 ± 0.034	0.266 ± 0.012	0.278 ± 0.045	0.280 ± 0.031
1+/10-	ROC	0.518 ± 0.036	0.540 ± 0.058	0.778 ± 0.168	0.788 ± 0.001	0.780 ± 0.003	0.776 ± 0.001
	PRC	0.064 ± 0.009	0.075 ± 0.019	0.052 ± 0.047	0.208 ± 0.007	0.283 ± 0.016	0.243 ± 0.007
1+/5-	ROC	0.515 ± 0.044	0.503 ± 0.061	0.801 ± 0.162	0.775 ± 0.002	0.773 ± 0.005	0.789 ± 0.001
	PRC	0.064 ± 0.010	0.064 ± 0.013	0.064 ± 0.041	0.235 ± 0.017	0.270 ± 0.018	0.229 ± 0.005
1+/1-	ROC	0.521 ± 0.045	0.520 ± 0.066	0.769 ± 0.084	0.770 ± 0.006	0.779 ± 0.007	0.772 ± 0.001
	PRC	0.066 ± 0.009	0.066 ± 0.015	0.125 ± 0.059	0.230 ± 0.024	0.266 ± 0.029	0.274 ± 0.008

Table 4.2: ROC-AUC and PR-AUC Scores for ML Models on TOX21 SR-HSE Target. Values are mean values with standard deviation over 20 rounds of testing. Best values are highlighted in bold text. The first column shows the composition of the support set as explained in text.

- **10+/10-. Best ROC - MN/PN/RN. Best PRC - MN/PN.** While PN obtained the best ROC score, there is no significant difference between those obtained from MN ($U=213.0$ and $p=0.735$) and RN ($U=230.0$ and $p=0.425$). There is no significant difference for the PRC score for MN when compared to that of PN ($U=214.0$ and $p=0.715$), however, this result is better than that of RN ($U=291.0$ and $p=0.014$).
- **5+/10-. Best ROC - MN/PN. Best PRC - MN/PN.** While PN obtained the best ROC scores, there is no statistical significance between those obtained by PN ($U=250.0$ and $p=0.181$), but there is a significant improvement over those obtained by RN ($U=361.0$ and $p=0.000$). PNs obtain the highest PRC, however there is no significant difference with those obtained through MN ($U=243.0$ and $p=0.250$). In contrast, there is a significant improvement from PRC scores obtained from RN ($U=328.0$ and $p=0.001$).
- **1+/10-. Best ROC - PN. Best PRC - PN.** For this support set combination, PNs outperform MN (ROC $U=366.0$ and $p=0.000$) (PRC $U=386.0$ and $p=0.000$) and RN (ROC $U=337.0$ and $p=0.000$) (PRC $U=378.0$ and $p=0.000$) significantly in both ROC and PRC scores.
- **1+/5-. Best ROC - MN. Best PRC - PN.** MNs significantly outperform PN ($U=400.0$ and $p=0.000$) and RN ($U=400.0$ and $p=0.000$) in the ROC scores. On the other hand,

PNs outperform MNs ($U=274.0$ and $p=0.047$) and RNs ($U=380.0$ and $p=0.000$) in the PRC scores.

- **1+/1-. Best ROC - MN. Best PRC - MN/PN.** For one-shot learning, MNs obtain significantly better results from PN ($U=297.0$ and $p=0.009$) and RNs ($U=361.0$ and $p=0.000$). However, there is no significant difference in PRC scores between MN and PN ($U=361.0$ and $p=0.000$), but a noticeable improvement from RN ($U=329.0$ and $p=0.001$).

TOX21 SR-MMP	Metric	RF	Graph Conv	SiameseNet	MatchingNet	ProtoNet	RelationNet
10+/10-	ROC	0.679 ± 0.047	0.654 ± 0.068	0.860 ± 0.025	0.844 ± 0.005	0.845 ± 0.015	0.842 ± 0.007
	PRC	0.296 ± 0.052	0.270 ± 0.066	0.371 ± 0.029	0.511 ± 0.017	0.510 ± 0.032	0.496 ± 0.023
5+/10-	ROC	0.647 ± 0.055	0.652 ± 0.056	0.858 ± 0.055	0.853 ± 0.007	0.846 ± 0.014	0.842 ± 0.007
	PRC	0.268 ± 0.044	0.273 ± 0.059	0.303 ± 0.064	0.519 ± 0.031	0.527 ± 0.029	0.494 ± 0.038
1+/10-	ROC	0.613 ± 0.063	0.587 ± 0.081	0.876 ± 0.068	0.842 ± 0.002	0.849 ± 0.005	0.844 ± 0.002
	PRC	0.236 ± 0.054	0.221 ± 0.050	0.158 ± 0.124	0.444 ± 0.005	0.508 ± 0.025	0.470 ± 0.011
1+/5-	ROC	0.539 ± 0.078	0.592 ± 0.095	0.815 ± 0.172	0.853 ± 0.001	0.846 ± 0.005	0.846 ± 0.001
	PRC	0.188 ± 0.035	0.226 ± 0.059	0.179 ± 0.099	0.499 ± 0.011	0.508 ± 0.025	0.470 ± 0.007
1+/1-	ROC	0.580 ± 0.060	0.572 ± 0.128	0.845 ± 0.055	0.851 ± 0.008	0.847 ± 0.003	0.840 ± 0.004
	PRC	0.209 ± 0.034	0.219 ± 0.075	0.311 ± 0.084	0.511 ± 0.037	0.506 ± 0.018	0.471 ± 0.022

Table 4.3: ROC-AUC and PR-AUC Scores for ML Models on TOX21 SR-MMP Target. Values are mean values with standard deviation over 20 rounds of testing. Best values are highlighted in bold text. The first column shows the composition of the support set as explained in text.

SR-p53 - 423 Actives - 6,351 Inactives

Table 4.4 contains the tabulated results for the SR-p53 target. Statistical analysis on the results show that for this target, PN outperform all other models in both ROC and PRC scores, excluding the ROC scores for one-shot learning.

- **10+/10-. Best ROC - PN. Best PRC - PN.** PN significantly outperform MNs (ROC $U=400$ and $p=0.000$) (PRC $U=392$ and $p=0.000$) and RNs (ROC $U=399$ and $p=0.000$) (PRC $U=364$ and $p=0.000$) in both ROC and PRC scores.
- **5+/10-. Best ROC - PN. Best PRC - PN.** PN significantly outperform MNs (ROC $U=325$ and $p=0.001$) (PRC $U=326$ and $p=0.001$) and RNs (ROC $U=384$ and $p=0.000$) (PRC $U=366$ and $p=0.000$) in both ROC and PRC scores.

- **1+/10-. Best ROC - PN. Best PRC - PN.** PNs significantly outperform MNs (ROC $U=375$ and $p=0.000$) (PRC $U=400$ and $p=0.000$) and RNs (ROC $U=400$ and $p=0.000$) (PRC $U=397$ and $p=0.000$) in both ROC and PRC scores.
- **1+/5-. Best ROC - PN. Best PRC - PN.** PNs significantly outperform MNs (ROC $U=380$ and $p=0.000$) (PRC $U=315$ and $p=0.002$) and RNs (ROC $U=380$ and $p=0.000$) (PRC $U=400$ and $p=0.000$) in both ROC and PRC scores.
- **1+/1-. Best ROC - MN. Best PRC - PN.** In contrast to the rest of the experiments for this target, MNs outperform PNs ($U=311$ and $p=0.003$) and RNs ($U=382$ and $p=0.000$) in ROC scores. However, PNs still outperform MNs ($U=350$ and $p=0.000$) and RNs ($U=383$ and $p=0.000$) in PRC scores.

TOX21 SR-p53	Metric	RF	Graph Conv	SiameseNet	MatchingNet	ProtoNet	RelationNet
10+/10-	ROC	0.610 ± 0.033	0.635 ± 0.048	0.835 ± 0.029	0.834 ± 0.003	0.850 ± 0.004	0.826 ± 0.011
	PRC	0.101 ± 0.011	0.101 ± 0.022	0.172 ± 0.027	0.317 ± 0.007	0.369 ± 0.027	0.313 ± 0.039
5+/10-	ROC	0.614 ± 0.033	0.614 ± 0.054	0.848 ± 0.060	0.842 ± 0.003	0.852 ± 0.009	0.834 ± 0.002
	PRC	0.102 ± 0.014	0.098 ± 0.023	0.151 ± 0.045	0.323 ± 0.014	0.358 ± 0.042	0.291 ± 0.028
1+/10-	ROC	0.558 ± 0.069	0.548 ± 0.083	0.855 ± 0.148	0.837 ± 0.001	0.848 ± 0.005	0.824 ± 0.001
	PRC	0.084 ± 0.019	0.081 ± 0.023	0.088 ± 0.061	0.249 ± 0.004	0.361 ± 0.030	0.262 ± 0.007
1+/5-	ROC	0.548 ± 0.070	0.582 ± 0.088	0.805 ± 0.154	0.833 ± 0.001	0.840 ± 0.005	0.822 ± 0.001
	PRC	0.083 ± 0.022	0.093 ± 0.028	0.076 ± 0.057	0.283 ± 0.012	0.310 ± 0.023	0.257 ± 0.003
1+/1-	ROC	0.549 ± 0.064	0.553 ± 0.103	0.840 ± 0.061	0.838 ± 0.004	0.834 ± 0.003	0.827 ± 0.001
	PRC	0.079 ± 0.016	0.085 ± 0.027	0.160 ± 0.046	0.316 ± 0.022	0.346 ± 0.021	0.279 ± 0.015

Table 4.4: ROC-AUC and PR-AUC Scores for ML Models on TOX21 SR-p53 Target. Values are mean values with standard deviation over 20 rounds of testing. Best values are highlighted in bold text. The first column shows the composition of the support set as explained in text.

4.3.2.2 | Few-Shot Learning on MUV

For the MUV dataset, five targets were reserved for testing, namely MUV-832, MUV-846, MUV-852, MUV-858, and MUV-859. The rest of the targets (refer back to Chapter 3 Section 3.2 Table 3.2) are used for training. The imbalance in this data is more pronounced and MUV data is more complex as during the generation of the dataset, MUV chooses decoys which are closely embedded to the actives. Hence, siamese (SN), Matching (MN), Prototypical (PN), and Relation (RN) Networks struggle in ROC and

PRC performance, as opposed to the Tox21 experiments. As a result, the Random Forest (RF) models excelled in performance based on ROC and PRC scores when compared to few-shot learning models. The following subsections present these results in further detail.

MUV-832 - 30 Actives - 14,637 Decoys

Table 4.5 contains the tabulated results for the MUV-832 target.

- **10+/10-. Best ROC - RF. Best PRC - RF.** The RF model outperformed all other models significantly, in both ROC and PRC scores - (i) GCN (ROC - $U=310.0$ and $p=0.003$) (PRC $U=395.0$ and $p=0.000$), (ii) SN (PRC $U=400$ and $p=0.000$) (iii) MN (ROC $U=356.0$ and $p=0.000$) (PRC $U=356.0$ and $p=0.000$), (iv) PN (ROC $U=356.0$ and $p=0.000$) (PRC $U=396.0$ and $p=0.000$), and (v) RN (ROC $U=400.0$ and $p=0.000$) (PRC $U=400.0$ and $p=0.000$). From the few-shot learning models, the MN outperformed SN (PRC $U=400.0$ and $p=0.000$), PN (ROC $U=396.0$ and $p=0.000$) (PRC $U=270.0$ and $p=0.060$) and RN (ROC $U=399.0$ and $p=0.000$) (PRC $U=374.0$ and $p=0.000$) in performance.
- **5+/10-. Best ROC - RF. Best PRC - RF.** The RF model outperformed all other models significantly, in both ROC and PRC scores - (i) GCN (ROC - $U=293$ and $p=0.012$) (PRC $U=378$ and $p=0.000$), (ii) SN (PRC $U=398$ and $p=0.000$) (iii) MN (ROC $U=346$ and $p=0.000$) (PRC $U=367$ and $p=0.000$), (iv) PN (ROC $U=350$ and $p=0.000$) (PRC $U=372$ and $p=0.000$), and (v) RN (ROC $U=377$ and $p=0.000$) (PRC $U=399$ and $p=0.000$). From the few-shot learning models, the PNs perform best, however, the ROC and PRC scores are not significantly different than MNs (ROC $U=213$ and $p=0.735$) (PRC $U=137$ and $p=0.091$). However, they are significantly better than SNs (PRC $U=381$ and $p=0.000$) and RNs (ROC $U=316$ and $p=0.002$) (PRC $U=396$ and $p=0.000$).
- **1+/10-. Best ROC - RN. Best PRC - RF.** This is the only instance for this target where a few-shot learning model, in this case the RN, outperforms the RF benchmark significantly in the ROC scores ($U=364$ and $p=0.000$). However, the RF still outperforms all other models in PRC performance - (i) GCN (PRC $U=321$ and $p=0.001$) (ii) SN (PRC $U=363$ and $p=0.000$) (iii) MN (PRC $U=343$ and $p=0.000$) (iv) PN (PRC $U=318$ and $p=0.001$) (v) RN (PRC $U=303$ and $p=0.006$).
- **1+/5-. Best ROC - RF/GCN/PN. Best PRC - RF.** The RF model once again obtains the best ROC and PRC scores. However, the ROC scores are not significantly

different than those obtained through GCN ($U=219$ and $p=0.617$) and PN ($U=194$ and $p=0.882$). On the other hand, the PRC scores from RFs clearly out-perform all other machine learning models.

- **1+/1-**. **Best ROC - RF/GCN/RN. Best PRC - RF.** The RF model once again obtains the best ROC and PRC scores. The ROC scores are not significantly better than those obtained through GCNs ($U=194$ and $p=0.882$) or RNs ($U=243$ and $p=0.250$). However, as observed in other support set sizes, the PRC score from RF out-performs all other machine learning models significantly.

MUV MUV-832	Metric	RF	Graph Conv	SiameseNet	MatchingNet	ProtoNet	RelationNet
10+/10-	ROC	0.824 ± 0.068	0.753 ± 0.073	0.608 ± 0.019	0.720 ± 0.043	0.585 ± 0.041	0.566 ± 0.050
	PRC	0.069 ± 0.047	0.006 ± 0.004	0.001 ± 0.000	0.005 ± 0.001	0.004 ± 0.003	0.003 ± 0.001
5+/10-	ROC	0.747 ± 0.072	0.683 ± 0.080	0.561 ± 0.039	0.651 ± 0.031	0.656 ± 0.021	0.561 ± 0.105
	PRC	0.122 ± 0.058	0.012 ± 0.017	0.001 ± 0.002	0.008 ± 0.001	0.007 ± 0.001	0.003 ± 0.001
1+/10-	ROC	0.601 ± 0.060	0.556 ± 0.076	0.722 ± 0.136	0.615 ± 0.026	0.607 ± 0.014	0.683 ± 0.010
	PRC	0.035 ± 0.034	0.007 ± 0.009	0.003 ± 0.004	0.004 ± 0.001	0.006 ± 0.000	0.007 ± 0.000
1+/5-	ROC	0.579 ± 0.071	0.551 ± 0.115	0.773 ± 0.084	0.514 ± 0.012	0.570 ± 0.021	0.500 ± 0.000
	PRC	0.036 ± 0.033	0.005 ± 0.003	0.003 ± 0.003	0.003 ± 0.000	0.004 ± 0.001	0.002 ± 0.000
1+/1-	ROC	0.582 ± 0.089	0.567 ± 0.155	0.768 ± 0.079	0.496 ± 0.028	0.517 ± 0.010	0.577 ± 0.011
	PRC	0.039 ± 0.038	0.006 ± 0.009	0.005 ± 0.003	0.003 ± 0.000	0.005 ± 0.001	0.004 ± 0.000

Table 4.5: ROC-AUC and PR-AUC Scores for ML Models on MUV MUV-832 Target. Values are mean values with standard deviation over 20 rounds of testing. Best values are highlighted in bold text. The first column shows the composition of the support set as explained in text.

MUV-846 - 30 Actives - 14,681 Decoys

Table 4.6 presents the results for the MUV-846 target. For this target, the benchmark models excel once again. The benchmark models outperform the few-shot learning models significantly, with the RFs having overall better performance than GCNs. The Mann-Whitney U Rank test is reported for the machine learning model which obtained the closest results to the model that obtained the best performance.

- **10+/10-**. **Best ROC - RF/GCN. Best PRC - RF/GCN.** The RF model obtains the best ROC and PRC scores. However, upon a closer look with statistical analysis,

we observe that there is no significant difference between the ROC scores of RF and GCN ($U=268$ and $p=0.068$).

- **5+/10-. Best ROC - RF. Best PRC - RF.** The best ROC and PRC results are obtained by RF. The closest results are those from GCNs, however, the results from RF are significantly better (ROC $U=349$ and $p=0.000$) (PRC $U=328$ and $p=0.001$).
- **1+/10-. Best ROC - RF/GCN. Best PRC - RF/GCN.** While the RF once again obtains the best ROC and PRC results, the results are not significantly better than those obtained from GCNs (ROC $U=210$ and $p=0.797$) (PRC $U=249$ and $p=0.190$).
- **1+/5-. Best ROC - RF/GCN. Best PRC - RF.** GCNs obtain the best ROC scores, however those obtained by RFs are statistically similar ($U=250$ and $p=0.181$). On the other hand, RFs obtain the best PRC score, and also outperform GCNs ($U=299$ and $p=0.008$).
- **1+/1-. Best ROC - RF/GCN. Best PRC - RF.** GCNs obtain the best ROC scores, however, these are not stochastically greater than those obtained by RFs ($U=235$ and $p=0.351$). The PRC scores obtained by RFs and GCNs are not different from each other ($U=262$ and $p=0.096$), however, due to the standard deviation of the two samples, we utilised a one-sided analysis on RFs to determine if the values are stochastically greater than those obtained through GCNs, and this hypothesis was confirmed ($U=262$ and $p=0.048$).

MUV-852 - 29 Actives - 14,622 Decoys

Table 4.7 presents the results for the MUV-852 target. For this target, the benchmark models excel once again. The benchmark models outperform the few-shot learning models significantly, with the RFs having overall better performance than GCNs. The Mann-Whitney U Rank test is reported for the closest performer below.

- **10+/10-. Best ROC - RF/GCN Best PRC - RF.** GCNs obtain the best ROC scores, however, the distributions are statistically similar within our confidence range to the RFs ROC scores ($U=231$ and $p=0.409$). RFs outperform all other machine learning models in PRC scores.
- **5+/10-. Best ROC - RF/GCN. Best PRC - RF.** RFs obtain the best ROC scores, but their distribution is close to that of GCNs ($U=221$ and $p=0.579$). RFs outperform all other machine learning models in PRC scores.

MUV MUV-846	Metric	RF	Graph Conv	SiameseNet	MatchingNet	ProtoNet	RelationNet
10+/10-	ROC	0.876 ± 0.048	0.849 ± 0.045	0.594 ± 0.024	0.715 ± 0.048	0.719 ± 0.036	0.482 ± 0.035
	PRC	0.115 ± 0.053	0.025 ± 0.019	0.001 ± 0.000	0.022 ± 0.012	0.003 ± 0.001	0.001 ± 0.000
5+/10-	ROC	0.833 ± 0.042	0.754 ± 0.065	0.555 ± 0.056	0.453 ± 0.024	0.560 ± 0.033	0.549 ± 0.047
	PRC	0.088 ± 0.055	0.038 ± 0.029	0.001 ± 0.000	0.002 ± 0.000	0.002 ± 0.000	0.002 ± 0.000
1+/10-	ROC	0.676 ± 0.104	0.656 ± 0.123	0.711 ± 0.104	0.437 ± 0.026	0.490 ± 0.017	0.556 ± 0.008
	PRC	0.024 ± 0.031	0.012 ± 0.012	0.001 ± 0.000	0.002 ± 0.000	0.002 ± 0.000	0.002 ± 0.000
1+/5-	ROC	0.661 ± 0.093	0.690 ± 0.124	0.688 ± 0.171	0.388 ± 0.011	0.441 ± 0.012	0.500 ± 0.000
	PRC	0.039 ± 0.034	0.014 ± 0.013	0.001 ± 0.001	0.002 ± 0.000	0.002 ± 0.000	0.002 ± 0.000
1+/1-	ROC	0.624 ± 0.123	0.642 ± 0.159	0.647 ± 0.094	0.518 ± 0.029	0.456 ± 0.025	0.490 ± 0.008
	PRC	0.031 ± 0.040	0.011 ± 0.010	0.009 ± 0.021	0.002 ± 0.000	0.002 ± 0.000	0.002 ± 0.000

Table 4.6: ROC-AUC and PR-AUC Scores for ML Models on MUV MUV-846 Target. Values are mean values with standard deviation over 20 rounds of testing. Best values are highlighted in bold text. The first column shows the composition of the support set as explained in text.

- **1+/10-**. **Best ROC - RF/GCN. Best PRC - RF/GCN.** RFs obtain the best ROC and PRC scores, however with a notable standard deviation. Upon closer investigation, we observe that there is no significant difference between the distributions to those from the scores obtained through GCNs (ROC $U=187$ and $p=0.735$) (PRC $U=260$ and $p=0.108$).
- **1+/5-**. **Best ROC - RF/GCN. Best PRC - RF/GCN.** RFs obtain the best ROC and PRC scores, however, upon closer investigation, we observe that there is no significant difference between the distributions to those from the scores obtained through GCNs (ROC $U=218$ and $p=0.636$) (PRC $U=250$ and $p=0.181$).
- **1+/1-**. **Best ROC - RF/GCN. Best PRC - RF/GCN.** GCNs obtain the best ROC scores, however, we observe that there is no significant difference between the distributions to those from the scores obtained through RFs ($U=182$ and $p=0.636$). The same applies for PRC scores, with the greatest value obtained through RFs, with no stochastic difference to those obtained through GCNs ($U=250$ and $p=0.181$).

MUV-858 - 29 Actives - 14,745 Decoys

Table 4.8 presents the results for the MUV-858 target. For this target, the benchmark models excel once again. The benchmark models outperform the few-shot learning

MUV MUV-852	Metric	RF	Graph Conv	SiameseNet	MatchingNet	ProtoNet	RelationNet
10+/10-	ROC	0.790 ± 0.048	0.803 ± 0.054	0.574 ± 0.019	0.647 ± 0.035	0.639 ± 0.052	0.515 ± 0.042
	PRC	0.137 ± 0.086	0.010 ± 0.004	0.001 ± 0.000	0.002 ± 0.000	0.002 ± 0.000	0.002 ± 0.001
5+/10-	ROC	0.787 ± 0.041	0.779 ± 0.031	0.612 ± 0.031	0.564 ± 0.033	0.592 ± 0.032	0.453 ± 0.026
	PRC	0.130 ± 0.093	0.021 ± 0.023	0.001 ± 0.000	0.002 ± 0.000	0.002 ± 0.000	0.002 ± 0.000
1+/10-	ROC	0.665 ± 0.103	0.654 ± 0.156	0.741 ± 0.129	0.556 ± 0.020	0.591 ± 0.022	0.594 ± 0.008
	PRC	0.036 ± 0.044	0.007 ± 0.005	0.001 ± 0.000	0.002 ± 0.000	0.002 ± 0.000	0.003 ± 0.000
1+/5-	ROC	0.639 ± 0.145	0.607 ± 0.168	0.665 ± 0.126	0.485 ± 0.020	0.523 ± 0.017	0.500 ± 0.000
	PRC	0.052 ± 0.062	0.007 ± 0.006	0.001 ± 0.000	0.002 ± 0.000	0.002 ± 0.000	0.002 ± 0.000
1+/1-	ROC	0.621 ± 0.114	0.674 ± 0.174	0.673 ± 0.099	0.462 ± 0.014	0.506 ± 0.020	0.510 ± 0.008
	PRC	0.034 ± 0.050	0.007 ± 0.005	0.006 ± 0.011	0.002 ± 0.000	0.002 ± 0.000	0.002 ± 0.000

Table 4.7: ROC-AUC and PR-AUC Scores for ML Models on MUV MUV-852 Target. Values are mean values with standard deviation over 20 rounds of testing. Best values are highlighted in bold text. The first column shows the composition of the support set as explained in text.

models significantly, however, MNs obtain the highest ROC scores in one-shot learning. Despite this being the highest score, the ROC scores in general for this target are quite low. Figure 4.3 illustrates how the ROC scores don't have a strong predictive capacity. PRC scores for this target are very low and hence we do report the Mann-Whitney U rank results for these values. The Mann-Whitney U Rank test is reported for the closest performer below.

- **10+/10-**. **Best ROC - RF/GCN. Best PRC - RF/GCN.** GCNs obtain the highest ROC scores, however, there is no significant difference between the ROC values obtained through RFs ($U=257$ and $p=0.126$).
- **5+/10-**. **Best ROC - RF/GCN. Best PRC - RF.** GCNs obtain the highest ROC scores, however, there is no significant difference between the ROC values obtained through RFs ($U=227$ and $p=0.473$).
- **1+/10-**. **Best ROC - RF/GCN. Best PRC - RF.** GCNs obtain the highest ROC scores, however, there is no significant difference between the ROC values obtained through RFs ($U=232$ and $p=0.394$).
- **1+/5-**. **Best ROC - RF/GCN. Best PRC - RF/GCN.** GCNs obtain the highest ROC scores, however, there is no significant difference between the ROC values obtained through RFs ($U=220$ and $p=0.598$).

- **1+/1-. Best ROC - MN/GCN. Best PRC - RF/GCN/MN/PN.** MNs obtain the highest ROC scores, and upon closer investigation, we notice that there is no significant difference between the values obtained from GCNs ($U=266$ and $p=0.076$). However, upon performing a one-sided Mann-Whitney U rank test, the null hypothesis that the ROC values from MN are stochastically greater than those from GCN ($U=169$ and $p=0.038$) is accepted.

MUV MUV-858	Metric	RF	Graph Conv	SiameseNet	MatchingNet	ProtoNet	RelationNet
10+/10-	ROC	0.609 ± 0.079	0.644 ± 0.071	0.533 ± 0.031	0.545 ± 0.057	0.539 ± 0.041	0.489 ± 0.052
	PRC	0.004 ± 0.003	0.003 ± 0.001	0.001 ± 0.000	0.002 ± 0.000	0.002 ± 0.000	0.001 ± 0.000
5+/10-	ROC	0.573 ± 0.070	0.598 ± 0.054	0.529 ± 0.036	0.446 ± 0.036	0.562 ± 0.026	0.469 ± 0.026
	PRC	0.011 ± 0.013	0.003 ± 0.002	0.001 ± 0.000	0.002 ± 0.000	0.002 ± 0.000	0.002 ± 0.000
1+/10-	ROC	0.537 ± 0.058	0.556 ± 0.052	0.527 ± 0.162	0.405 ± 0.009	0.488 ± 0.014	0.477 ± 0.008
	PRC	0.005 ± 0.008	0.003 ± 0.001	0.000 ± 0.000	0.002 ± 0.000	0.002 ± 0.000	0.002 ± 0.000
1+/5-	ROC	0.534 ± 0.065	0.550 ± 0.071	0.514 ± 0.140	0.440 ± 0.016	0.482 ± 0.014	0.500 ± 0.000
	PRC	0.005 ± 0.008	0.003 ± 0.002	0.001 ± 0.001	0.002 ± 0.000	0.002 ± 0.000	0.002 ± 0.000
1+/1-	ROC	0.521 ± 0.065	0.505 ± 0.069	0.558 ± 0.103	0.554 ± 0.017	0.526 ± 0.014	0.437 ± 0.014
	PRC	0.002 ± 0.001	0.003 ± 0.001	0.002 ± 0.001	0.002 ± 0.000	0.002 ± 0.000	0.002 ± 0.000

Table 4.8: ROC-AUC and PR-AUC Scores for ML Models on MUV MUV-858 Target. Values are mean values with standard deviation over 20 rounds of testing. Best values are highlighted in bold text. The first column shows the composition of the support set as explained in text.

MUV-859 - 24 Actives - 14,722 Decoys

Table 4.9 contains the results for the MUV-859 target. Results are once again not convincing and the predictive performance of the classifier, based on ROC and PRC is low. PRC scores for this target are extremely low, while ROC scores show that the classifier is close to having no predictive skill. Hence, we do not report the Mann-Whitney U rank results for the results from this target. Unlike previous targets, we also do not provide a discussion on separate support set composition experiments as all results are close to random performance.

4.3.2.3 | Few-Shot Learning on DUD-E GPCR Subset

For the DUD-E dataset, we make use of the GPCR subset, which contains five targets. The imbalance of the data is significant in this dataset too, however, not as pronounced

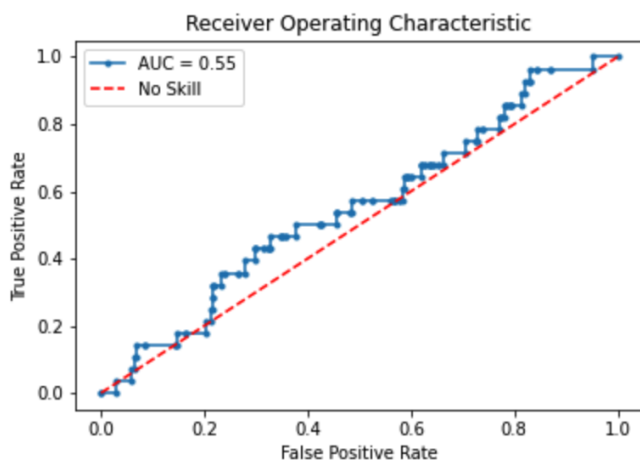


Figure 4.3: ROC Scores for Matching Networks for target MUV-858, showing how this model does not have a strong predictive capacity.

MUV MUV-859	Metric	RF	Graph Conv	SiameseNet	MatchingNet	ProtoNet	RelationNet
10+/10-	ROC	0.539 ± 0.077	0.517 ± 0.057	0.499 ± 0.018	0.516 ± 0.042	0.513 ± 0.041	0.398 ± 0.053
	PRC	0.003 ± 0.005	0.001 ± 0.000	0.001 ± 0.000	0.001 ± 0.000	0.001 ± 0.000	0.001 ± 0.000
5+/10-	ROC	0.538 ± 0.066	0.515 ± 0.067	0.492 ± 0.020	0.464 ± 0.023	0.510 ± 0.024	0.481 ± 0.052
	PRC	0.002 ± 0.001	0.002 ± 0.001	0.000 ± 0.000	0.001 ± 0.000	0.001 ± 0.000	0.005 ± 0.004
1+/10-	ROC	0.517 ± 0.081	0.504 ± 0.060	0.539 ± 0.109	0.446 ± 0.015	0.522 ± 0.011	0.427 ± 0.014
	PRC	0.002 ± 0.001	0.002 ± 0.000	0.000 ± 0.000	0.001 ± 0.000	0.002 ± 0.000	0.001 ± 0.000
1+/5-	ROC	0.524 ± 0.064	0.527 ± 0.057	0.426 ± 0.121	0.480 ± 0.013	0.453 ± 0.013	0.500 ± 0.000
	PRC	0.002 ± 0.001	0.002 ± 0.002	0.000 ± 0.000	0.002 ± 0.000	0.001 ± 0.000	0.002 ± 0.000
1+/1-	ROC	0.517 ± 0.064	0.499 ± 0.061	0.455 ± 0.072	0.504 ± 0.016	0.522 ± 0.009	0.406 ± 0.014
	PRC	0.002 ± 0.001	0.002 ± 0.001	0.001 ± 0.000	0.002 ± 0.000	0.002 ± 0.000	0.001 ± 0.000

Table 4.9: ROC-AUC and PR-AUC Scores for ML Models on MUV MUV-859 Target. Values are mean values with standard deviation over 20 rounds of testing. Best values are highlighted in bold text. The first column shows the composition of the support set as explained in text.

as in the MUV dataset. Two targets are reserved for testing, in which ADRB2 contains decoys that are auto-generated against a set of known active ligands, while for the CXCR4 target these are hand-picked. The rest of the targets (refer back to Chapter 3 Section 3.2 Table 3.3) are used for training.

For the ADRB2 target, the few-shot learning models achieve stellar performance based on ROC and PRC scores (see Table 4.10). The results are close to a perfect classifier,

which raises concerns about the underlying data. Our hypothesis is that the underlying data contains an inherent bias, which is confirmed by further research on the matter. Some studies indicate that the DUD-E dataset has limited chemical space and bias from the decoy compound selection process (Smusz et al., 2013; Wallach and Heifets, 2018). Chen et al. (2019) investigate this further to establish the effect these characteristics have on CNN models. The authors conclude that there is analogue bias within the set of actives within the targets (intra-target analogue bias), and also between the actives of different targets (inter-target analogue bias). They also provide evidence that there is also bias in decoy selection through the selection criteria for decoys.

DUD-E GPCR adrb2	Metric	RF	Graph Conv	SiameseNet	MatchingNet	ProtoNet	RelationNet
10+/10-	ROC	0.969 ± 0.018	0.914 ± 0.038	0.998 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	0.500 ± 0.000
	PRC	0.806 ± 0.103	0.311 ± 0.099	0.963 ± 0.014	0.986 ± 0.002	0.997 ± 0.001	0.015 ± 0.000
5+/10-	ROC	0.948 ± 0.022	0.875 ± 0.043	0.998 ± 0.005	0.996 ± 0.000	1.000 ± 0.000	0.999 ± 0.000
	PRC	0.729 ± 0.088	0.230 ± 0.102	0.959 ± 0.047	0.982 ± 0.003	0.996 ± 0.001	0.970 ± 0.003
1+/10-	ROC	0.842 ± 0.055	0.748 ± 0.072	0.998 ± 0.001	0.999 ± 0.000	0.998 ± 0.000	0.998 ± 0.000
	PRC	0.339 ± 0.135	0.099 ± 0.056	0.909 ± 0.117	0.975 ± 0.002	0.966 ± 0.009	0.974 ± 0.000
1+/5-	ROC	0.842 ± 0.088	0.744 ± 0.080	0.998 ± 0.000	0.995 ± 0.000	0.996 ± 0.000	0.997 ± 0.000
	PRC	0.339 ± 0.142	0.095 ± 0.059	0.913 ± 0.131	0.953 ± 0.005	0.993 ± 0.001	0.917 ± 0.005
1+/1-	ROC	0.776 ± 0.122	0.731 ± 0.088	0.998 ± 0.001	0.996 ± 0.000	0.997 ± 0.000	0.998 ± 0.000
	PRC	0.227 ± 0.180	0.058 ± 0.032	0.909 ± 0.052	0.981 ± 0.003	0.982 ± 0.006	0.964 ± 0.002

Table 4.10: ROC-AUC and PR-AUC Scores for ML Models on DUD-E GPCR adrb2 Target. Values are mean values with standard deviation over 20 rounds of testing. Best values are highlighted in bold text. The first column shows the composition of the support set as explained in text.

On the other hand, for the hand-picked decoys for the CXCR4 target, the RF model excels and outperforms the few-shot learning models (see Table 4.11). The benchmark models are not trained on other targets, but are instead only trained on a small support set from the same target. As expected, performance decreases with a decrease in support set size. However, the performance of the RF model still outperforms that of few-shot learning models. Seeing that the GCN benchmark model also performed significantly better than few-shot learning models, this implies that there is a clear benefit of training on the same data from the target, as opposed to the few-shot learning models which are trained on other targets instead. The fantastic performance obtained from training on such a small dataset raises questions about the biases within the data as explained previously.

DUD-E GPCR cxcr4	Metric	RF	Graph Conv	SiameseNet	MatchingNet	ProtoNet	RelationNet
10+/10-	ROC	0.994 ± 0.005	0.965 ± 0.020	0.570 ± 0.028	0.801 ± 0.030	0.631 ± 0.045	0.500 ± 0.000
	PRC	0.938 ± 0.043	0.697 ± 0.134	0.014 ± 0.002	0.085 ± 0.022	0.107 ± 0.020	0.009 ± 0.000
5+/10-	ROC	0.968 ± 0.019	0.927 ± 0.062	0.525 ± 0.038	0.693 ± 0.025	0.685 ± 0.130	0.701 ± 0.018
	PRC	0.794 ± 0.141	0.626 ± 0.188	0.031 ± 0.012	0.029 ± 0.003	0.122 ± 0.054	0.077 ± 0.006
1+/10-	ROC	0.866 ± 0.086	0.828 ± 0.108	0.520 ± 0.087	0.764 ± 0.021	0.684 ± 0.032	0.734 ± 0.018
	PRC	0.380 ± 0.140	0.361 ± 0.162	0.039 ± 0.066	0.066 ± 0.027	0.041 ± 0.009	0.108 ± 0.009
1+/5-	ROC	0.874 ± 0.081	0.781 ± 0.094	0.520 ± 0.084	0.707 ± 0.082	0.590 ± 0.087	0.700 ± 0.007
	PRC	0.417 ± 0.092	0.347 ± 0.111	0.050 ± 0.082	0.079 ± 0.037	0.045 ± 0.017	0.063 ± 0.005
1+/1-	ROC	0.833 ± 0.088	0.689 ± 0.150	0.545 ± 0.040	0.593 ± 0.034	0.732 ± 0.022	0.496 ± 0.010
	PRC	0.375 ± 0.124	0.174 ± 0.151	0.091 ± 0.102	0.041 ± 0.008	0.104 ± 0.012	0.035 ± 0.005

Table 4.11: ROC-AUC and PR-AUC Scores for ML Models on DUD-E GPCR cxcr4 Target. Values are mean values with standard deviation over 20 rounds of testing. Best values are highlighted in bold text. The first column shows the composition of the support set as explained in text.

4.3.3 | ECFP vs GCN Learned Embeddings on Tox21

In line with our objectives, we also ran an experiment to test whether the molecular representation affects the performance in few-shot learning. These experiments were run on the Tox21 dataset, using Prototypical Networks as these performed consistently well in the main experiments. ECFPs are based on the topology and a number of atom descriptors, in which the molecule is fragmented into local neighbourhoods and hashed into a vector. On the other hand, graph-learned embeddings are guided by gradient descent during training to produce a more relevant latent space embedding for the molecule. ECFPs are processed through a neural network with three layers as outlined in Chapter 3 Section 3.4.4. The neural network was used to learn a differentiable molecular embedding of the same size (a vector of size 128) as the one produced by the GCN. The same TanH activation function is used as the final non-linearity function in both neural networks. The results obtained using an embedding generated through GCNs outperform the ones in which an ECFP with a neural network was used. The Mann-Whitney U rank test is used to determine whether the ROC and PRC scores obtained through a GCN are statistically different. Table 4.12 contains results for the SR-HSE, SR-MMP, and SR-p53 targets. This table also contains the p-value in the two-sided Mann-Whitney U Rank test. As the p-values are less than 0.05, we reject the null hypothesis that the ROC and PRC distributions between results obtained from Prototypical Networks trained with embeddings generated through ECFPs versus training on

embeddings generated through GCNs, are statistically similar. All results show a clear advantage of using embeddings generated through GCNs, with one exception in PRC scores for the 1+/5- support set experiment on the SR-p53 target.

4.4 | Machine Learning Models Training Run Times

Training times vary between machine learning models. Figure 4.4 shows the training times for all machine learning models on the Tox21 dataset for the 10+/10- support set. This is representative of the rest of the datasets as the training time just varies accordingly with the size of the dataset. The RF and GCN benchmark model run very efficiently on the hardware utilised, which is outlined in Chapter 3 Section 3.8 Table 3.12. This is followed by PNs and RNs, which take on average the same amount of time to run, and then by SNs, which take longer to run than the former two networks. The network which takes the most time to run is the Matching Networks, which takes much longer than all other networks when run on the same hardware. Comparable results can be obtained using a network such as the Prototypical Networks, which takes a fraction of the time to train.

4.5 | Discussion

In line with the main aim of this study, we explore the effectiveness of few-shot learning algorithms for virtual screening through four few-shot techniques, namely, Siamese Networks, Matching Networks, Prototypical Networks, and Relation Nets. The attention based bi-directional LSTM, coined as the iterative refinement LSTM (IterRefLSTM), developed by Altae-Tran et al. (2017) is reproduced and applied to all embeddings generated through GCNs in our experiments. The choice of datasets were the Tox21, MUV and the GPCR subset of the DUD-E dataset, on which we applied two baseline machine learning models in addition to the few-shot learning models. RFs and GCNs are used as a benchmark test, in which a support set is sampled from the test target and a model is trained using only these few data points. This technique differs from few-shot learning training, as in the latter, we train over a set of molecular scaffolds reserved for training. Training consists of a set of episodes that replicate the conditions at test time, in which an N -way K -shot support set is sampled and tested against a set of queries. Our problem formulation was limited to a binary classification task so we always assume a value of two for N as we are trying to predict whether a molecule is active or not, or whether it is an inactive or decoy, depending on the dataset. Our choice of the K for the number

Tox21 Target	Support Set	Metric	ECFP	Graphs	p-value
SR-HSE	10+/10-	ROC	0.751 ± 0.017	0.782 ± 0.017	0.000
		PRC	0.227 ± 0.042	0.271 ± 0.055	0.004
SR-HSE	5+/10-	ROC	0.741 ± 0.016	0.772 ± 0.013	0.000
		PRC	0.202 ± 0.045	0.278 ± 0.045	0.000
SR-HSE	1+/10-	ROC	0.665 ± 0.156	0.780 ± 0.003	0.000
		PRC	0.148 ± 0.062	0.283 ± 0.016	0.000
SR-HSE	1+/5-	ROC	0.715 ± 0.102	0.773 ± 0.005	0.000
		PRC	0.181 ± 0.060	0.270 ± 0.018	0.000
SR-HSE	1+/1-	ROC	0.641 ± 0.168	0.779 ± 0.007	0.000
		PRC	0.163 ± 0.081	0.266 ± 0.029	0.000
SR-MMP	10+/10-	ROC	0.793 ± 0.022	0.845 ± 0.015	0.000
		PRC	0.477 ± 0.044	0.510 ± 0.032	0.015
SR-MMP	5+/10-	ROC	0.765 ± 0.046	0.846 ± 0.014	0.000
		PRC	0.425 ± 0.049	0.527 ± 0.029	0.000
SR-MMP	1+/10-	ROC	0.763 ± 0.080	0.849 ± 0.005	0.000
		PRC	0.444 ± 0.109	0.508 ± 0.025	0.021
SR-MMP	1+/5-	ROC	0.736 ± 0.156	0.846 ± 0.005	0.000
		PRC	0.445 ± 0.121	0.508 ± 0.025	0.002
SR-MMP	1+/1-	ROC	0.647 ± 0.219	0.847 ± 0.003	0.000
		PRC	0.355 ± 0.151	0.506 ± 0.018	0.000
SR-p53	10+/10-	ROC	0.793 ± 0.013	0.850 ± 0.004	0.000
		PRC	0.309 ± 0.051	0.369 ± 0.027	0.000
SR-p53	5+/10-	ROC	0.782 ± 0.022	0.852 ± 0.009	0.000
		PRC	0.278 ± 0.068	0.358 ± 0.042	0.000
SR-p53	1+/10-	ROC	0.780 ± 0.087	0.848 ± 0.005	0.000
		PRC	0.279 ± 0.086	0.361 ± 0.030	0.000
SR-p53	1+/5-	ROC	0.770 ± 0.036	0.840 ± 0.005	0.000
		PRC	0.273 ± 0.087	0.310 ± 0.023	0.323
SR-p53	1+/1-	ROC	0.753 ± 0.116	0.834 ± 0.003	0.000
		PRC	0.274 ± 0.089	0.346 ± 0.021	0.001

Table 4.12: ROC-AUC and PR-AUC Scores for Prototypical Networks on Tox21 with ECFP or GCN embeddings. Values are mean values with standard deviation over 20 runs. Best values are highlighted in bold text. A Mann-Whitney U Rank p-value less than 0.05 indicates significant difference of the GCN Embeddings over ECFPs.

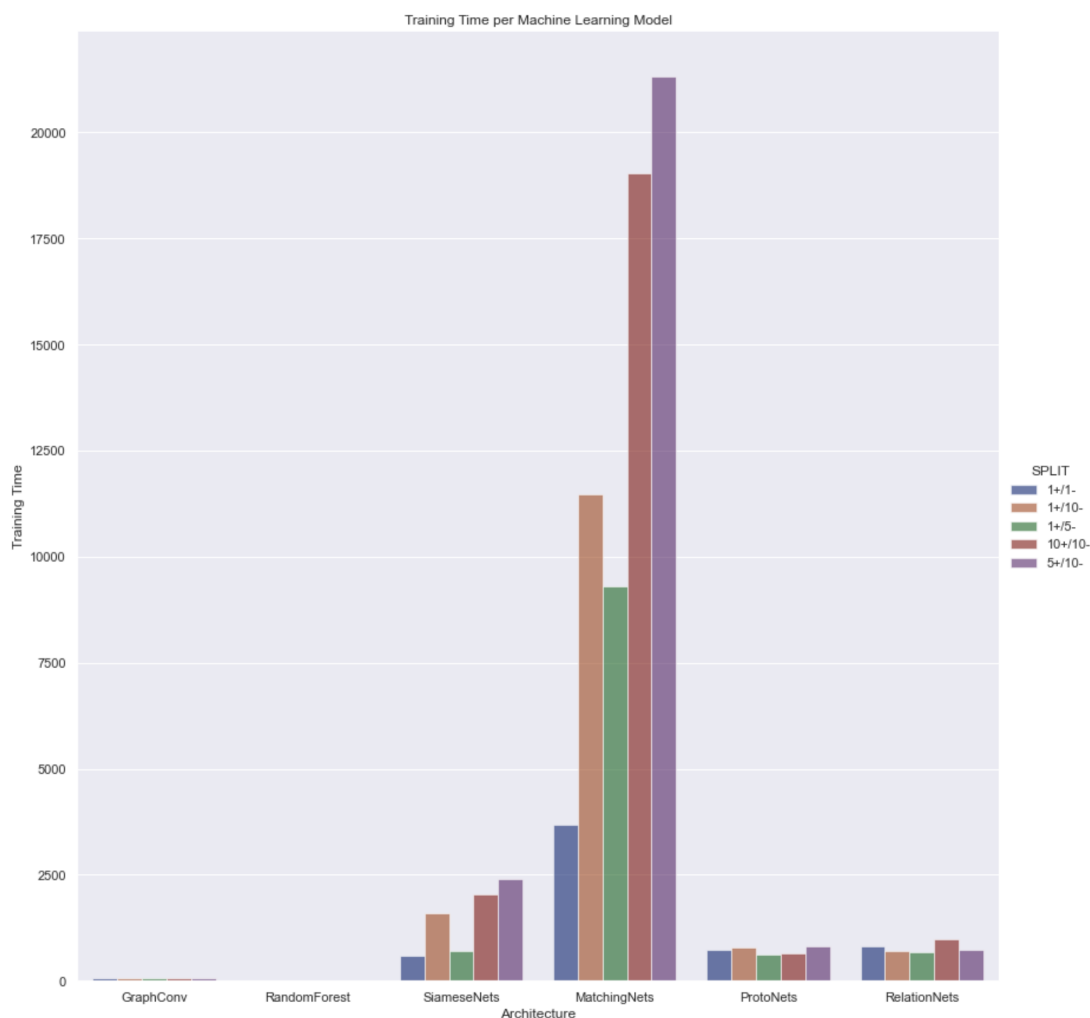


Figure 4.4: Training Times in seconds for the Machine Learning Models on the Tox21 Dataset for the 10+/10- support set experiments.

of examples per class that make up the support set follows the methodology of Altae-Tran et al. (2017), taking into consideration 10+/10-, 5+/10-, 1+/10-, 1+/5- and 1+/1- support sets. Interestingly, Altae-Tran et al. (2017) experiment with imbalanced support sets where the number of actives in the support set is not always equal to the number of the negative class. Therefore, the choice of support sets in our experiments ranges from few-shot learning to one-shot learning, where the latter only uses one example per class to train and test the machine learning model.

Tox21. When testing the machine learning models on the Tox21 dataset, the few-shot learning models outperform the baseline models. This performance is in line with that reported by Altae-Tran et al. (2017). In Table 4.13, we present the best results obtained

by their work for each Tox21 target. In the same table, we report our highest ROC score and tabulate the best identified few-shot learning models. These models are identified by comparing the highest obtained scores from 20 rounds of testing, with those obtained by other few-shot learning models using the Mann-Whitney U rank test. In line with the established state of the art, the MN with IterRefLSTM perform well and obtain the best ROC results in a number of experiments. The fact that the same implementation for the Matching Networks obtained better results than the state of the art work, can be attributed to the set of atom descriptors used for the initial graph representations. Our few-shot learning architecture implementation is identical to their work, but different hyperparameters in the implementation which were not clear in the original work could also contribute to variations in results. Hence, we focus mainly on the performance of how our implementations performed against each other. Our implementation of the state of the art, labelled as MN for Matching Networks, have the edge on the SR-MMP target, performing comparably well in 10-shot learning and the 5+/10- support set to PN and RN in the former and to PN in the latter. They obtain the best ROC performance on the 1+/5- support set experiments and in one-shot learning for the SR-MMP target. Meanwhile, performance is mixed for the SR-HSE ROC scores, as MN obtained the best ROC results for the 10+/10- and 1+/5- support set experiments, RNs obtained the best results on the 5+/10- support sets and finally PNs obtained the best results in one-shot learning. For the SR-p53 target, PNs outperform other models consistently, except for one-shot learning in which MNs take the advantage.

Results for one-shot learning are mixed between our implementations for MNs and PNs with the IterRefLSTM. They both achieve comparable performance on Tox21 targets for one-shot learning. The performance of MNs for this scenario is consistent with the state of the art work. Based on the underlying theories of both architectures, the results are consistent with our expectations. In a one-shot learning scenario, MNs and PNs are conceptually similar. The *prototypes* in PNs are a mean of all embeddings for each class in the support set. The euclidean distance between the *prototypes* and each embedding from the query set is calculated to predict the activity of the query. As in one-shot learning we only have one example per class, the *prototypes* are equivalent to the embedding for each class, making this identical to MNs. The difference in our implementation is that for MNs we use the cosine distance, while for PNs, we make use of the euclidean distance, as proposed in the literature which introduced these two techniques.

We remind the reader that while we also report the PR-AUC score from our experiments, this metric is not available in the study by Altae-Tran et al. (2017). For the PRC metrics, PNs consistently performed well, obtaining the best PRC scores throughout all Tox21 targets. Using statistical analysis, MNs and RNs also match the performance

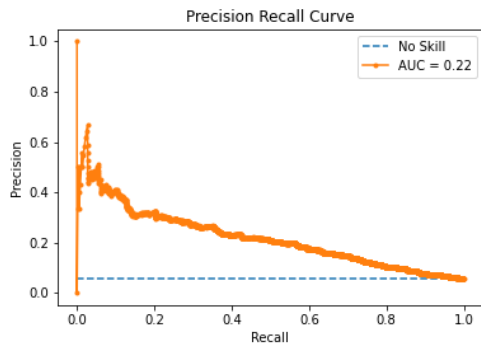


Figure 4.5: PRC plot for Tox21 SR-HSE - 10+/10- support set.

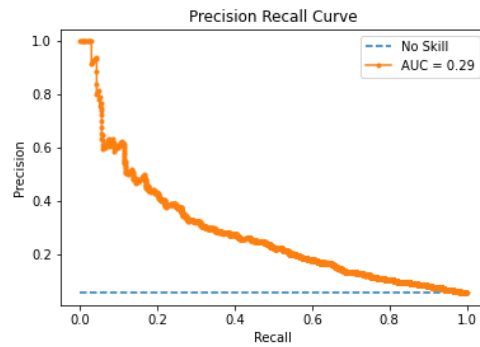


Figure 4.6: PRC plot for Tox21 SR-HSE - 1+/1- support set.

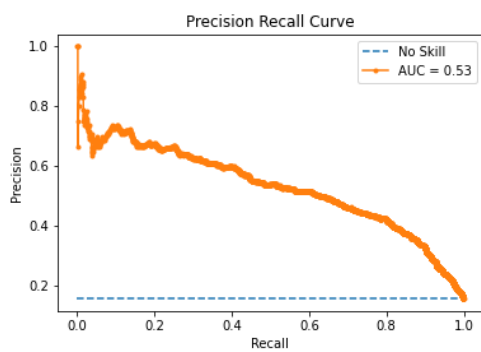


Figure 4.7: PRC plot for Tox21 SR-MMP - 10+/10- support set.

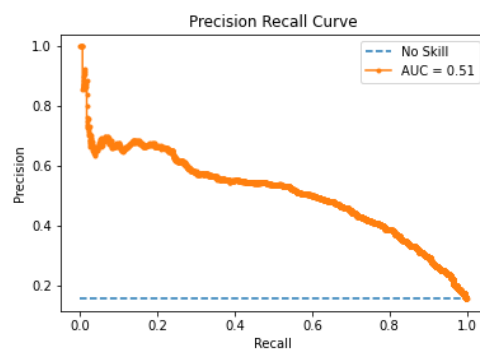


Figure 4.8: PRC plot for Tox21 SR-MMP - 1+/1- support set.

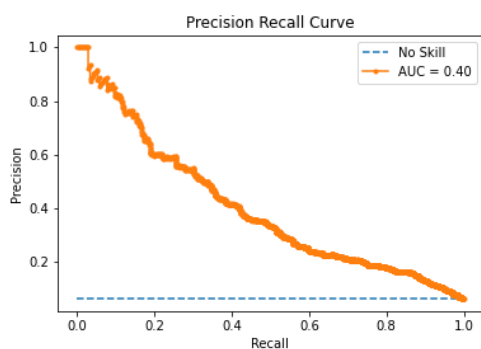


Figure 4.9: PRC plot for Tox21 SR-HSE - 10+/10- support set.

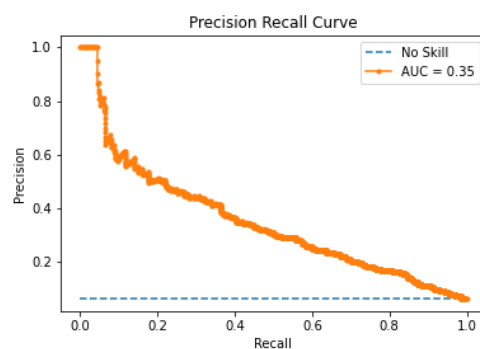


Figure 4.10: PRC plot for Tox21 SR-HSE - 1+/1- support set.

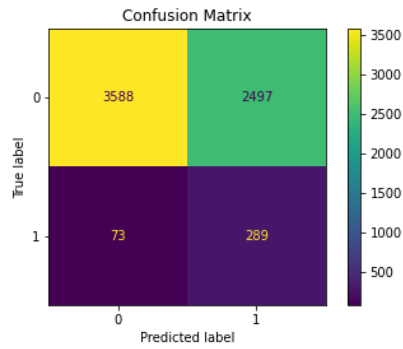


Figure 4.11: Confusion Matrix for Tox21 SR-HSE - 10+/10- support set.

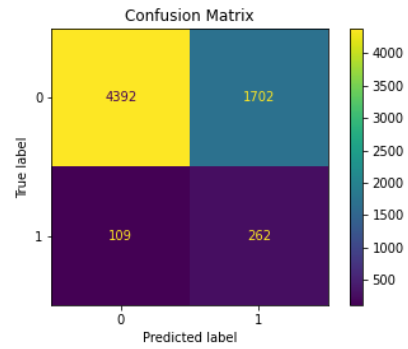


Figure 4.12: Confusion Matrix for Tox21 SR-HSE - 1+/1- support set.

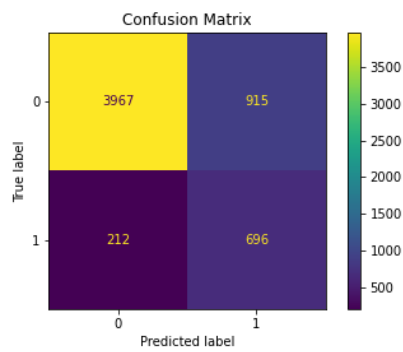


Figure 4.13: Confusion Matrix for Tox21 SR-MMP - 10+/10- support set.

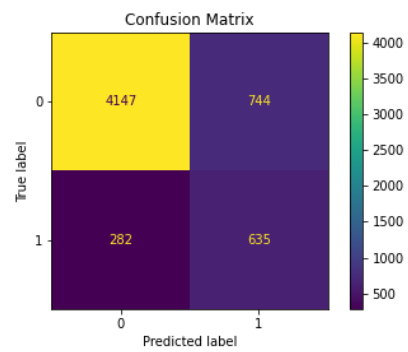


Figure 4.14: Confusion Matrix for Tox21 SR-MMP - 1+/1- support set.

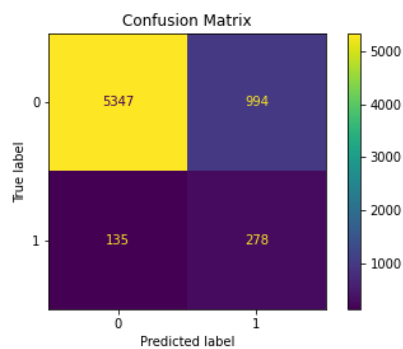


Figure 4.15: Confusion Matrix for Tox21 SR-HSE - 10+/10- support set.

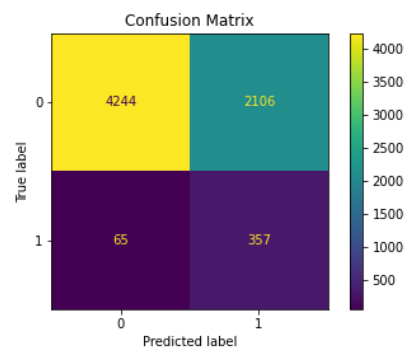


Figure 4.16: Confusion Matrix for Tox21 SR-HSE - 1+/1- support set.

Target	Support Set	SOTA	SOTA ROC	Best ROC	Best Networks
SR-HSE	10+/10-	MN	0.772 \pm 0.002	0.793 \pm 0.002	MN
SR-HSE	5+/10-	MN	0.771 \pm 0.002	0.791 \pm 0.003	RN
SR-HSE	1+/10-	MN	0.671 \pm 0.007	0.788 \pm 0.001	MN
SR-HSE	1+/5-	MN	0.729 \pm 0.003	0.789 \pm 0.001	RN
SR-HSE	1+/1-	MN	0.767 \pm 0.001	0.779 \pm 0.007	PN
SR-MMP	10+/10-	MN	0.838 \pm 0.001	0.845 \pm 0.015	MN/PN/RN
SR-MMP	5+/10-	MN	0.847 \pm 0.001	0.853 \pm 0.007	MN/PN
SR-MMP	1+/10-	SN	0.809 \pm 0.020	0.849 \pm 0.005	PN
SR-MMP	1+/5-	MN	0.799 \pm 0.002	0.853 \pm 0.001	MN
SR-MMP	1+/1-	MN	0.835 \pm 0.001	0.851 \pm 0.008	MN
SR-p53	10+/10-	MN	0.823 \pm 0.002	0.850 \pm 0.004	PN
SR-p53	5+/10-	MN	0.830 \pm 0.001	0.852 \pm 0.009	PN
SR-p53	1+/10-	SN	0.726 \pm 0.173	0.848 \pm 0.005	PN
SR-p53	1+/5-	MN	0.795 \pm 0.005	0.840 \pm 0.005	PN
SR-p53	1+/1-	MN	0.827 \pm 0.001	0.838 \pm 0.004	MN

Table 4.13: Comparison of our best ROC-AUC scores against the state of the art (SOTA) results from Altae-Tran et al. (2017) on the Tox21 dataset. The best networks reported are based on the statistical analysis carried out in Section 4.3.2.1. Values are mean values with standard deviation over 20 rounds of testing. Best values are highlighted in bold text.

in some cases, as reported in Section 4.3.2.1. The PRC is used to determine how well the model predicts active compounds, as it is the ratio of true positives divided by the sum of true positives and false positives. Therefore, we strongly believe that in machine learning experiments for virtual screening, this metric should be used in addition to ROC scores. Figures 4.5 - 4.10 show the PRC curves for the Tox21 targets for the 10+/10-support sets. The PRC curves show high precision and low recall, which is confirmed by the confusion matrices in Figures 4.11 - 4.16. This indicates that we have a good number of true positives, but due to the imbalance and nature of our data we have a low recall due to the number of false negatives predicted by the model. In this problem domain, the negative data points greatly exceed the positive examples. By observing the confusion matrices for 10-shot and 1-shot learning on PNs in Figures 4.11 - 4.16, we see that the model manages to predict a good proportion of the active class, indicating

good precision. As the PRC plots indicate, the confusion matrices also indicate a bigger number of false negatives when compared to true positives, which results in a low recall. Given the complexity and imbalance of the tasks at hand, the PRC scores obtained are acceptable.

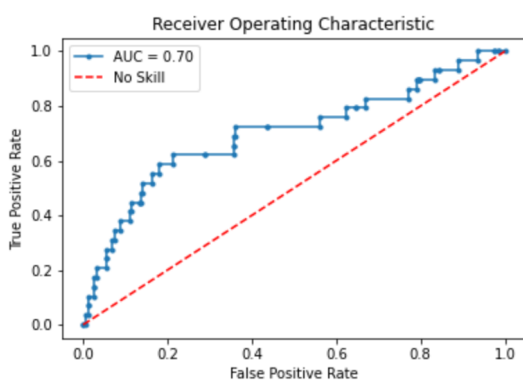


Figure 4.17: ROC plot for MUV-832, trained and tested with a 1+/10- support set composition. The plot is for the median ROC score obtained from 20 rounds of testing.

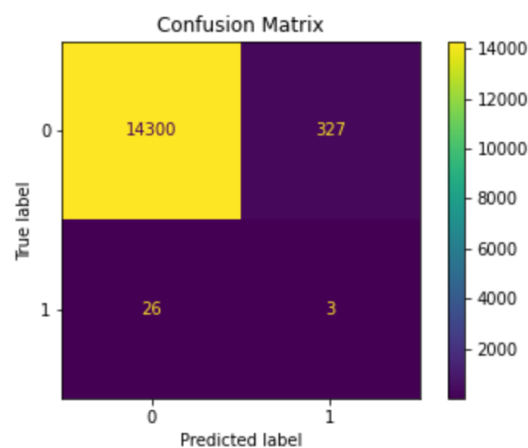


Figure 4.18: Confusion Matrix for MUV-832, trained and tested with a 1+/10- support set composition. The plot is for the median ROC score obtained from 20 rounds of testing.

MUV. Each active in the MUV dataset is structurally distinct from the other, making each data sample maximally informative. Therefore, structural similarities cannot be exploited on unseen active molecules. In fact, the dataset obtained the worst results, since the baseline benchmark tests consistently outperformed few-shot learning techniques. Altae-Tran et al. (2017) report that the results obtained through the GCNs baseline also struggle in performance, however, from our tests and statistical analysis we find that this is not the case for all MUV targets. For most targets, there is no significant difference between the scores obtained through the RFs and GCNs baselines. RNs obtain the best ROC scores in one instance on the MUV-832 target when trained with a 1+/10- support set, obtaining a mean ROC-AUC score of 0.683 ± 0.010 . However, this result is not consistent and the performance is only observed in this single instance. The confusion matrix in Figure 4.18 shows that predictive power for active compounds is still not reliable. The ROC curve for the median ROC score obtained over 20 rounds of testing is shown in Figure 4.17. Other than this rare instance, our results are consistent with the conclusion from Altae-Tran et al. (2017) that baseline machine learning outperforms

few-shot machine learning techniques on the MUV dataset. As the models for few-shot learning do not hold any predictive power, both in the state of the art and our experiments, we do not tabulate the best performing ROC scores with that of Altae-Tran et al. (2017).

DUD-E. Results obtained from the DUD-E dataset are not conclusive. The CXCR4 target returned results similar to the MUV targets, where the baseline models outperformed all few-shot learning ones. On the other hand, the few-shot machine learning models returned stellar performance on the ADRB2 target. However, as discussed previously in Section 4.3.2.3, this is evidence for bias within the data as reported by Chen et al. (2019). Having such mixed results on two different targets within the same subset of the dataset does not give us a conclusive picture of whether few-shot learning is effective on this dataset.

Training Times. From the results on the Tox21 dataset, MNs, PNs and RNs obtain good predictive performance, however, it is evident from the presented result that the two latter networks are much faster to train on the same hardware. From our experiments on the three Tox21 targets, MNs and PNs were the most consistent in results. As the decrease in training times is significant, by over 150% between MNs and both PNs and RNs, we believe that this puts the latter two networks at an advantage. Faster training times allow faster turnaround of results from datasets, while requiring less intense use of computer hardware. This increase in efficiency also allows scientists to perform a more rigorous hyperparameter search on various datasets in a shorter time.

ECFPs versus GCN learned embeddings. In their research paper, Altae-Tran et al. (2017) suggest that future work might investigate the performance of fingerprints with learned embeddings. This was one of the motivations for our objectives. The results presented in Section 4.3.3 clearly show that learned embeddings are advantageous for few-shot learning when compared to using ECFPs. This shows that augmenting molecular structures with features and then learning an embedding based on the problem domain adds meaningful information to the embeddings.

4.6 | Summary

In this chapter, we present and discuss the results from the designed experiments, which we introduce in Chapter 3 for the few-shot learning techniques on the Tox21, MUV and the GPCR subset of the DUD-E datasets. We first revisit the aims and objectives of our study before delving into the results obtained. The results obtained across the three different datasets do not provide a clear-cut conclusion as performance is highly depen-

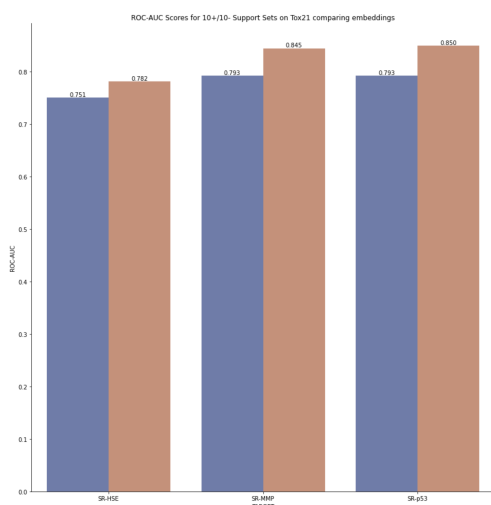


Figure 4.19: ROC-AUC scores when using ECFP versus GCN embeddings on Tox21 with Prototypical Networks trained with 10-shot support sets.

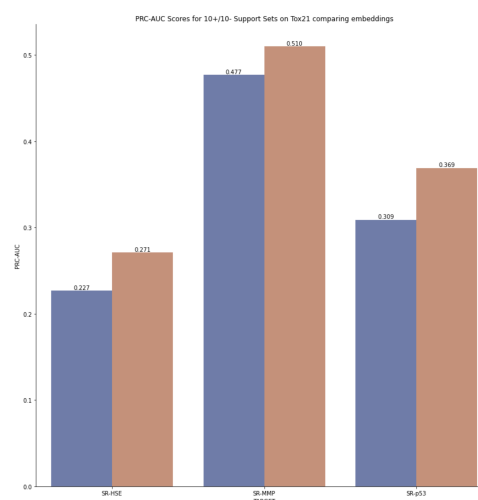


Figure 4.20: PR-AUC scores when using ECFP versus GCN embeddings on Tox21 with Prototypical Networks trained with 10-shot support sets.

dent on the nature of the data. On the Tox21 dataset, we conclude that the Prototypical Networks models obtain the best performance overall, with Matching Networks (the state of the art), being a close contender. Results on the MUV and DUD-E datasets are not conclusive as the few-shot learning models do not perform well and are outperformed by our baseline models. On one DUD-E target, performance is exceptional, but we note that this could be attributed to an inherent bias within the data. We also performed experiments to compare the performance of one of the best performing architectures, the Prototypical Networks, when using ECFP versus using learned embeddings through GCNs. The latter proved to be superior in all cases. Training times between models are also evaluated and we find that the proposed few-shot learning models, which obtain comparable performance to the state of art, take a fraction of the time to train. This improvement in training efficiency affords further experiments on Prototypical and Relation Networks when using the same hardware. The results obtained from our work are compared and evaluated with that of the state of the art work by Altae-Tran et al. (2017). We find that our findings are consistent with those from the state of the art, with Prototypical and Relation Networks performing better in a number of experiments on the Tox21 dataset.

Conclusions

Human beings exhibit a remarkable ability to quickly and effectively learn from exposure to just a few examples. After seeing a cat for the first time, a child can correctly identify cats upon future encounters. Conventional supervised deep learning techniques require a plethora of data to train a model. In order to bridge this gap between human learning and machine learning, the meta-learning domain aims to train a model to *learn how to learn*. By training on a small number of data points from a set of related, but not identical tasks, a machine learning model can be trained to generalise using only a few examples sampled from previously unseen tasks to classify new data points from these new tasks. This method of learning affords invaluable capabilities especially for areas in which data is not easily available, such as the drug discovery process.

The drug discovery process is a resource-intensive and time-consuming process. The whole process takes years to complete from a drug's original conception to launch. Additionally, data acquisition is not only expensive, but also difficult to obtain. The main goal in drug discovery is to find small molecules which activate or exhibit a therapeutic effect against biological targets. This study focuses on ligand based virtual screening (LBVS), which uses information from known ligands to identify new ones, effectively using known ligands as a "template". Datasets used for virtual screening are based around a number of experimental assays and supply a number of active and inactive or decoy compounds for a number of biological targets. For this study we explore a toxicology dataset, the Toxicology for the 21st Century (Tox21) dataset which is typically used for lead optimisation to identify toxicology effects of molecules. The Tox21 dataset is composed of active and inactive molecules against 12 biological targets. We also make use of two other datasets, the Maximum Unbiased Validation (MUV) and the G-protein-coupled receptors (GPCR) subset of the Database of Useful Decoys: Enhanced (DUD-E),

which are used for hit identification in virtual screening. These two datasets consist of actives and decoys against a number of biological targets. Decoys differ from inactives in that it is presumed that they are inactive against a biological target based on physical properties, and are not validated to be truly inactive. These datasets contain a significant imbalance between the number of actives and inactives/decoys, with the latter far exceeding the number of actives. As the number of actives is limited and difficult to obtain, it would be significantly advantageous if a machine learning model could generalise using merely a few examples.

Hence, in this study, we explore the application of few-shot machine learning techniques for lead optimisation and hit identification in virtual screening. Few-shot learning is a type of meta-learning technique and has been effectively applied in domains such as computer vision and natural language processing. One commonly used technique is metric-based few-shot machine learning, in which the model learns to embed inputs and computes a distance function over these embeddings to classify them. In chronological order, this has been accomplished through Siamese (Koch et al., 2015), matching (Vinyals et al., 2016), Prototypical (Snell et al., 2017), and Relation (Sung et al., 2018) Networks. These neural networks need to take in the molecules in the designated datasets as inputs. However molecules, which are provided in SMILES format, first need to be represented in computational space. Based on existing literature, we mainly focus on graph representations, but we also explore the use of extended-connectivity fingerprint (ECFP) molecular representations. Graphs are a natural representation for molecules as they are made up of a set of nodes and edges, which correspond to atoms and bonds in molecules. Atom descriptors are added to the nodes to add further information to the graph. These graph objects are processed using a graph convolutional network (GCN) to map the molecular graph to a learned embedding in latent space.

For effective few-shot machine learning, the conditions during training must match those at test time. Firstly, a subset of the targets within the dataset are reserved for training, and the rest for testing. Training the model is achieved by repeatedly emulating the conditions at test by sampling a small number of examples from a training target, referred to as the support set, and attempting to generalise for the activity of a number of sampled query molecules, referred to as the query set, using only the small support set. This is done through episodic learning, where the learning problem is framed as a series of **N-way, K-shot** tasks. N always assumes a value of two as our problem is a binary classification one, in which we are trying to predict whether a molecule is active or an inactive/decoy for a biological target. K is the value of examples sampled from every class in our dataset to make up the support set. For our few-shot learning methodology, K assumes a maximum value of 10, and a minimum of one example per

class. We also experiment with an imbalanced support set in which we sample more of the inactive/decoy class than the active class. This support set emulates the conditions at test time. During test time, we sample the support set randomly from a previously unseen target, and the trained model uses this new information to generalise for the rest of the query molecules in this new target.

Altae-Tran et al. (2017) explore the application of few-shot learning for the drug discovery domain. The authors develop the aforementioned Matching Networks further by expanding on concepts such as attention and bi-directional long-short term memory (LSTM) networks to propose the Iterative Refinement LSTM (IterRefLSTM). Their work preserves the context-awareness of Matching Networks, but eliminates order dependencies and embeds the support and query sets simultaneously using information from both sets. For the purposes of this study and to the best of our knowledge, their work is the state of the art in this field at the time of writing. In this study, we build on their work by first reproducing their work and results, and then also applying Prototypical Networks and Relation Networks, which, to the best of our knowledge, have previously been unexplored for this problem domain. We augment these networks through the use of IterRefLSTMs from Altae-Tran et al. (2017), in order to objectively compare and contrast results. We reiterate from Altae-Tran et al. (2017) that this is not merely an application of past work on one-shot learning to molecular data. One application of few-shot learning to the computer vision domain is to explore object recognition in computer vision for new image classes when provided with only a few data points for each class. The equivalent learning challenge would be to generalise for the activity of compounds in a new molecular scaffold using only a few data points, for a fixed experimental assay. In our study, the machine learning techniques proposed should be able to train a model that can generalise the activity of molecules for new experimental assays, which are related, but not the same as the experimental assays used for training.

5.1 | Revisiting this Study's Aims and Objectives

The main aim of this study is to explore the efficacy of few-shot learning for low-data LBVS, which was guided by two research questions, namely:

1. Since we have limited active data, a model that can generalise using only a few examples is highly advantageous. Therefore, are few-shot machine learning techniques effective for low-data ligand-based virtual screening?
2. Do Prototypical (Snell et al., 2017) and Relation Networks (Sung et al., 2018) af-

ford better performance for ligand-based virtual screening than the Matching Networks (Vinyals et al., 2016) component in the established state of the art (Altae-Tran et al., 2017)?

The work of Altae-Tran et al. (2017), established to be the state of the art for this problem domain, served as the foundation for our study. After a thorough review of few-shot machine learning techniques and setting up two conventional machine learning pipelines as a baseline, we reproduced the state of the art by implementing the proposed IterRefLSTM, which expands on Matching Networks. Our experiments are run on three datasets, namely the Tox21, MUV and the GPCR subset of the DUD-E dataset. Besides from reproducing the models and results in the state of the art, we also report Precision-Recall Area under the Curve (PR-AUC) scores in addition to the Receiver-Operator Characteristic (ROC) scores, aiming to add further insight into the performance of few-shot learning models. We strongly believe that the introduction of PRC is crucial in this problem domain due to the high imbalance inherent to the datasets used, and the importance of correctly identifying active compounds. Furthermore, we also apply Prototypical and Relation Networks, previously explored architectures for computer vision, and which to the best of our knowledge have not been applied to this problem domain before. Besides the standard implementation of these two networks, the support and query learned embeddings from the GCNs are also processed using the IterRefLSTM component, which is reproduced from the state of the art.

Our implementation of the work by Altae-Tran et al. (2017) matched their performance based on the provided ROC scores in their study. However, the results for the chosen datasets do not provide us with a clear-cut answer to our first research question. Our results are consistent with the state of the art results, where we see few-shot learning models able to generalise and predict molecular activity against new unseen targets in the Tox21 data using only a small support set. Results from few-shot learning models on Tox21 data consistently perform better than our baseline machine learning approaches. The same generalising capabilities is not achieved on MUV data due to the nature of the data available within this dataset. The results on the DUD-E data does not give a clear indication of performance, and the excellent results obtained on one DUD-E target raises questions about hidden bias within the data. Therefore, we conclude that few-shot machine learning is effective for low-data ligand-based virtual screening depending on the nature of the data used. For data such as MUV, in which active compounds per target are scarce and each compound is structurally distinct from all others, the few-shot learning models struggle to generalise well.

For our second research question, we find that on the Tox21 datasets, the Proto-

typical network is the best performing network, with much faster training times than our implementation of the Matching Networks. Prototypical Networks dominate all other networks in PRC scores, and also have a slight edge when comparing ROC scores compared to the state of the art. The state of the art and Prototypical Networks perform significantly better than our implementation of Relation Networks. Hence, we conclude that Prototypical Networks offer better generalising capabilities for few-shot learning in ligand-based virtual screening than the Matching Networks component in the state of the art.

To further explore the efficacy of few-shot machine learning, we also run experiments to investigate the performance of low-data learning with different molecular embeddings. This experiment was limited to comparing the performance of using ECFPs against learned embeddings from GCNs. Our results show that the latter is superior in all the tests we carry out.

5.2 | Critique and Limitations

Having carried out the main objectives set out for this study and answered our research questions, we now highlight some limitations in our work.

This study focused on few-shot learning architectures and therefore, we did not do a comprehensive analysis into how learned embeddings are created through GCNs. The architecture for the utilised GCN was motivated from Altae-Tran et al. (2017), and we did not carry out any further exploratory work into other implementations. For the purposes of this study, we did not attempt to perform a comprehensive implementation of meta-learning techniques for this problem domain. We focused our efforts around metric-based implementations to build on the state of the art contributions. In the next sub-section, we propose other fields of interest in the meta-learning domain such as optimisation-based techniques as opposed to metric-based ones. One difficulty we encountered was that the state of the art implementation was based on deprecated code. As a result, the implementations were reproduced so as to avoid unnecessary bias when evaluating with our other implementations. This might lead to having slight variations in the implementation itself, even though our results lead us to believe otherwise as they are consistent with those published in literature. Additionally, our work focused on implementing the machine-learning architectures for ligand-based data and we did not do an extensive hyper parameter search.

We report that Prototypical Networks obtain better performance than Relation Networks. Further exploration could go into the processing of the feature map concatena-

tions in Relation Networks to establish whether this could improve performance. To create feature map concatenations, the query embeddings are replicated and appended to the embeddings of the support set. These feature map concatenations are then further processed through a feed-forward neural network with dense layers. In the original paper, (Sung et al., 2018) further process the feature maps using convolutional neural networks, before processing through a feed-forward neural network. Hence, one could say that the equivalent implementation for graph data would be to process the feature maps using graph neural networks before processing through the final dense layers of the neural network. However, appending graphs together is not as trivial as appending two images; as the order of the nodes in both graphs is arbitrary and new edges are needed between the nodes of two individual graphs. Additionally, to obtain graph embeddings, we would have needed to change our GCN architecture to output a graph rather than a vectorised embedding, resulting in comparing the results of Relation Networks with other networks using a different GCN architecture. Hence, we opted for creating feature maps with the vectorised embeddings. Future work could possibly explore the feature map concatenations of two molecular graphs.

5.3 | Future Work

In this study we explored the application of metric-based few-shot machine learning techniques, mainly with GCNs, and augmented with the IterRefLSTM (Altae-Tran et al., 2017) to improve the support and query embeddings. For this section, we start at the start of our machine learning pipeline and propose ideas for further exploratory work.

Our machine learning pipeline starts from the conversion of SMILES molecular representations to a graph structure with nodes and edges representing atoms and bonds respectively. In our study, atoms are featurised with atomic properties outlined in Chapter 3. More atomic properties such as chirality and atom mass could be added to the graph data structures. Additionally, bond properties, such as bond type and conjugation, were not utilised for the purposes of this study. These additions could augment the embeddings created from the GCNs by possibly adding more valuable information.

For the purposes of our study, we have focused mainly on the few-shot learning architectures for the learned embeddings. As the exploration of the GCN architecture itself was out of the scope of this study, we only carried out experiments with one GCN architecture, based on the work of Altae-Tran et al. (2017). Further work could go into experimenting with different GCN architectures such as GINs (Xu et al., 2018), Graph-

SAGE (Hamilton et al., 2017) or Relational-GCNs (Schlichtkrull et al., 2018). The GCNs create the embeddings that we use further down the pipeline. These embeddings are processed with the IterRefLSTM and then ultimately through the few-shot learning architecture. Hence, the information that the learned embeddings contain is valuable for the few-shot learning paradigm. Our experiments have shown an improvement from using learned embeddings over ECFPs, highlighting the importance of the embedding used for the techniques employed. Another interesting area of research for consideration is the inclusion of 3D structural information in the molecular embedding. So far, we have only considered 2D molecular graphs, encoding only the atom features in a GCN. Stärk et al. (2021) propose 3D Infomax, which pre-trains a graph neural network to encode 3D structural information in the latent vector representations. Once this process is complete, the pre-trained GNNs can be transferred and used in datasets which do not contain 3D information to produce latent 3D information from 2D molecular graphs.

Vinyals et al. (2016), in their work proposing Matching Networks, have highlighted the fact that conditions during training and testing must match. What this effectively means is that the task formulation, *N-way K-shot* during training should match the support set composition that will be available during inference time. To the best of our knowledge, this has not been explored for molecular data and a more in depth investigation will lead to further insight into the way the data available is organised for training. Additionally, to our knowledge, no study has been done about the amount of data on different targets needed to build a well-performing model.

In this study we focused mainly on metric-based learning augmented with LSTMs. The field of meta-learning is extensive and has gained a lot of momentum over the past years. Finn et al. (2017) proposed an optimisation-based model-agnostic meta-learning (MAML) technique that performs better than Matching Networks in their reported results. Nichol and Schulman (2018) propose Reptile, another meta-learning optimisation algorithm that is model-agnostic, similar to MAML. The goal of these techniques is to find the optimal initialisation of weights so that training on similar tasks converges through gradient descent as fast as possible. Munkhdalai and Yu (2017) also propose Meta-Networks, which achieves better performance than Matching Networks on the Omniglot dataset. Meta-Networks are model-based meta-learners whose goal is rapid generalisation by updating weights through a few training steps.

For the drug-discovery domain, Guo et al. (2021) recently proposed the Meta-MGNN, a meta-learning framework for model optimisation, designed for molecular graph neural networks to address the low-data requirement of training a model to predict the property of a molecule. The authors report that they obtain better results than MAML and other methods. Additionally, Wang et al. (2021) build further on Meta-MGNN and

propose the Property-Aware Relation Networks (PAR) to handle few-shot learning in molecular property prediction. We note that this is different from the Relation Networks we explored in this study. PAR attempts to propagate information across similar molecules and the authors report better results than the work of Altae-Tran et al. (2017). However, in the PAR study, the authors utilise bond information, and use a different graph convolution network based on the work of Xu et al. (2018). Therefore, the results are not directly comparable and further work is needed as they do not reproduce the work of Altae-Tran et al. (2017) and thus it is difficult to quantify whether the improvement in results is due to the technique employed in PAR or other aforementioned factors.

5.4 | Final Remarks

In this study we explored how a machine learning model can *learn how to learn* and generalise using only a few examples. This research project builds on the work from Altae-Tran et al. (2017), who have set important foundations for this problem domain. Their work has been one of our principal starting points, and we developed their work further. First and foremost, we reproduce their work effectively and provide deeper insight into the study by introducing PRC reporting, over and above the ROC scores. Secondly, we also introduce two new few-shot machine learning models and explore their performance against the state of the art. The Prototypical and Relation Networks have been previously explored for the computer vision domain, but to our knowledge, have never been applied to the drug discovery domain. While our results vary across the datasets used, they are consistent with the work of Altae-Tran et al. (2017). The Prototypical Networks we introduce to this problem domain perform better on the Tox21 dataset based on ROC performance, while outperforming all other machine learning models in PRC performance. We believe that this is a valuable contribution as, in addition to obtaining better results than the state of the art, given the nature of the data used, the PRC provides more reliable insight into the performance of the models. We also find that making use of learned embeddings through GCNs, as opposed to ECFPs, consistently results in better ROC and PRC performance. For datasets in which the ligands provided are structurally distinct, holding no relationship whatsoever between them, the conventional machine learning techniques, used as a baseline in our experiments, perform better.

During the COVID-19 pandemic, at the time during which this dissertation was written, the general public developed an appreciation for advancements in the pharmaceutical industry for relatively rapid development of new drugs. Machine learning can be an effective enabler for such developments, as it can ameliorate and refine components of the drug discovery process. The results obtained through such techniques can only be verified experimentally, but they can drive research and innovation in the industry forward. Having motivated the reasons behind the need for few-shot learning approaches, we hope that our contributions inspire further research in this problem domain. Tackling the low-data problem in virtual screening can lead to more efficient drug development, leading to faster access to medicine, which can directly and positively impact the lives of many around the world.

— A —

Running Jupyter Notebooks

We open-source the developed code for this study, which is made available on GitHub¹. The developed code is contained within Jupyter notebooks, which are run on Google Colab. To successfully run the projects, Google Drive needs to be mounted to the Colab environment. Running all the code blocks will mount your Google Drive automatically. We made use of the Colab Pro paid plan, which supplies faster GPUs, more memory and longer runtimes. The notebooks can also be run locally by installing all the required packages.

Data can be downloaded from the links in Chapter 3 and saved to the `./data/raw/` directory as CSV files. The raw data can be processed directly by each of the machine learning pipelines contained in the provided notebooks. The code was tested with Tox21, MUV and the GPCR subset of the DUD-E datasets.

The Github repository contains empty file structures, which are used to save outputs from the Jupyter notebooks. Each network is contained within its own notebook. The folder structure must be uploaded to your Google Drive if running the notebooks on Google Colab. The following Jupyter notebooks are available:

- **Create Dataset.** Can be used to create a dataset to be used for further processing by the developed machine learning models.
- **Random Forest Benchmark.** Contains the code to run the random forest baseline machine learning model.
- **GCN Benchmark.** Contains the code to run the random forest baseline machine learning model.

¹Accessed From: <https://github.com/danielvlla/Few-Shot-Learning-for-Low-Data-Drug-Discovery>

- **Siamese Nets.** Contains the code to run the Siamese network few-shot learning model.
- **Matching Nets.** Contains the code to run the Matching network few-shot learning model with the iterative-refinement LSTM.
- **Prototypical Nets.** Contains the code to run the Prototypical network few-shot learning model with the iterative-refinement LSTM.
- **Relation Nets.** Contains the code to run the Relation network few-shot learning model with the iterative-refinement LSTM.
- **Prototypical Nets Tox21 ECFP.** Contains the code for training and testing prototypical networks on the Tox21 dataset with ECFP embeddings, rather than GCN learned embeddings.
- **Create DUD-E Dataset.** Contains functions for amalgamating the actives and decoys from the targets downloaded from the DUD-E website, into a CSV file.

The notebooks containing machine learning models contain a code block nested under the *Initiate Training and Testing* markdown header that contains setting of variables to match your environment. The other variables are hyperparameters you can experiment with. Default set values are the ones used for this study.

- `drive_path`. Set to `"/content/drive/MyDrive/{DIRECTORY_NAME}"`.
- `method_dir`. Needs to match the directory name from the supplied directory structure. If not, make sure that the directory name matches this variables' string.
- `dataset`. Takes on the values of `tox21`, `muv` or `dude-gpcr`

Outputs from the models will be saved to each specific directory. Loss plots will be saved to the `loss_plots` directory and ROC-AUC, PR-AUC and confusion matrices will be saved in the `graphs` directory.

References

- Altae-Tran, H., Ramsundar, B., Pappu, A. S., and Pande, V. Low data drug discovery with one-shot learning. *ACS central science*, 3(4):283–293, 2017.
- An, W. F. and Tolliday, N. Cell-based assays for high-throughput screening. *Molecular biotechnology*, 45(2):180–186, 2010.
- Arús-Pous, J., Johansson, S. V., Prykhodko, O., Bjerrum, E. J., Tyrchan, C., Reymond, J.-L., Chen, H., and Engkvist, O. Randomized smiles strings improve the quality of molecular generative models. *Journal of cheminformatics*, 11(1):1–13, 2019.
- Arya, H. and Coumar, M. S. Chapter 4 - lead identification and optimization. In Bhatt, T. K. and Nimesh, S., editors, *The Design Development of Novel Drugs and Vaccines*, pages 31–63. Academic Press, 2021. ISBN 978-0-12-821471-8.
- Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.
- Bennequin, E. Meta-learning algorithms for few-shot computer vision. *arXiv preprint arXiv:1909.13579*, 2019.
- Bento, A. P., Hersey, A., Félix, E., Landrum, G., Gaulton, A., Atkinson, F., Bellis, L. J., De Veij, M., and Leach, A. R. An open source chemical structure curation pipeline using rdkit. *Journal of Cheminformatics*, 12(1):1–16, 2020.
- Bissantz, C., Folkers, G., and Rognan, D. Protein-based virtual screening of chemical databases. 1. evaluation of different docking/scoring combinations. *Journal of medicinal chemistry*, 43(25):4759–4767, 2000.
- Bondy, J. A. and Murty, U. S. R. *Graph theory with applications*, Volume 290. Macmillan London, 1976.
- Brecher, J. Graphical representation of stereochemical configuration (iupac recommendations 2006). *Pure and applied chemistry*, 78(10):1897–1970, 2006.
- Bromley, J., Bentz, J. W., Bottou, L., Guyon, I., LeCun, Y., Moore, C., Säckinger, E., and Shah, R. Signature verification using a “siamese” time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(04):669–688, 1993.
- Bronstein, M. M., Bruna, J., Cohen, T., and Velicković, P. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.
- Brown, F. Chapter 35. chemoinformatics: What is it and how does it impact drug discovery. volume 33 of. *Annual Reports in Medicinal Chemistry*, pages 375–384.
- Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.
- Chan, W., Jaitly, N., Le, Q., and Vinyals, O. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages

- 4960–4964. IEEE, 2016.
- ChemAxon. Extended connectivity fingerprint ECFP, 2020. URL <https://docs.chemaxon.com/display/docs/extended-connectivity-fingerprint-ecfp.md>. [Last Accessed 2021-04-25].
- Chen, L., Cruz, A., Ramsey, S., Dickson, C. J., Duca, J. S., Hornak, V., Koes, D. R., and Kurtzman, T. Hidden bias in the dud-e dataset leads to misleading performance of deep learning in structure-based virtual screening. *PLoS one*, 14(8): e0220113, 2019.
- Chung, F. R. and Graham, F. C. *Spectral graph theory*. Number 92. American Mathematical Soc., 1997.
- David, L., Thakkar, A., Mercado, R., and Engkvist, O. Molecular representations in ai-driven drug discovery: a review and practical guide. *Journal of Cheminformatics*, 12(1):1–22, 2020.
- Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29:3844–3852, 2016.
- Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.
- Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.
- Duvenaud, D., Maclaurin, D., Aguilera-Iparraguirre, J., Gómez-Bombarelli, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. Convolutional networks on graphs for learning molecular fingerprints. *arXiv preprint arXiv:1509.09292*, 2015.
- Eckert, H. and Bajorath, J. Molecular similarity analysis in virtual screening: foundations, limitations and novel approaches. *Drug discovery today*, 12(5-6):225–233, 2007.
- Fawcett, T. An introduction to ROC analysis. *Pattern recognition letters*, 27(8):861–874, 2006.
- Fei-Fei, L., Fergus, R., and Perona, P. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):594–611, 2006.
- Ferreira, L. G., Dos Santos, R. N., Oliva, G., and Andricopulo, A. D. Molecular docking and structure-based drug design strategies. *Molecules*, 20(7):13384–13421, 2015.
- Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.
- Food, F. and Administration, D. Substance definition manual. *Standard Operating Procedure, "Substance Definition Manual," Version 5c, 94*, 2007.
- Gaulton, A., Hersey, A., Nowotka, M., Bento, A. P., Chambers, J., Mendez, D., Motow, P., Atkinson, F., Bellis, L. J., and Cibrián-Uhalte, E. The chEMBL database in 2017. *Nucleic acids research*, 45(D1):D945–D954, 2017.
- Geppert, H., Vogt, M., and Bajorath, J. Current trends in ligand-based virtual screening: molecular representations, data mining methods, new application areas, and performance evaluation. *Journal of chemical information and modeling*, 50(2):205–216, 2010.
- Glorot, X., Bordes, A., and Bengio, Y. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323. JMLR Workshop and Conference Proceedings, 2011.
- Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., and Aspuru-Guzik, A. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.
- Goodfellow, I., Bengio, Y., and Courville, A. *Deep learning*. MIT press, 2016.
- Gori, M., Monfardini, G., and Scarselli, F. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, Volume 2, pages 729–734. IEEE, 2005.

- Graves, A., Wayne, G., and Danihelka, I. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.
- Guo, Z., Zhang, C., Yu, W., Herr, J., Wiest, O., Jiang, M., and Chawla, N. V. Few-shot graph learning for molecular property prediction. In *Proceedings of the Web Conference 2021*, pages 2559–2567, 2021.
- Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.
- Hamza, A., Wei, N.-N., and Zhan, C.-G. Ligand-based virtual screening approach using a new scoring function. *Journal of chemical information and modeling*, 52(4):963–974, 2012.
- Hay, M., Thomas, D. W., Craighead, J. L., Economides, C., and Rosenthal, J. Clinical development success rates for investigational drugs. *Nature biotechnology*, 32(1):40–51, 2014.
- He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- Henaff, M., Bruna, J., and LeCun, Y. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.
- Hinton, G., Srivastava, N., Swersky, K., Tieleman, T., and Mohamed, A. Coursera: Neural networks for machine learning. *Lecture 9c: Using noise as a regularizer*, 2012.
- Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- Hospedales, T., Antoniou, A., Micaelli, P., and Storkey, A. Meta-learning in neural networks: A survey. *arXiv preprint arXiv:2004.05439*, 2020.
- Huang, N., Shoichet, B. K., and Irwin, J. J. Benchmarking sets for molecular docking. *Journal of medicinal chemistry*, 49(23):6789–6801, 2006.
- Huang, R., Xia, M., Nguyen, D.-T., Zhao, T., Sakamuru, S., Zhao, J., Shahane, S. A., Rossoshek, A., and Simeonov, A. Tox21challenge to build predictive models of nuclear receptor and stress response pathways as mediated by exposure to environmental chemicals and drugs. *Frontiers in Environmental Science*, 3:85, 2016.
- Hughes, J. P., Rees, S., Kalindjian, S. B., and Philpott, K. L. Principles of early drug discovery. *British journal of pharmacology*, 162(6):1239–1249, 2011.
- James, G., Witten, D., Hastie, T., and Tibshirani, R. Statistical learning. In *An introduction to statistical learning*, pages 15–57. Springer, 2021.
- Jiang, D., Wu, Z., Hsieh, C.-Y., Chen, G., Liao, B., Wang, Z., Shen, C., Cao, D., Wu, J., and Hou, T. 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.
- Johnson, M. A. and Maggiora, G. M. *Concepts and applications of molecular similarity*. Wiley, 1990.
- Kearnes, S., McCloskey, K., Berndl, M., Pande, V., and Riley, P. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30(8):595–608, 2016.
- Kim, S., Chen, J., Cheng, T., Gindulyte, A., He, J., He, S., Li, Q., Shoemaker, B. A., Thiessen, P. A., and Yu, B. Pubchem in 2021: new data content and improved web interfaces. *Nucleic acids research*, 49(D1):D1388–D1395, 2021.
- Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.
- Koch, G., Zemel, R., and Salakhutdinov, R. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, Volume 2. Lille, 2015.

- Kotsiantis, S. B., Zaharakis, I., and Pintelas, P. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160(1):3–24, 2007.
- Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.
- Kuhn, M., Letunic, I., Jensen, L. J., and Bork, P. The sider database of drugs and side effects. *Nucleic acids research*, 44 (D1):D1075–D1079, 2016.
- Lake, B., Salakhutdinov, R., Gross, J., and Tenenbaum, J. One shot learning of simple visual concepts. In *Proceedings of the annual meeting of the cognitive science society*, Volume 33, 2011.
- Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.
- Lavecchia, A. and Di Giovanni, C. Virtual screening strategies in drug discovery: a critical review. *Current medicinal chemistry*, 20(23):2839–2860, 2013.
- LeCun, Y. and Bengio, Y. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.
- LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *nature*, 521(7553):436–444, 2015.
- Li, Z., Liu, F., Yang, W., Peng, S., and Zhou, J. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.
- Maggiore, G. M. On outliers and activity cliffs why QSAR often disappoints, 2006.
- Malin, B. A., Emam, K. E., and O’Keefe, C. M. Biomedical data privacy: problems, perspectives, and recent advances, 2013.
- Mann, H. B. and Whitney, D. R. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, pages 50–60, 1947.
- Mayr, A., Klambauer, G., Unterthiner, T., Steijaert, M., Wegner, J. K., Ceulemans, H., Clevert, D.-A., and Hochreiter, S. Large-scale comparison of machine learning methods for drug target prediction on chembl. *Chemical science*, 9(24): 5441–5451, 2018.
- McKnight, P. E. and Najab, J. Mann-whitney u test. *The Corsini encyclopedia of psychology*, pages 1–1, 2010.
- Micheli, A. Neural network for graphs: A contextual constructive approach. *IEEE Transactions on Neural Networks*, 20(3): 498–511, 2009.
- Mitchell, T. M. *Machine Learning*. McGraw-Hill, New York, 1997.
- Morgan, H. L. 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.
- Mufei, L., Jinjing, Z., Jiajing, H., Wenxuan, F., Yangkang, Z., Yaxin, G., and George, K. Dgl-lifesci: An open-source toolkit for deep learning on graphs in life science. *arXiv preprint arXiv:2106.14232*, 2021.
- Munkhdalai, T. and Yu, H. Meta networks. In *International Conference on Machine Learning*, pages 2554–2563. PMLR, 2017.
- Mysinger, M. M., Carchia, M., Irwin, J. J., and Shoichet, B. K. Directory of useful decoys, enhanced (dud-e): better ligands and decoys for better benchmarking. *Journal of medicinal chemistry*, 55(14):6582–6594, 2012.
- Nichol, A. and Schulman, J. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*, 2(3):4, 2018.
- NIH. Tox21 data challenge 2014, 2014. Accessed on 20.08.2021.
- Pal, U. *Interaction of proteins with small molecules and peptides*. PhD thesis, Doctoral dissertation, Jadavpur University,

- 2016., 2016.
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems* 32, pages 8024–8035. Curran Associates, Inc., 2019.
- Prakash, N. and Gareja, D. Cheminformatics. *J Proteomics Bioinform*, 3:249–252, 2010.
- Ramsundar, B., Eastman, P., Walters, P., and Pande, V. *Deep learning for the life sciences: applying deep learning to genomics, microscopy, drug discovery, and more.* " O'Reilly Media, Inc.", 2019.
- RDKit. Open-source cheminformatics. <https://www.rdkit.org>, 2012. Last Accessed on 25/11/2021.
- Réau, M., Langenfeld, F., Zagury, J.-F., Lagarde, N., and Montes, M. Decoys selection in benchmarking datasets: overview and perspectives. *Frontiers in pharmacology*, 9:11, 2018.
- Reddy, A. S., Pati, S. P., Kumar, P. P., Pradeep, H., and Sastry, G. N. Virtual screening in drug discovery—a computational perspective. *Current Protein and Peptide Science*, 8(4):329–351, 2007.
- Rogers, D. and Hahn, M. Extended-connectivity fingerprints. *Journal of chemical information and modeling*, 50(5):742–754, 2010.
- Rohrer, S. G. and Baumann, K. Maximum unbiased validation (muv) data sets for virtual screening based on pubchem bioactivity data. *Journal of chemical information and modeling*, 49(2):169–184, 2009.
- Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *nature*, 323 (6088):533–536, 1986.
- Rupp, M., Tkatchenko, A., Müller, K.-R., and Von Lilienfeld, O. A. Fast and accurate modeling of molecular atomization energies with machine learning. *Physical review letters*, 108(5):058301, 2012.
- Saito, T. and Rehmsmeier, M. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PLoS one*, 10(3):e0118432, 2015.
- Sandryhaila, A. and Moura, J. M. Discrete signal processing on graphs. *IEEE transactions on signal processing*, 61(7): 1644–1656, 2013.
- Santorini, B. Part-of-speech tagging guidelines for the Penn Treebank Project. Technical report, Department of Computer and Information Science, University of Pennsylvania, 1990.
- Sarle, W. S. Neural networks and statistical models. Citeseer, 1994.
- Schlichtkrull, M., Kipf, T. N., Bloem, P., Van Den Berg, R., Titov, I., and Welling, M. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pages 593–607. Springer, 2018.
- Schuster, M. and Paliwal, K. K. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11): 2673–2681, 1997.
- Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A., and Vandergheynst, P. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine*, 30(3):83–98, 2013.
- Smusz, S., Kurczab, R., and Bojarski, A. J. The influence of the inactives subset generation on the performance of machine learning methods. *Journal of cheminformatics*, 5(1):1–8, 2013.
- Snell, J., Swersky, K., and Zemel, R. S. Prototypical networks for few-shot learning. *arXiv preprint arXiv:1703.05175*, 2017.
- Spitzer, G. M., Heiss, M., Mangold, M., Markt, P., Kirchmair, J., Wolber, G., and Liedl, K. R. One concept, three implementations of 3d pharmacophore-based virtual screening: distinct coverage of chemical search space. *Journal of*

- chemical information and modeling*, 50(7):1241–1247, 2010.
- Stärk, H., Beaini, D., Corso, G., Tossou, P., Dallago, C., Günnemann, S., and Liò, P. 3D infomax improves GNNs for molecular property prediction. *arXiv preprint arXiv:2110.04126*, 2021.
- Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P. H., and Hospedales, T. M. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1199–1208, 2018.
- Thrun, S. and Pratt, L. *Learning to learn*. Springer Science & Business Media, 2012.
- Vamathevan, J., Clark, D., Czodrowski, P., Dunham, I., Ferran, E., Lee, G., Li, B., Madabhushi, A., Shah, P., and Spitzer, M. Applications of machine learning in drug discovery and development. *Nature Reviews Drug Discovery*, 18(6): 463–477, 2019.
- Van De Waterbeemd, H. and Gifford, E. Admet in silico modelling: towards prediction paradise? *Nature reviews Drug discovery*, 2(3):192–204, 2003.
- Verdonk, M. L., Berdini, V., Hartshorn, M. J., Mooij, W. T., Murray, C. W., Taylor, R. D., and Watson, P. Virtual screening using protein-ligand docking: avoiding artificial enrichment. *Journal of chemical information and computer sciences*, 44(3):793–806, 2004.
- Vinyals, O., Blundell, C., Lillicrap, T., and Wierstra, D. Matching networks for one shot learning. *Advances in neural information processing systems*, 29:3630–3638, 2016.
- Voulodimos, A., Doulamis, N., Doulamis, A., and Protopapadakis, E. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018, 2018.
- Wallach, I. and Heifets, A. Most ligand-based classification benchmarks reward memorization rather than generalization. *Journal of chemical information and modeling*, 58(5):916–932, 2018.
- Wang, M., Zheng, D., Ye, Z., Gan, Q., Li, M., Song, X., Zhou, J., Ma, C., Yu, L., Gai, Y., Xiao, T., He, T., Karypis, G., Li, J., and Zhang, Z. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*, 2019.
- Wang, Y., Yao, Q., Kwok, J. T., and Ni, L. M. Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys (CSUR)*, 53(3):1–34, 2020.
- Wang, Y., Abuduweili, A., and Dou, D. Property-aware relation networks for few-shot molecular property prediction. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.
- Waring, M. J., Arrowsmith, J., Leach, A. R., Leeson, P. D., Mandrell, S., Owen, R. M., Paireudeau, G., Pennie, W. D., Pickett, S. D., and Wang, J. An analysis of the attrition of drug candidates from four major pharmaceutical companies. *Nature reviews Drug discovery*, 14(7):475–486, 2015.
- Weininger, D. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36, 1988.
- Wong, C. H., Siah, K. W., and Lo, A. W. Estimation of clinical trial success rates and related parameters. *Biostatistics*, 20(2):273–286, 2019.
- Wu, Z., Ramsundar, B., Feinberg, E. N., Gomes, J., Geniesse, C., Pappu, A. S., Leswing, K., and Pande, V. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.
- Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S. Y. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.
- Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

- Yan, S., Xiong, Y., and Lin, D. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Thirty-second AAAI conference on artificial intelligence*, 2018.
- Young, T., Hazarika, D., Poria, S., and Cambria, E. Recent trends in deep learning based natural language processing. *IEEE Computational Intelligence Magazine*, 13(3):55–75, 2018.
- Zang, R., Li, D., Tang, I.-C., Wang, J., and Yang, S.-T. Cell-based assays in high-throughput screening for drug discovery. *International Journal of Biotechnology for Wellness Industries*, 1(1):31–51, 2012.
- Zhu, X. and Goldberg, A. B. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130, 2009.