PADME: A Deep Learning-based Framework for Drug-Target Interaction Prediction

by

Qingyuan Feng

B.Sc., The University of Hong Kong, 2016

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

in the
School of Computing Science
Faculty of Applied Sciences

Approval

Name:	**Qingyuan Feng**
Degree:	**Master of Science (Computer Science)**
Title:	**PADME: A Deep Learning-based Framework for Drug-Target Interaction Prediction**
Examining Committee:	**Chair:** Jiannan Wang Assistant Professor
	Martin Ester Senior Supervisor Professor
	Artem Cherkasov Supervisor Professor Department of Urologic Sciences University of British Columbia
	Leonid Chindelevitch Supervisor Assistant Professor
	Kay C. Wiese Internal Examiner Associate Professor
Date Defended:	**Mar 13, 2019**

Abstract

In silico drug-target interaction (DTI) prediction is an important and challenging problem in biomedical research with a huge potential benefit to the pharmaceutical industry and patients. Most existing methods for DTI prediction including deep learning models generally have binary endpoints, which could be an oversimplification of the problem, and those methods are typically unable to handle cold-target problems, i.e., problems involving target protein that never appeared in the training set. Towards this, we contrived PADME (Protein And Drug Molecule interaction prEdiction), a framework based on Deep Neural Networks, to predict real-valued interaction strength between compounds and proteins without requiring feature engineering. PADME takes both compound and protein information as inputs, so it is capable of solving cold-target (and cold-drug) problems. To our knowledge, we are the first to combine Molecular Graph Convolution (MGC) for compound featurization with protein descriptors for DTI prediction. We used multiple cross-validation split schemes and evaluation metrics to measure the performance of PADME on multiple datasets, including the ToxCast dataset, which we believe should be a standard benchmark for DTI problems, and PADME consistently dominates baseline methods. The results of a case study, which predicts the binding affinity between various compounds and androgen receptor (AR), suggest PADME's potential in drug development. The scalability of PADME is another advantage in the age of Big Data.

Keywords: cheminformatics; deep learning; Molecular Graph Convolution; QSAR; drug-target interaction

Acknowledgements

Many thanks to Dr. Martin Ester and Dr. Artem Cherkasov for your guidance, and notably Dr. Evgenia Dueva, a former lab member in Dr. Cherkasov's lab. Dr. Fuqiang Ban and Dr. Michael Hsing gave us some useful information that we incorporated into this thesis. Helene Morin and Eric LeBlanc were in charge of wet-lab experiments for validating our results. I also thank the help and suggestions received from other fellow lab members, including but not limited to Zaccary Alperstein, Oliver Snow, Michael Lllamosa, Hossein Sharifi, Beidou Wang, Jiaxi Tang and Sahand Khakabimamaghani. We also express our gratitude towards our family and friends, especially Jun Li, as well as Wen Xie, Qiao He, Yue Long, Aster Li, Lan Lin, Xuyan Qian, Yue Zhang, Qing Rong, Si Chen, Fengjie Lun and Stephen Tseng, the list goes on; most important of all, the late Mr. Xiefu Zang.

We also thank George Lucas (and his fantastic prequel trilogy) and [48] for the inspiration of naming.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

Proteins are polymers made of amino acids, typically very large and complex, serving as the building blocks of organisms[1]. They also have important biological functions other than being the building block, including but not limited to catalyzing biochemical reactions (enzymes), regulating physiological processes (hormones), defending against pathogens (antibody), carrying vital substances (carrier protein), etc.

In biochemical processes and the development of diseases, there are usually some proteins performing critical functions. They are often called "target proteins", in the sense that, if a compound can interact with the protein in some way (typically binding), the biochemical properties of the protein would change, thus its activity levels could be improved or inhibited, which in turn greatly affects the disease. Figure 1.1 offers an illustration. The said protein is the "target" of the said compound, hence the name. Though "target" can include molecules other than proteins, for simplicity, we treat "target" and "protein" synonymously in this thesis, similarly, "drug" and "compound" are used interchangeably. Scientists and pharmaceutical companies often hope to find such compounds(drugs) to bind with target proteins, so that the disease can be suppressed or cured.

Thus, finding out the interaction strengths between compounds (candidate drugs) and target proteins is of crucial importance in the drug development process. However, it is both expensive and time-consuming to be done in wet lab experiments, while virtual screening using computational (also called *"in silico"*) methods to predict the interactions between compounds and target proteins can greatly accelerate the drug development process at a significantly reduced cost. Indeed, machine learning models for drug-target interaction (DTI) prediction are often used in computer-aided drug design [12].

Datasets used for training and evaluating machine learning models for DTI prediction often include compounds' interaction strengths with enzymes, ion channels, nuclear re-

[1]Including viruses

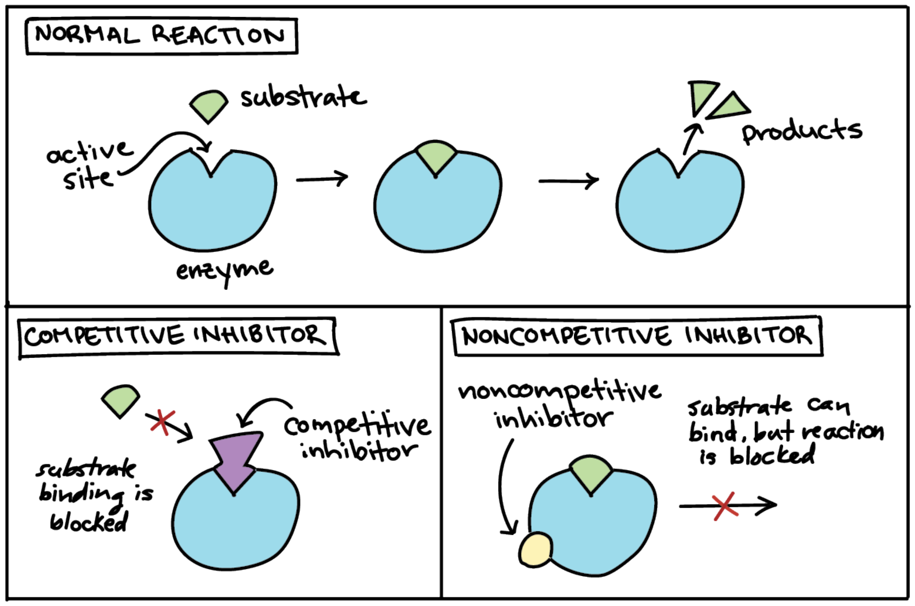

Figure 1.1: Illustration of how compound-protein interaction affects the biochemical properties of the protein. Includes competitive binding and non-competitive binding. The image only shows inhibition (antagonistic effect), but agonists are also possible in drug-target interaction. Image from [2].

ceptors, etc [62]. Traditionally, these datasets contain binary labels for the interaction of certain drug-target pairs, with 1 indicating a known interaction. Recently, the community has also explored the usage of datasets with real-valued interaction strength measurements [39, 17], which include the Davis dataset [11] that uses the inhibition constant (K_i), the Metz dataset [33] that uses the dissociation constant (K_d) and the KIBA dataset [52] whose authors devised their own measurement index.

1.2 Traditional Machine Learning models for DTI Prediction

Existing traditional machine learning methods for predicting DTI can be roughly divided into similarity-based and feature-based approaches, and most of them formulate the problem as a classification problem. Similarity-based methods depend on the assumption that compounds with similar structures should have similar effects. Feature-based methods construct feature vectors as input, which are generated by combining descriptors of compounds with descriptors of targets, and the feature vectors serve as inputs for algorithms such as support vector machine (SVM) [17].

SimBoost [17] and KronRLS [39] are two state-of-the-art methods for DTI prediction. Both of them have single outputs. KronRLS is based on Regularized Least Squares and utilizes the similarity matrices for drugs and targets to get the parameter values. SimBoost is a feature-based method, but in its feature construction, similarity matrices of the drugs and those of targets are also involved. These methods can both predict continuous values and binarized values. However, these methods either simply rely on similarities, or require expert knowledge to define the relevant features of proteins and compounds, called “feature engineering”. Additionally, they are often unable to model highly complex interactions within compound molecules [31] and between the compounds and their target proteins.

Deep Neural Networks (DNN) promise to address these challenges.

1.3 Deep Learning models for DTI Prediction

Deep learning, the machine learning method based on DNN, has been enjoying an ever-rising popularity in the past few years. It has seen wide and exciting applications in computer vision, speech recognition, natural language processing, reinforcement learning, and drug-target interaction prediction. DNNs can automatically extract important features from the input data, synthesize and integrate low-level features into high-level features, and capture complicated nonlinear relationships in a dataset [25, 44]. Deep learning-based DTI prediction has been shown to consistently outperform the existing methods and has become the new “golden standard” [9, 54, 27].

The current deep learning approaches to drug-target interaction prediction can be roughly categorized based on their neural network types and prediction endpoints. Simple feedforward neural networks, Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) have been adopted in various papers [61]. To our knowledge, almost all existing deep learning methods, except those that have 3D structural information as input, treat the problem as a classification problem, most of which are binary, namely active/inactive. Though there are deep learning models using 3D structural information that yield good results in regression problems [58, 15], the requirement of 3D structural information limits the applicability of a model since such information is not always available, so we do not consider them in this thesis.

As deep learning for DTI is still in its infancy, the current models have several disadvantages.

First, formulating the problem as a classification problem has several disadvantages: obviously, the classification result depends on a predefined binarization threshold, which introduces some arbitrariness into the data; some useful information is lost, for instance, true-negative and missing values may not be discriminated in some chemical datasets [39, 17]. On the other hand, if we formulate it as a regression problem, not only can we avoid the problems above, but given the regression results, the real-valued outputs can be easily converted to produce a ranking or classification. Some existing non-DNN methods formulate the problem as a regression problem, in which the interaction strength between the drug molecule and the target protein is a real number, serving as the regression target [17]. Common real-valued interaction strength metrics include K_i (inhibition constant),K_d (dissociation constant), etc.

The second problem is that most of the existing deep learning methods do not incorporate the target protein information into the network, except very few recent works, like [59]. As a result, the models are unable to solve the “cold target” problem, i.e. to predict the drug-target interactions for target proteins absent in the training dataset.

A recent model, DeepDTI [59], addressed the second problem by combining the protein information with the compound feature vector. It uses the classical Extended-Connectivity Fingerprint (ECFP) [42] for describing compounds, which relies on a fixed hashing function and cannot adjust to specific problems at hand. DeepDTI concatenates ECFP and Protein Sequence Composition (PSC) descriptors [6] (describing the target proteins’ sequence information) to construct a feature vector, which is fed into a Deep Belief Network (DBN) to predict a binary endpoint. DeepDTI outperformed the state-of-the-art methods on a dataset extracted from DrugBank.

1.4 Our model: PADME

In this thesis, we propose PADME (Protein And Drug Molecule interaction prEdiction), a deep learning-based framework for predicting DTI, which can be roughly categorized into the feature-based methods. PADME overcomes the limitations of the existing methods by predicting real-valued interaction strengths instead of binary class labels, and, to address the cold-start problems (drugs or targets that are absent from the training set but appear in the test set), PADME utilizes a combination of drug and target protein features/fingerprints as the input vector, where no feature engineering is required. The drug and target vectors can be generated from SMILES representation and Amino Acid sequence, respectively, without loss of information. Because the DBN used in DeepDTI has fallen out of favor in the deep learning community after Rectified Linear Units (ReLU) were introduced to improve the performance of feedforward networks, PADME uses a feedforward network, mainly composed of ReLU layers, to connect the input vector to the output layer. PADME adopts Molecular Graph Convolution (MGC) which is more flexible than ECFP, because it learns the mapping function from molecular graph representations to feature vectors [13, 20, 3], rather than a fixed-rule mapping. Similar to DeepDTI, we used Protein Sequence Composition (PSC) descriptor to represent the protein. To the best of our knowledge, this work is the first to integrate MGC with protein descriptors for the DTI prediction problem. In addition to the kinase inhibitor datasets used by previous researchers, we also used the ToxCast dataset [55], and we believe this large high-quality dataset, with its much larger variety of proteins, could be another useful benchmarking dataset for future researches of the same type.

We conducted computational experiments with multiple cross-validation settings and evaluation metrics. The results demonstrated the superiority of PADME over baseline methods across all experimental settings. Besides, PADME is more scalable than SimBoost and KronRLS since it does not rely on computationally expensive similarity matrices and can accommodate multiple outputs. As a case study, we also applied PADME to predict the binding affinity between some compounds and the androgen receptor (AR). We examined the top compounds among them and confirmed this prediction through literature research, suggesting that the predictions of PADME have practical implications.

We believe that PADME will be helpful in lots of tasks in medicinal chemistry, including but not limited to toxicity prediction, computer-aided drug discovery, precision medicine, etc.

1.5 Structure of this Thesis

The subsequent sections of the thesis are organized as follows. Chapter 2 will introduce the related work in more detail, specifically the traditional ML models and Deep Learning models for DTI prediction. Chapter 3 introduces the methods for compound featurization, protein featurization, and network structure, as well as the implementation. Chapter 4 will present the experiments conducted, introducing the baseline methods, datasets used, experimental design, etc. Chapter 5 is dedicated to the presentation of experimental results, including case studies. Chapter 6 is a Discussion, which clarifies some implementation and design choices, and outlines possible future directions to further this work. The last chapter concludes the whole thesis.

Chapter 2

Related Work

This chapter introduces a selected set of traditional machine learning models and deep learning models for DTI prediction.

2.1 Traditional Machine Learning models for DTI prediction

There are many models for DTI prediction, we are mainly interested in 2 of them: SimBoost [17] and KronRLS [39], which are state-of-the-art methods for the DTI regression task, while we are not as interested in classification models like [35]. They serve as the baseline models in our comparison studies.

2.1.1 SimBoost

Simboost predicts continuous DTI values using gradient boosting regression trees. Each drug-target pair corresponds to a continuous DTI value, and the authors defined 3 types of features to characterize the drug-target pairs: type 1 features for individual entities (drugs or targets); type 2 features, derived from the drug similarity networks and target similarity networks; type 3 features, which are derived from drug-target interaction network. The 3 types of features are concatenated to form a feature vector.

Let $x_i \in R^d$ denote the vector of features for the i-th drug-target pair, while $y_i \in R$ is its binding affinity. The score $\hat{y}_i$ predicted for input x_i is computed as follows:

$$\hat{y}_i = \phi(x_i) = \sum_{k=1}^{K} f_k(x_i), f_k \in F$$

where K is the number of regression trees and F is the space of possible trees.To learn the set of trees f_k, they defined a regularized objective function:

$$L(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k)$$

where l is a loss function that evaluates the prediction error, Ω is a function that penalizes overfitting. The model is trained additively: for each iteration t, the tree space F is searched to find a new tree f_t that optimizes the objective function, which is added to the ensemble afterwards. Trees that optimize $L^{(t)}$ are iteratively added to the model for a number of pre-specified iterations.

SimBoost cannot handle cold-start problems, which means it does not work for pairs in the test set with a drug or target that is absent from the training set.

2.1.2 KronRLS

KronRLS stands for Kronecker Regularized Least Squares. It learns a prediction function $f(x)$ for drug-target pairs, with the following objective function, in which the definition of x and y are similar to those in SimBoost, and m is the total number of drug-target pairs:

$$J(f) = \sum_{i=1}^{m}(y_i - f(x_i))^2 + \lambda\|f\|_k^2$$

Where $\|f\|_k^2$ is the norm of f, associated to a kernel function k, λ is a user-specified parameter. A minimizer of this objective function is:

$$f(x) = \sum_{i=1}^{m} a_i k(x, x_i)$$

In which k is a kernel function, and can be a symmetric similarity measure between two drug-target pairs. We can compute a similarity matrix K which contains all the $k(x_i, x_j)$ for all i and j, using the similarity matrices K_d and K_t for drugs and targets by $K_d \otimes K_t$, in which $\otimes$ denotes Kronecker product. If the training set contains all possible drug-target pairs, the parameter vector a can be obtained by solving the following system of linear equations:

$$(K + \lambda I)a = y$$

Where I is an identity matrix. If only a subset of possible drug-target pairs are available in the training set, the authors suggest to use conjugate gradient with Kronecker algebraic optimization to solve the system of linear equations, in order to get the parameter vector a.

Thus, unlike SimBoost, KronRLS is applicable to cold-start problems.

2.2 Deep Learning models for DTI prediction

This section is focused on those deep learning models that do not use 3D information.

2.2.1 Ma et al., 2015

This study [27] was a follow-up on a Merck-sponsored Kaggle challenge on drug discovery. The authors pointed out that Neural Nets were already used for QSAR problems in 1990s, but could not outperform more robust methods like support vector machine (SVM) and random forest (RF), until very recently, due to the improvement in computational power and introduction of new techniques in DNN.

The prediction endpoints are pharmacokinetic properties like ADME (absorption, distribution, metabolism and excretion), as well as target protein inhibition assay results. Those endpoints are real-valued, so the model is a regression model. Sadly, the datasets they used are proprietary, and they did not open source their code, so direct comparisons cannot be done by other researchers. Possibly due to the proprietary nature of the study, they did not explicitly explain the molecular descriptors they used, just saying that their set of descriptors include AP (atom pair) and DP (donor-acceptor pair).

They used multi-task networks to predict DTI across different datasets, called joint-DNN, they also trained individual DNNs with single datasets, and compared their performance against each other, as well as against RF, the state-of-the-art method by then. The molecular descriptors are used as inputs to the model. In the output layer, each neuron represents a prediction endpoint, which could be an assay corresponding to a target protein. They showed that DNN consistently outperforms RF, and joint DNNs slightly outperforms individual DNNs, which could be due to some transfer learning.

Apparently, because there is no protein information in the model input, the model cannot predict for "cold proteins", though it can predict for "cold drugs". Because the authors did not open source their code, we were unable to run comparison experiments using this model.

Given the results of this paper, we tried to implement multi-task learning in the early phase of this project, hoping to use one model to predict all endpoints in this study at once, which was both time-consuming and unsuccessful. So we had to resort to simple DNN, where we predict the endpoints of only one dataset in each Neural Network, which is a much more dedicated and straightforward model, without the additional layer of complexity. Future researchers might find it interesting to construct a model that we initially envisioned.

2.2.2 DeepDTI

DeepDTI [59] is the first deep learning-based model that combines the compound vector and protein vector, each only based on its sequence information, so it can predict the interactions for cold drugs or cold targets. It uses Deep Belief Network (DBN) to connect the input layer to the output layer, in order to construct a **binary classifier** in which the last layer is performing logistic regression.

DeepDTI uses PSC (Protein Sequence Composition descriptor) [6] as protein feature vector, which led us to the same choice (more on it in Section 3.4). It has 8420 entries. For compound featurization, it used ECFP2, ECFP4, and ECFP6, where the numbers indicate the “diameter” used in calculating ECFP fingerprint (more on it in Section 3.3), each of the ECFP fingerprints has 2048 entries, so there are 6144 entries for compound feature vector. After concatenating the compound and protein feature vectors, the resulting vector has 14564 entries in total.

The training of DBN involves the layer-wise unsupervised training and the supervised fine-tuning. Interested readers can refer to [59] for details. Because it is a classification model, we cannot use it as a baseline method to compare with.

Chapter 3

Method

3.1 Overview and Problem Definition

PADME is a deep learning-based DTI prediction model which uses the combined small-molecule compound (candidate drug) and target protein feature vectors.

For each drug-target pair, there could be one or more real-valued interaction strength measurements. PADME takes in the information of the drug and target (details to be followed in Section 3.2), and produces the prediction of the interaction strength. So each input x_{ij} is a concatenation of compound (drug) vector d_i and protein vector p_j, while the output y_{ij} can be either a real-valued number or vector.

We consider two variants of PADME with either Molecular Graph Convolution (MGC) [13, 20] or ECFP [42] as the compound featurization method. For the protein, we use Protein Sequence Composition (PSC) descriptor [6]. The rationale for choosing those compound and protein featurization methods are explained in Section 3.3 and Section 3.4. In fact, PADME is compatible with all kinds of protein descriptors and molecular featurization methods, but we will not study other variants here.

The compound vector is concatenated with a target protein vector to form the Combined Input Vector (CIV) for the neural network. PADME predicts one or more real-valued interaction strengths, i.e., it solves **DTI regression** problems.

The structure of the network is shown in Figure 3.1. If we use the MGC network to get the molecular vector, that network will be trained together with the feedforward network connecting the CIV to the prediction endpoint in an end-to-end fashion. If we use PADME-ECFP, then there is no network before the CIV.

3.2 Processing of raw inputs

Conceptually we use the MGC or ECFP as inputs representing compounds, PSC as inputs representing proteins, but the raw inputs are actually different.

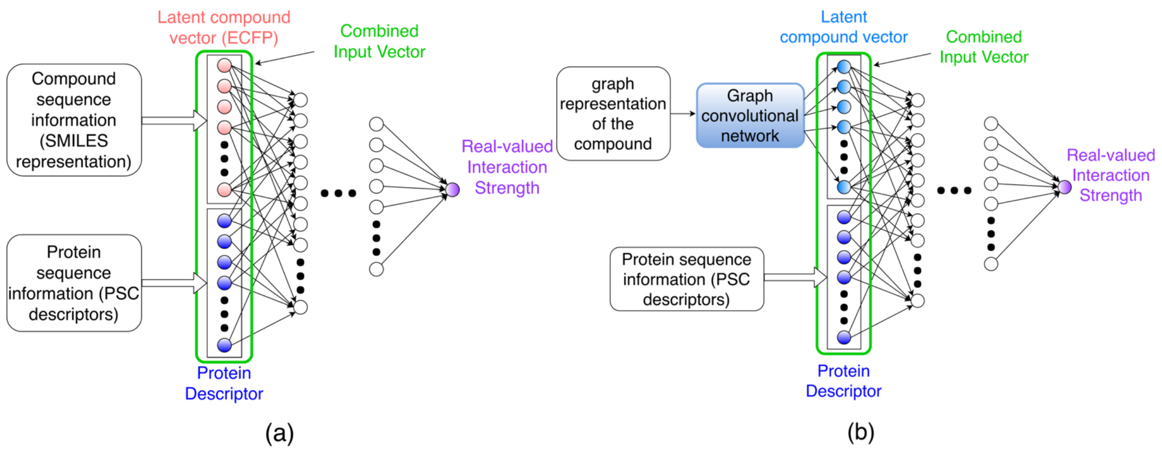
Latent compound vector (ECFP)
Combined Input Vector
Compound sequence information (SMILES representation)
Protein sequence information (PSC descriptors)
Real-valued Interaction Strength
Protein Descriptor
(a)
graph representation of the compound
Graph convolutional network
Latent compound vector
Combined Input Vector
Protein sequence information (PSC descriptors)
Real-valued Interaction Strength
Protein Descriptor
(b)

Figure 3.1: **a) PADME-ECFP architecture**. The Extended-Connectivity Fingerprint was used as the molecular input to the model. **b) PADME-GraphConv architecture.** Note that the graph convolutional network generating the latent molecular vector is trained together with the rest of the network, while the protein descriptor generation process is independent from the training of the network. The black dots represent omitted neurons and layers.

To make the model as parsimonious and general as possible, we only use the SMILES (Simplified Molecular Input Line-Entry System) representation of the chemical compounds as part of the raw input to the system, which is comprised of ASCII characters. For example, benzene can be represented as "C1=CC=CC=C1", acetic acid can be represented as "CC(=O)O". SMILES is one among many descriptors and fingerprints (including ECFP) to represent molecules. Because we consider the ECFP and MGC variants of PADME, in the real implementation, our program either converts SMILES representation to ECFP, or to graph representation[1] and construct a Molecular Graph Convolution (MGC) network based on the graph representation to get a feature vector, and this ECFP or MGC-output feature vector is then used as part of the input to the network. As shown in Figure 3.1, if we convert SMILES representation to ECFP, this conversion process is not part of the bigger neural network, while if we convert SMILES to graph representation and build MGC network, this MGC network is part of the bigger network. Because we do not consider 3D information anyway, those conversions from SMILES representation do not cause inaccuracies or information loss.

For protein representation, we used PSC descriptors obtained from amino acid sequences as the raw input. They were generated independently from the training process: we used the propy python package [6] to get PSC descriptors, and manually added a binary entry

[1]Graph representation and graphical representation of compounds are different concepts: a graph representation denotes atoms by nodes and bonds by edges, while a graphical representation of a molecule is a 2D representation commonly seen.

indicating phosphorylation (elaborated in Section 3.4). Afterwards, PSC was saved in a standalone file, which the program reads into the memory in the runtime, as the other part of the input to the network. The storage and retrieval of PSC was similar to key-value pairs.

3.3 Compound Featurization

There has been a lot of research on representing small molecules (compounds) as a descriptor or fingerprint.

Among the traditional molecular descriptors and fingerprints, ECFP [42] is widely adopted as the state-of-the-art method for compound featurization [13], and was also used in DeepDTI [59]. It produces a binary vector with typically 1024 or 2048 entries, the 1 or 0 at each position signals the existence or absence of a certain chemical substructure, like a functional group. Like MGC to be mentioned later, ECFP takes into account the chemical environment adjacent to each atom, and users can pre-specify the "radius" of the chemical environment surrounding each atom. It gives a comprehensive and simple summary of the molecular structure. However, like most other fingerprints like MACCS or DRAGON, it has a fixed set of mapping and hashing functions, unable to be tailored for the specific task at hand automatically.

DNN, especially MGC, can be used to generate more flexible feature vectors. Instead of depending only on the molecule, compound feature vectors generated using DNN depend on both the molecule and the prediction task (Boolean or continuous). DNN-derived feature vectors can outperform the ECFP baseline and at times offer some good interpretability [13, 3, 61].

MGC [13, 20, 3] is an extension of Convolutional Neural Network which learns a vector representing the compound from the graph-based representation of the molecule. In the graph representation of molecules, the atoms are denoted by nodes, while the bonds are denoted by edges. MGC takes into account the neighbors of a node when computing the intermediate feature vector for a specific node, and the same operation is applied to the neighborhood of each node (atom), hence it is analogous to ordinary convolutional networks typically used in Computer Vision [13, 16]. Originally, it was proposed as a differentiable version of ECFP [13]. Due to the GraphConv model [60] among MGC models being more recent and popular with an easier implementation, we use the GraphConv model as a representative of MGC under the time and resource constraints.

We applied both types of compound featurization methods: ECFP and GraphConv, and compared their performances.

3.4 Target Protein Featurization

Mapping a protein into a feature vector is a task in proteochemometrics. However, most existing methods in proteochemometrics require expert knowledge and often involve 3D structural information [57, 40], like PLIF (Protein-Ligand Interaction Fingerprint), which is often not available. Thus, we only considered sequence information for both drugs and targets in this work to make our model more generally applicable.

Still, there exist many schemes to represent the target protein as a feature vector based on its amino acid sequence information.

DeepDTI [59] used Protein Sequence Composition (PSC) descriptor, which has 8420 entries for each protein, consisting of amino acid composition (AAC), dipeptide composition (DC), and tripeptide composition (TC) [6]. The AAC and DC are the occurring frequencies of the amino acids or dipeptide sequences, hence they are real-valued numbers, while the TC is the binary status of whether a tripeptide sequence appears in the protein. Because there are 20 amino acids in biological proteins, there are 400 unique dipeptides, and 8000 tripeptides. It captures rich information and does not transform the protein as much as some other protein descriptors (which implies less human knowledge required and less information loss), which we think could be a desirable attribute as the input to a neural network. In addition to the 8420 entries for each protein sequence, we added an additional binary entry signaling the phosphorylation status so that the Davis dataset in Section 4.2 can be represented more accurately, with '1' denoting phosphorylated, resulting in 8421 entries in total.

[35] used PSSM (Position Specific Scoring Matrix) descriptor to represent the protein, which focuses on dipeptide sequences and is related to the evolutionary history of proteins [46]. It is observed that PSSM performed pretty well. Other popular protein sequence descriptors include Autocorrelation, CTD (Composition, Transition and Distribution) descriptor, Quasi-sequence order, etc [6].

As PSC contains rich information (like tri-peptide sequence occurrence) with high dimensionality, and has already shown promising performance in deep learning-based models for DTI prediction [59], we use PSC in this research. There could be future comparisons of the performance of PSC and other protein featurization methods as an extension to this work.

3.5 Architecture of the Deep Neural Network

PADME uses a feedforward neural network taking the CIV as the input, which is much simpler and more popular than the DBN used in DeepDTI [59]. The PADME architecture has one output neuron per prediction endpoint, i.e., one output neuron for most datasets, and 61 output neurons for the ToxCast dataset in Section 4.2. DNNs with single output neuron are called single-task networks, and those with multiple output neurons are called multi-task networks. Although we only consider the DTI regression problem in this thesis, PADME can also be used for constructing classification models with minimal changes, either by binarizing the continuous prediction results or by directly using a softmax/sigmoid layer as the output layer, of which the latter could be more preferable.

For regularization, we use Early Stopping, Dropout and Batch Normalization techniques [16]. Hyperparameters like dropout rates are automatically searched to find the best set of them before running cross-validation, as elaborated in Section 4.3. The Adam optimizer [23] was used to train the network. The activation functions used for fully connected layers are all Rectified Linear Units (ReLU).

3.5.1 Hyperparameters to be tuned and their ranges

As to be mentioned in Section 4.3, we use Bayesian Optimization to automatically search the hyperparameters. For each of the hyperparameters, we set an initial value x, and we let the search range to be 4, which means that the system will randomly select values ranging from $\frac{x}{4}$ to $4x$.

Table 3.1 lists the hyperparameters to be tuned and their initial values. Those initial values were heuristically obtained from some trial runs before the main round of hyperparameter tuning.

Table 3.1: Hyperparameters to be tuned for each PADME model and their initial values. Note that the system will randomly choose values ranging from $\frac{x}{4}$ to $4x$ for initial value x, except the epoch number, which is only tuned through early stopping.

PADME-ECFP		PADME-GraphConv	
Name of parameter	initial value	Name of parameter	initial value
dropout probability	0.262	dropout probability	0.1
batch size	155	batch size	128
epoch number	180	epoch number	180
learning rate	0.000311	learning rate	9.789e-05
number of dense layers	2	number of dense layers	3
number of nodes in each dense layer	1211	number of nodes in each dense layer	512
—	—	number of filters in the GraphConv network	128
—	—	number of fully connected nodes in the GraphConv network	256
L2 weight decay coefficient	0.0005	—	—

3.6 Time Complexity

PADME does not require drug-drug or target-target similarity matrices or matrix factorization, so it is much more scalable than KronRLS and SimBoost. Suppose there are n compounds and m proteins, since KronRLS and SimBoost need the similarity matrices, both of the methods have at least $O(n^2 + m^2)$ time and space complexity, SimBoost involves matrix factorization so it is even more expensive. But in each epoch of PADME's training process, the time complexity only depends on the number of drug-target pairs in the training set, which, in the best case, is $O(max(n, m))$, and $O(nm)$ in the worst case. There is no closed-form relationship between the optimal number of epochs required and n or m, so it is uncertain whether PADME is actually faster when we are trying to get the best results. Nevertheless, we can get some very crude results after running PADME for one or a constant number of epochs using $O(max(n, m))$ to $O(nm)$ time, while KronRLS and SimBoost strictly require at least $O(n^2 + m^2)$ time to get any results. In our real implementation, the best epoch numbers across the datasets range from several dozens to over 100, we guess the optimal number of epochs might increase with n and m sub-linearly, or close to constant.

3.7 Implementation

The model was constructed based on the implementation of the DeepChem python package [41], in which RDKit [24] was used; the networks were constructed using TensorFlow 1.3 [1].

The experiments were conducted on a Linux server with 8 Nvidia Geforce GTX 1080Ti graphics cards, among which 4 were used. The server has 40 logical CPU cores and 256 GB of RAM. A computer with less than 110 GB RAM might not be able to perform cross-validation for the ToxCast dataset using GraphConv-based PADME.

Chapter 4

Experiments

4.1 Methods to compare against

As mentioned in Chapter 2, there are two baseline methods used in the experiments: SimBoost [17] and KronRLS [39].

In addition, to investigate the usefulness of including protein feature vector (PSC in this thesis) in PADME, we implemented a version of PADME with only compound information as input. Different from the full PADME, this version has one output unit for each specific target protein, resulting in a network structure similar to that of [27]. Though [27] did not open source its code, these DNN models can serve as its proxies for comparison. Similar to PADME, we considered ECFP and GraphConv variants of this DNN model. We call these PADME versions Compound-Only DNNs later in this thesis to avoid confusions.

4.2 Datasets and Preprocessing

Similar to [17], we used kinase inhibitor datasets. Following its naming convention, we call them Davis dataset [11], Metz dataset [33] and KIBA dataset [52], respectively. However, the versions of these datasets curated by [39] that [17] used was slightly different from the original dataset, and did not give the corresponding justifications. We thus used the data provided by the respective original authors, then preprocessed them ourselves as described in Section 4.2.1. We assume the observations within each dataset are under the same experimental settings. Metz dataset contained lots of imprecise values, which we discarded in the preprocessing step.

Because of the limitations of SimBoost and KronRLS, we filtered the datasets. The original KIBA dataset contains 52498 compounds, a large proportion of which only have the interaction values with very few proteins. Considering the huge compound similarity matrix required and the time-consuming matrix factorization used in SimBoost, it would be infeasible to work directly on the original KIBA dataset. Thus, we had to filter it rather aggressively so that the size becomes more manageable. We chose a threshold of 6 (drugs and targets with no more than 6 observations are removed), more lenient than the threshold of 10 used in [17], aiming at a reduction of the unfair advantages that SimBoost can gain by keeping only the denser submatrix of the interaction matrix.

For the Metz and Davis datasets, as SimBoost cannot handle cold drug/target problem, we had to ensure that in creating Cross-Validation folds, each drug or target appear in at least 2 folds, thus those drugs/targets with no more than 1 observation are discarded.

We also used the ToxCast dataset [55], containing a much larger variety of proteins [56]. It contains toxicology data obtained from high-throughput *in vitro* screening of chemicals, mainly measured in AC_{50}, which means the concentration at half of the maximum activity. The prepared dataset (see Section 4.2.1) contains observations for 530605 drug-target pairs. Its large size and coverage of diverse protein types allows us to test the robustness and scalability of computational models for DTI prediction. After the preprocessing, it still contains a total of 672 assays, compared to single assay/interaction strength measurement of the other 3 datasets. Some of those assays are closely related, but most of them are different from each other. Because it contains so many heterogeneous endpoints, we manually grouped those assays into 61 different measurements for interaction strength based on assay type, such that observations in each measurement are reasonably homogeneous, also increasing the number of observations for each measurement endpoint. The number of observations in each measurement range from ∼290 to ∼160,000. For the ToxCast dataset, we constructed multi-task networks, in which each measurement corresponds to a neuron in the output layer. As KronRLS and SimBoost are both single-task models, to evaluate the performance of those two models on the ToxCast dataset, one must train 61 models for each of them, which would be an extraordinarily tedious job, so we did not run the SimBoost and KronRLS models on ToxCast. This indicates PADME is not only possibly more scalable in the number of drugs/targets, but certainly also much more scalable in the number of endpoints, since it can have multiple outputs in one model. As the ToxCast dataset does not have the bottlenecks imposed by KronRLS and SimBoost, we did not filter it.

Please refer to table 4.1 for the sizes of the datasets after filtering.

Table 4.1: Dataset sizes after filtering.

Dataset	Number of drugs (compounds)	Number of target proteins	Total number of drug-target pairs used
Davis	72	442	31824
Metz	1423	170	35259
KIBA	3807	408	160296
ToxCast (No filtering)	7657	335	530605

We applied the same numerical transformation as [17] to the datasets: $transformed = 4 - log_{10}(original)$. For the ToxCast dataset, we changed the inactive value from 1,000,000 to 1,000, so that there would be no large gaps in the distribution after transforming the data.

To clarify, the missing values are represented as NA in the datasets, so they are clearly different from the truly inactive values.

4.2.1 Preprocessing of Datasets

This subsection briefly introduces the details of the dataset preprocessing. Uninterested readers can skip this part.

Davis dataset

The compound names were extracted from Davis dataset [11]. The corresponding compound CIDs and SMILES strings were extracted from PubChem. NCBI GenBank Protein accession numbers from Davis dataset were used to download the corresponding amino acid sequences via NCBI Batch Entrez [37]. As sequences with accession numbers P0C1S8 and P0C264 were no longer available in GenBank Protein, their updated versions P0C1S8.2 and P0C264.2 were used. Protein sequences were modified according to descriptions from the original paper, e.g. mutations were introduced and only sequences corresponding to specified domains, if any, were left (domains were detected according to GenBank Protein domains annotation).

Metz Dataset

Kinase names extracted from Metz Dataset [33] were searched in KinBase [22]. The gene names found were saved and the corresponding amino acid sequences were extracted from Human Kinome Database [28] [22]. Compounds with identical SMILES strings, but different ChEMBL IDs and activity measurement results were deemed suspicious, and filtered out in the execution of the program.

KIBA dataset

ChEMBL IDs and protein IDs were extracted from KIBA dataset [52]. Canonical smiles strings were loaded from ChEMBL database [14] via ChEMBL webresource client [10]. NCBI GenBank Protein accession numbers from KIBA dataset were used to download the corresponding sequences via NCBI Batch Entrez.

ToxCast Dataset

The following file archives were downloaded from ToxCast website:

1. INVITRODB_V2_SUMMARY (October 2015). File Assay_Summary_151020.xls contains summary information about assays. File oldstyle_ac50_Matrix_151020.xls contains summary of testing results.

2. DSSTox_ToxCastRelease_20151019. File DSSTox_ToxCastRelease_20151019.xls contains summary of chemicals tested.

Compound structures contained in DSSTox_ToxCastRelease_20151019.xls were processed using MOE 2013.8 [53] as follows: water samples, mixtures with unidentified content, and polymers were excluded; structures were "washed" with MOE: salts were split and the largest part of each salt was retained, the structures were then neutralized; compounds containing metal atoms were removed; duplicated structures were filtered using MOE sdsort tool.

In the file Assay_Summary_151020.xls, only assays with single corresponding "intended target" were selected and split in groups, whose Uniprot IDs were extracted and used to get protein sequences from Uniprot [8].

4.3 Experimental Design

To examine PADME's prediction power, we used cross-validation (CV), which is the convention of the prior researches, also because we believe the comprehensive coverage of the whole dataset will offer a more thorough evaluation of the performance of the model, rather than only using 1 hold-out test set. To measure the performance of the model under different settings, multiple CV splitting schemes were employed to evaluate the predictions of the models trained from the training sets against the known interaction strengths in the test sets. The performances of PADME-ECFP and PADME-GraphConv were compared against each other under identical settings.

We performed 5-fold CV. For SimBoost to work, every compound (candidate drug) or target must be present in at least 2 folds, this splitting scheme is called "warm split" in this thesis. There are no such restrictions for KronRLS, since it can handle cold-start data. Since we did not run SimBoost on ToxCast data, there is no need to perform warm-split on it, we then used random split in that case. If we force a warm split on the ToxCast dataset, a filter threshold of 1 must be used to reduce the size of the dataset, which is undesirable. As cold-start prediction is an important objective in DTI prediction (and an advantage of PADME), we also included cold-splitting in constructing the cross-validation folds, such that all compounds (candidate drugs) in the test fold are absent from the training fold (cold-drug split), or all targets in the test fold are absent from the training fold (cold-target split). In addition, similar to [31], we also implemented a cold-drug cluster split, using single-linkage clustering with Tanimoto similarity (Jaccard distance) to create compound clusters. Compounds whose ECFP4 fingerprint had higher similarity than 0.7 were assigned to the same cluster. Compounds belonging to the same cluster were assigned to same folds, so that compounds in the validation fold would not be similar to those in the training fold. The cold-drug cluster split can prevent the performance estimation from being overly optimistic. Though [39] suggested another splitting scheme which results in simultaneous cold-drug and cold-target in each validation fold, as it greatly decreases the size of the training set in each fold (4/9 of the original data instead of 4/5 in other splitting schemes), we decided that it would cause unfair comparison and did not use it.

For every dataset, we performed four types of CV splitting (warm, cold-target, cold-drug, cold-drug cluster), and for every CV splitting scheme, we calculated the prediction errors of the applicable models (KronRLS and PADME for all splitting schemes, SimBoost for warm splits only). To reduce the random effects, we repeated the splitting several times for each splitting scheme on Davis, Metz and KIBA datasets and calculated the average values of the evaluation metrics of the prediction results across the splits. For the Davis and Metz datasets, we repeated 3 splits for each splitting scheme; for the KIBA dataset, we did 2 for each, as it is a much bigger dataset; for the ToxCast dataset, the largest one, we only did 1 split for each scheme. The Compound-Only DNNs (as mentioned in Section 4.1) take only compound information as input and predict the response for multiple proteins simultaneously. Therefore, they cannot handle cold-target scenarios, and it is unnatural to test them in a warm-split scenario. We only use them to compare against PADME in cold-drug splits.

Not only do we have multiple splitting methods, we also used multiple model settings and evaluation metrics. For each of PADME-ECFP and PADME-GraphConv, a single-task network was trained for every splitting scheme of every dataset, except ToxCast, for which we constructed a multi-task network with 61 output neurons to avoid the complexity resulting from 61 separate single-task networks.

We also wanted to investigate whether PADME can predict the ordering of the interaction strengths correctly, so in addition to metrics focusing on value correctness (RMSE (Root Mean Squared Error) and R^2), we also used metrics focusing on order correctness, like concordance index (CI). Using CI as a metric in cheminformatics setting was proposed by [39]. It measures the probability of correctly ordering the non-equal pairs in the dataset, ranging across [0, 1], with bigger values indicating better results. If you use the same value (e.g. mean value of the training set) as the predicted results across the test set, the CI would be 0.5. We note that the CI neglects the magnitude of values while focusing on the pairwise comparison, and it does not consider the prediction correctness for datapoints that truly have values equal to each other. Thus, CI should be used alongside other metrics like RMSE. However, in virtual screening, we are typically only interested in the top predictions, so that the drawback of neglecting the magnitude is not a big concern.

To improve the readability of the reported results for the ToxCast dataset, the performance metrics are averaged across the 61 different measurements, weighted by the number of records for each of the measurements, so the results reported for the ToxCast dataset look the same as other datasets with single endpoints. For Compound-Only DNNs, it is slightly more complicated, since we need to pool similar endpoints together before calculating the metric, instead of calculating a weighted average of evaluation metrics across different endpoints, but the basic idea is the same. In Compound-Only DNNs, the special case is the ToxCast dataset, where we pool the endpoints across the 61 subgroups of endpoints, calculate the metrics for each subgroup, and compute the weighted average across the 61 subgroups.

As an exploratory analysis of the datasets, we found the ToxCast to be special. As shown in Figure 4.1, the transformed ToxCast dataset is extremely concentrated at a value of 1 which corresponds to no interaction. This led us to ignore the R^2 values for this dataset: because R^2 is sensitive to the overall departure of the predicted values from the true values, we argue that the huge concentration of values has rendered R^2 uninformative in measuring the performance of the model on the ToxCast dataset. This concentration of values also makes RMSE less informative than it otherwise would be (since one can blindly guess inactive values for all and still get pretty good RMSE), so we argue that CI is the most useful metric in the ToxCast dataset prediction evaluation. This pronounced imbalance in the dataset caused us to consider balancing it through oversampling (see Section 5.2.1).

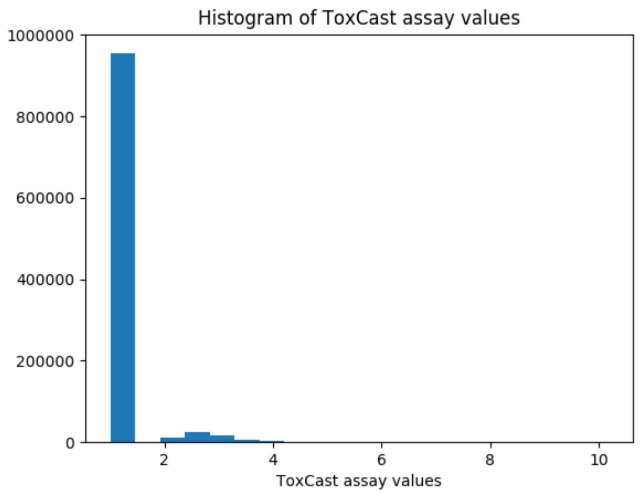

Figure 4.1: The histogram of the distribution of the negative log transformed ToxCast measurement results. The majority (over 94%) are concentrated at one inactive value.

Following the principle of parsimony, we wanted to use a minimal number of hyperparameter sets wherever possible, to keep the time and computational expenses manageable. If, instead, we do one hyperparameter tuning to get the hyperparameters for each CV iteration, to ensure a reasonable coverage of parameter space, it would have taken well over a month to run a CV for a dataset due to the intrinsic complexity of DNN models involving protein information, which would have been unrealistic, both for us and future users. So we cannot use the nested/double cross-validation as used in [4] and [32]. Also, since the datasets are not very large for deep learning, we wanted to use the full datasets for cross-validation, to maximize the training set in each iteration. Thus, in our hyperparameter tuning process, we randomly selected 90% of the elements in the dataset to be the training set, the remaining 10% to be the validation set, and the validation set was used for determining the best set of hyperparameters. Then in the 5-fold CV splits, we excluded the aforementioned validation set elements from each validation fold, but the validation set elements are retained in the training folds, thus in each CV iteration, the training folds are 80% the size of the dataset, while the validation fold is 18% the size of the dataset. This simultaneously makes the training folds as large as possible, and avoids bias in evaluation.

To efficiently tune the hyperparameters (like dropout rates, batch size, learning rate, number of layers, nodes per layer, etc.) for both PADME models and Compound-Only DNN models, we used Bayesian Optimization [45] implemented by the Python package pygpgo [19]. We also used early stopping to determine the optimal number of training epochs needed. To guide early stopping, we used $mean(RMSE) - mean(CI)$ calculated on the validation set as the composite score to be minimized. We only store one optimal set of hyperparameters per *(dataset, PADME variant)* pair, which were then used for all CV settings for that *(dataset, PADME variant)* pair. Note that, for simplicity and to examine the robustness of PADME, the set of hyperparameters found in the random splitting was used in all CV settings, though we believe better CV results could be achieved if the hyperparameter searching processes are specifically designed for that CV fold split scheme, e.g., for cold-target CV folds, we could use the hyperparameters found by running the Bayesian Optimization on cold-target splitted datasets.

The resulting networks typically have 2 or 3 fully-connected ReLU layers connecting the CIV to the output unit, with thousands of neurons in each of the layers. Each fully-connected layer is batch-normalized.

In addition to the quantitative experimental design introduced in this section, we also used plots to visualize the prediction performances so that a qualitative study can be conducted, as shown in Section 5.2.

Chapter 5

Experimental Results

5.1 Quantitative Results

Based on the experimental design in Section 4.3, we obtained the quantitative results for PADME. In the results listed in Tables 5.1 to 5.3, the bold numbers indicate the best mean CV results attained by PADME models for each setting. The sample standard deviation of CV mean results are also calculated, based on which we performed two-sample t-tests with unequal variances. For each t-test, the null hypothesis was that the mean of the CV results of the model is not worse than the best PADME result (boldfaced ones), while the alternative was that the model was worse than the best PADME model. For RMSE, worse means larger, while for CI or R^2, worse means smaller. The p-values are reported. We observe that the two versions of PADME dominate the other methods[1], including the Compound-Only DNN models though to a lesser degree, across all datasets and splits for all evaluation metrics.

We note the following exceptions. SimBoost outperforms PADME-GraphConv on the Metz dataset, which could be due to the small dataset size: PADME-GraphConv could be overfitting for Metz data, while SimBoost uses gradient boosting trees, a machine learning model better suited for small datasets than Deep Neural Networks. Because it does not use MGC, PADME-ECFP has a much smaller network than PADME-GraphConv, which may explain why the former peforms slightly better on the Metz dataset. However, we do not observe the same phenomenon on the Davis dataset, which has a similar size and even fewer entities. Comparing PADME-ECFP against Compound-Only ECFP and PADME-GraphConv against Compound-Only GraphConv, we observe that PADME consistently performs better in Concordance Index, for example, they outperform Compound-Only DNNs by around 10% or even more in Concordance Index on the ToxCast dataset. But in RMSE and R^2, Compound-Only DNNs sometimes perform similarly to the PADME models in Davis and

[1]Note that the SimBoost results reported here are considerably worse than the results reported in their original paper. It is because we have examined their source code and found they calculated MSE but reported it as RMSE.

KIBA datasets, or even **insignificantly** outperform PADME models. In general, it seems that PADME models are better than others, but more so in predicting the order, as reflected in CI. Besides, Compound-Only DNNs outperform KronRLS in general, except the Compound-Only GraphConv on the Davis dataset.

It is somewhat surprising that PADME-ECFP is not outperformed by PADME-GraphConv; instead, it slightly outperforms PADME-GraphConv in many cases, though in general their performances are very close to each other. The Compound-Only ECFP models usually outperforms Compound-Only GraphConv. PADME-ECFP only takes about 23% of the time and 45% the space (RAM) of PADME-GraphConv in the training process and yields similar (and sometimes better) results, so PADME-ECFP is a more reasonable choice. Nonetheless, we cannot be certain that PADME-GraphConv and PADME-ECFP truly have similar performances, as there might be a better set of hyperparameters for each model that would differentiate their performances significantly. We think that the higher complexity of PADME-GraphConv introduced by the MGC network makes it harder to find a good set of hyperparameters, while it is relatively easier to find a good set of hyperparameters for PADME-ECFP which has a simpler network. This could be a possible reason why MGC cannot beat ECFP in our experiments. Thus, future researchers should continue investigating MGC and find better sets of hyperparameters in PADME-GraphConv, and perhaps propose better MGC models.

From Tables 5.1 to 5.3 we can observe an interesting phenomenon: when there are many compounds and few targets in the training set, the cold-drug predictions tend to outperform the cold-target predictions; on the other hand, when there are many targets and few compounds, the cold-target predictions tend to be better than the cold-drug ones. We hypothesize that it is because the models can be more robust in entities (drugs or targets) with more information in the training set, thus performing better in the corresponding scenario. This trend is not only present in the PADME models, but in KronRLS as well. It seems that the models also require much more types of compounds than proteins for learning their chemical features, as can be seen from the KIBA dataset, whose cold-drug and cold-target performances are very similar, though it has 3807 compounds and only 408 proteins. And, as expected, there is a universal trend that the performance of warm splits is always better than that of cold-drug, cold-drug cluster, or cold-target splits.

The use of cold-drug clusters prevents us from overestimating the performance of the models: in the Metz and KIBA datasets, the performances for cold-drug cluster CV are noticeably worse than those for cold-drug CVs, while the performances on the Davis and ToxCast datasets stay almost the same. This could be due to the different distributions of compounds in different datasets. We suggest that future researches also employ cold-drug clusters splits in their experiments, so that a more stringent evaluation could be performed.

The fact that PADME outperforms both SimBoost and KronRLS demonstrates the power of DNN to learn complicated nonlinear relationships between drug-target pairs and in-

Table 5.1: The regression performance across the datasets measured in RMSE (smaller is better), averaged across independent repetitions of CV. The mean RMSE are enclosed in square brackets; sample standard deviations are also reported. The best results in the PADME models are boldfaced, and one-sided two-sample t-tests are conducted against them. The blue values are insignificantly bigger (worse) ($p > 0.05$) than the boldfaced values, while the orange ones are insignificantly smaller (better) ($p < 0.95$) than them. The uncolored ones are significantly worse than the boldfaced values.

			RMSE					
Dataset	Cross Validation Splitting type	Value Type	PADME-ECFP	PADME-GraphConv	SimBoost	KronRLS	Compound-Only ECFP	Compound-Only GraphConv
Davis	Warm	mean	**[0.4287]**	[0.4313]	[0.4820]	[0.5729]	—	—
		std	0.0029	0.0029	0.0019	0.0051	—	—
		p-val	—	0.0969	4.25E-9	2.97E-12	—	—
	Cold Drug	mean	**[0.8054]**	[0.8280]	—	[0.8405]	[0.8038]	[0.8505]
		std	0.0118	0.0192	—	0.0210	0.0148	0.0238
		p-val	—	0.0312	—	0.0041	0.5733	0.0047
	Cold Drug Cluster	mean	**[0.7671]**	[0.7922]	—	[0.8368]	[0.8040]	[0.8351]
		std	0.0171	0.0228	—	0.0382	0.0435	0.0144
		p-val	—	0.0435	—	0.0024	0.0675	7.86E-5
	Cold Target	mean	**[0.5639]**	[0.5748]	—	[0.6596]	—	—
		std	0.0065	0.0080	—	0.0020	—	—
		p-val	—	0.0229	—	6.5E-7	—	—
Metz	Warm	mean	**[0.5556]**	[0.6100]	[0.5813]	[0.7813]	—	—
		std	0.0022	0.0111	0.0016	3.87E-4	—	—
		p-val	—	1.4E-4	4.29E-8	4.42E-10	—	—
	Cold Drug	mean	**[0.7119]**	[0.7533]	—	[0.7843]	[0.7738]	[0.7775]
		std	0.0016	0.0086	—	0.0052	0.0146	0.0045
		p-val	—	1.6E-4	—	2.41E-8	3.10E-4	3.16E-7
	Cold Drug Cluster	mean	**[0.7770]**	[0.8099]	—	[0.8315]	[0.8250]	[0.8386]
		std	0.0115	0.0089	—	0.0054	0.0075	0.0051
		p-val	—	5.8E-4	—	5.81E-5	5.56E-5	2.94E-5
	Cold Target	mean	**[0.7905]**	[0.8239]	—	[0.8989]	—	—
		std	0.0127	0.0107	—	0.0101	—	—
		p-val	—	0.0011	—	2.56E-7	—	—
KIBA	Warm	mean	[0.4334]	**[0.4247]**	[0.4689]	[0.6566]	—	—
		std	0.0069	0.0027	0.0010	1.74E-4	—	—
		p-val	0.0405	—	4.93E-6	2.01E-7	—	—
	Cold Drug	mean	**[0.6007]**	[0.6444]	—	[0.7024]	[0.6319]	[0.6421]
		std	0.0036	0.0149	—	0.0024	0.0045	0.0024
		p-val	—	0.0040	—	1.85E-8	3.93E-6	2.42E-6
	Cold Drug Cluster	mean	**[0.7132]**	[0.7263]	—	[0.7536]	[0.7029]	[0.7196]
		std	0.0270	0.0224	—	0.0039	0.0064	4.71E-4
		p-val	—	0.2409	—	0.0285	0.7459	0.3329
	Cold Target	mean	[0.6226]	**[0.6225]**	—	[0.6811]	—	—
		std	0.0035	0.0058	—	0.0082	—	—
		p-val	0.4867	—	—	2.37E-5	—	—
ToxCast	Warm	mean	**[0.4049]**	[0.4092]	—	—	—	—
		std	0.0011	0.0013	—	—	—	—
		p-val	—	0.0012	—	—	—	—
	Cold Drug	mean	**[0.4447]**	[0.4448]	—	—	[0.4550]	[0.4682]
		std	6.37E-4	3.3E-4	—	—	0.0012	0.0036
		p-val	—	0.4343	—	—	1.03E-6	4.42E-5
	Cold Drug Cluster	mean	[0.4480]	**[0.4476]**	—	—	[0.4509]	[0.4566]
		std	0.0006	0.0014	—	—	9.53E-4	0.0025
		p-val	0.3264	—	—	—	0.0048	1.82E-4
	Cold Target	mean	**[0.4794]**	[0.4896]	—	—	—	—
		std	0.0089	0.0120	—	—	—	—
		p-val	—	0.1133	—	—	—	—

Table 5.2: The regression performance across the datasets measured in Concordance Index (larger is better), averaged across independent repetitions of CV. Similar to Table 5.1, the mean CI are enclosed in square brackets; sample standard deviations are also reported. One-sided two-sample t-tests are conducted against best PADME models. The blue values are insignificantly smaller (worse) ($p > 0.05$) than the boldfaced values, while the orange ones are insignificantly larger (better) ($p < 0.95$). The uncolored ones are significantly worse than the boldfaced values.

			Concordance Index					
Dataset	Cross Validation Splitting type	Value Type	PADME-ECFP	PADME-GraphConv	SimBoost	KronRLS	Compound-Only ECFP	Compound-Only GraphConv
Davis	Warm	mean	[0.9034]	**[0.9040]**	[0.8871]	[0.8758]	—	—
		std	0.0020	0.0012	5.97E-4	0.0015	—	—
		p-val	0.2780	—	1.35E-7	2.97E-11	—	—
	Cold Drug	mean	**[0.7120]**	[0.7099]	—	[0.6924]	[0.7027]	[0.6668]
		std	0.0026	0.0139	—	0.0117	0.0162	0.0187
		p-val	—	0.3801	—	0.0042	0.1361	0.0026
	Cold Drug Cluster	mean	**[0.7238]**	[0.7190]	—	[0.6800]	[0.6994]	[0.6828]
		std	0.0094	0.0096	—	0.0421	0.0265	0.0215
		p-val	—	0.2208	—	0.0254	0.0547	0.0047
	Cold Target	mean	**[0.8538]**	[0.8428]	—	[0.8075]	—	—
		std	0.0034	0.0031	—	0.0027	—	—
		p-val	—	3.70E-4	—	8.45E-9	—	—
Metz	Warm	mean	**[0.8065]**	[0.7931]	[0.7944]	[0.7485]	—	—
		std	0.0012	0.0016	6.92E-4	6.98E-4	—	—
		p-val	—	3.49E-7	5.05E-7	3.71E-11	—	—
	Cold Drug	mean	**[0.7432]**	[0.7384]	—	[0.7092]	[0.7110]	[0.7197]
		std	0.0021	0.0013	—	0.0021	0.0033	0.0037
		p-val	—	0.0018	—	5.63E-10	2.60E-7	6.11E-6
	Cold Drug Cluster	mean	**[0.7158]**	[0.7132]	—	[0.6818]	[0.6878]	[0.6966]
		std	0.0048	0.0014	—	0.0037	0.0018	0.0026
		p-val	—	0.1485	—	1.02E-6	2.82E-5	9.20E-5
	Cold Target	mean	[0.6961]	**[0.7099]**	—	[0.6470]	—	—
		std	0.0076	0.0031	—	0.0048	—	—
		p-val	0.0058	—	—	7.82E-10	—	—
KIBA	Warm	mean	[0.8577]	**[0.8616]**	[0.8405]	[0.7831]	—	—
		std	0.0011	0.0014	1.35E-4	3.26E-4	—	—
		p-val	0.0028	—	3.60E-5	2.59E-7	—	—
	Cold Drug	mean	**[0.7742]**	[0.7524]	—	[0.6890]	[0.7405]	[0.7356]
		std	0.0011	0.0032	—	0.0014	0.0027	0.0023
		p-val	—	1.64E-4	—	9.49E-11	3.05E-7	2.66E-8
	Cold Drug Cluster	mean	**[0.7465]**	[0.7190]	—	[0.6654]	[0.7074]	[0.7068]
		std	0.0019	0.0030	—	0.0038	0.0034	0.0023
		p-val	—	9.77E-6	—	5.43E-9	1.37E-7	9.90E-9
	Cold Target	mean	[0.7684]	**[0.7687]**	—	[0.7122]	—	—
		std	0.0020	0.0029	—	0.0045	—	—
		p-val	0.4210	—	—	1.82E-6	—	—
ToxCast	Warm	mean	[0.7908]	**[0.7963]**	—	—	—	—
		std	0.0041	8.62E-4	—	—	—	—
		p-val	0.0380	—	—	—	—	—
	Cold Drug	mean	[0.7196]	**[0.7329]**	—	—	[0.6692]	[0.6414]
		std	0.0062	0.0027	—	—	0.0041	0.0068
		p-val	0.0082	—	—	—	1.25E-8	2.18E-7
	Cold Drug Cluster	mean	[0.7161]	**[0.7290]**	—	—	[0.6440]	[0.6192]
		std	0.0012	0.0032	—	—	0.0045	0.0061
		p-val	0.0010	—	—	—	3.27E-9	1.20E-8
	Cold Target	mean	[0.6752]	**[0.6979]**	—	—	—	—
		std	0.0108	0.0116	—	—	—	—
		p-val	0.0144	—	—	—	—	—

Table 5.3: The regression performance across the datasets measured in R^2 (larger is better), averaged across independent repetitions of CV. Similar to Table 5.1, one-sided two-sample t-tests are conducted against best PADME models. The blue values are insignificantly smaller (worse) ($p > 0.05$) than the boldfaced values, while the orange ones are insignificantly larger (better) ($p < 0.95$). The uncolored ones are significantly worse than the boldfaced values.

			R^2					
Dataset	Cross Validation Splitting type	Value Type	PADME-ECFP	PADME-GraphConv	SimBoost	KronRLS	Compound-Only ECFP	Compound-Only GraphConv
Davis	Warm	mean	**[0.7652]**	[0.7614]	[0.7031]	[0.5801]	—	—
		std	0.0029	0.0031	0.0023	0.0077	—	—
		p-val	—	0.0402	2.79E-10	2.42E-10	—	—
	Cold Drug	mean	**[0.1439]**	[0.0851]	—	[0.0478]	[0.1358]	[0.0612]
		std	0.0335	0.0499	—	0.0592	0.0446	0.0606
		p-val	—	0.0325	—	0.0047	0.3762	0.0178
	Cold Drug Cluster	mean	**[0.2253]**	[0.1631]	—	[0.0540]	[0.1174]	[0.0991]
		std	0.0340	0.0520	—	0.0931	0.1172	0.0321
		p-val	—	0.0304	—	0.0024	0.0545	1.57E-4
	Cold Target	mean	**[0.5915]**	[0.5741]	—	[0.4393]	—	—
		std	0.0087	0.0127	—	0.0064	—	—
		p-val	—	0.0195	—	2.02E-9	—	—
Metz	Warm	mean	**[0.6654]**	[0.5961]	[0.6323]	[0.3355]	—	—
		std	0.0027	0.0153	0.0020	6.15E-4	—	—
		p-val	—	2.1E-4	2.56E-8	1.18E-10	—	—
	Cold Drug	mean	**[0.4477]**	[0.3813]	—	[0.3285]	[0.3471]	[0.3430]
		std	0.0024	0.0145	—	0.0084	0.0241	0.0114
		p-val	—	2E-4	—	2.59E-8	3.41E-4	8.93E-6
	Cold Drug Cluster	mean	**[0.3390]**	[0.2779]	—	[0.2416]	[0.2457]	[0.2241]
		std	0.0183	0.0187	—	0.0080	0.0158	0.0099
		p-val	—	4E-4	—	3.93E-5	1.46E-5	7.22E-6
	Cold Target	mean	**[0.3185]**	[0.2562]	—	[0.1130]	—	—
		std	0.0214	0.0207	—	0.0195	—	—
		p-val	—	8E-4	—	6.30E-8	—	—
KIBA	Warm	mean	[0.7449]	**[0.7560]**	[0.7007]	[0.4128]	—	—
		std	0.0084	0.0032	0.0013	3.34E-4	—	—
		p-val	0.0356	—	2.92E-6	8.54E-8	—	—
	Cold Drug	mean	**[0.5093]**	[0.4352]	—	[0.3266]	[0.4599]	[0.4312]
		std	0.0057	0.0270	—	0.0048	0.0039	0.0054
		p-val	—	0.0051	—	3.57E-9	9.92E-6	1.93E-7
	Cold Drug Cluster	mean	**[0.2948]**	[0.2793]	—	[0.2215]	[0.3267]	[0.2887]
		std	0.0687	0.0463	—	0.0070	0.0140	0.0042
		p-val	—	0.3620	—	0.0611	0.7878	0.4354
	Cold Target	mean	[0.4715]	**[0.4734]**	—	[0.3631]	—	—
		std	0.0051	0.0100	—	0.0136	—	—
		p-val	0.3752	—	—	1.14E-5	—	—
ToxCast	Warm	mean	—	—	—	—	—	—
		std	—	—	—	—	—	—
		p-val	—	—	—	—	—	—
	Cold Drug	mean	—	—	—	—	—	—
		std	—	—	—	—	—	—
		p-val	—	—	—	—	—	—
	Cold Drug Cluster	mean	—	—	—	—	—	—
		std	—	—	—	—	—	—
		p-val	—	—	—	—	—	—
	Cold Target	mean	—	—	—	—	—	—
		std	—	—	—	—	—	—
		p-val	—	—	—	—	—	—

[a] We did not report R^2 for ToxCast because of its imbalanced/skewed nature.

teraction strength. We were also able to show the superiority of PADME over the Compound-Only DNN models, both in applicability of cold-target scenario and overall performance, which might suggest the improvement introduced by protein-specific descriptors (PSC in this paper). Furthermore, the results presented might be an understatement of the real performance of PADME in cold-drug and cold-target scenarios, as the training and validation sets for hyperparameter searching are randomly split, resulting in a set of hyperparameters that suit well for randomly split CV folds, but perhaps not for cold-drug and cold-target folds. This deliberately unfair comparison shows the robustness of the PADME models.

5.1.1 Applicability Domain

Applicability Domain (AD) is an important issue in considering the usage of QSAR models, because every QSAR model has limitations and cannot be applied to all possible inputs. Conceptually, AD defines the region of “normal” objects in the chemical space, for which the QSAR model can be applied and get reliable predictions [43, 30]. [43] categorizes different types of approaches for estimating AD, including ranges in the descriptor space, geometrical methods, distance-based methods, probability density distribution, and range of the response variable.

Some approaches for AD estimation are widely used but are not applicable to PADME, like the standard deviation of ensemble predictions, or the bounding box method, or in general descriptor space analysis [50, 43]. In particular, descriptor space analysis is not applicable because the input of PADME does not involve chemical descriptors obtained through feature engineering, like Dragon descriptors. Furthermore, unlike typical QSAR models which only take compounds as input, PADME also has protein information as part of the input, adding a lot more complexity into AD estimation, like measuring the distance in the distance-based methods. Unlike conventional QSAR methods which only need to calculate the distance between compounds, for PADME, we have to consider the distance between compound-protein (drug-target) pairs, but there is no widely accepted way of measuring the distance between those pairs.

Thus, we decided to simply use the range of the response variable to define AD, which is a common approach for regression models [30, 43]. A natural method is to use mean and standard deviation, but since DTI datasets are often highly skewed (like the Davis dataset), using mean and standard deviation does not give reasonable AD range estimations. For example, in 5-fold CV of Davis dataset, for one iteration, the training folds have range $[5.0, 10.796]$, but the mean is 5.398, and the standard deviation is 0.8506. If we deem a test DT pair with response values lying within $(mean - k * std, mean + k * std)$ as being inside AD, where k is some constant, k must be very large to encompass the right end of the training set value range, but that would make the left end of the AD range too small to be realistic.

Instead, we propose the following simple method. Let min denote the minimum value of response variable y in the training set, and max denote the maximum value in the training set, and the $rangesize = max - min$. We define the AD to be $(min - 0.15 * rangesize, max + 0.15 * rangesize)$. If a test drug-target pair has response value (experimental measurement) outside this range, we deem it as outside of AD. If the experimental measurement is unknown, we use the predicted response value as an approximation to the true response. Fig. 5.1 demonstrates that, in an iteration of CV of Davis dataset, the predicted response values in the validation fold lie within the aforementioned range calculated from the training folds. So we know the elements in the validation fold belong to AD, even if the true response values of the validation fold are unknown. Specifically, the range of response values of the training folds is $[5.0, 10.796]$, so the range size is 5.796. The AD is thus $[5 - 0.15 * 5.796, 10.796 + 0.15 * 5.796]$, which equals $[4.131, 11.665]$. The range of predicted values in the validation fold is $[4.558, 10.123]$, so all the predicted values lie in the AD range, thus all validation fold elements are in the AD.

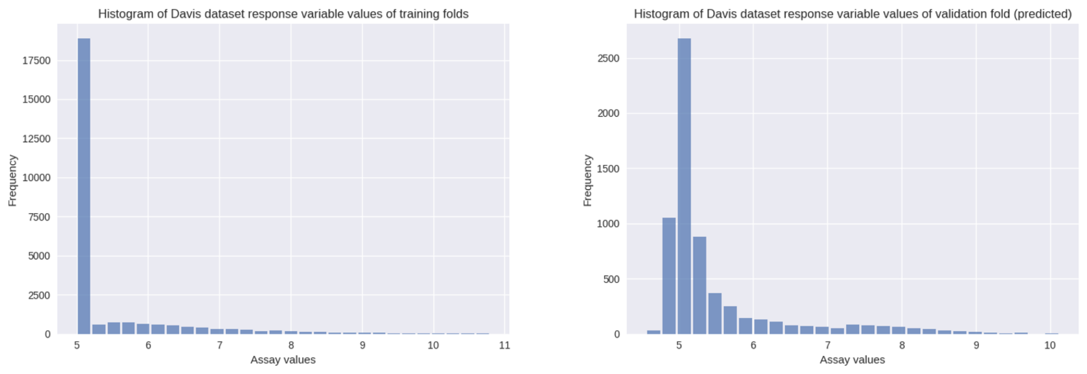

Figure 5.1: The histograms of response variable values of the training folds and the predicted response variable values of the validation fold of a CV iteration of the Davis dataset. The range of response values of training folds is $[5.0, 10.796]$, while the range of predicted values in the validation fold is $[4.558, 10.123]$, demonstrating the validation fold elements are within the AD even if we don't know the true values of the response variable.

5.2 Qualitative Results

We used plots to visualize the prediction performance, so we can assess the results qualitatively.

Fig. 5.2 presents the predicted values (by PADME-ECFP) VS true values for each dataset. For each panel(row) in the figure, there is a scatter plot, and a contour plot (the darkness of the color represents density of data points) with univariate density curves on the margins. Both plots in each row are plotted from the exact same data. Figure 5.2(d) is an exception, it includes a hexagon plot instead of a contour plot, because the contour plot fails to show anything (but the density curves on the margins are plotted), possibly because the data points of ToxCast are too concentrated to be shown correctly on the contour plot, as can be observed from the hexagon plot. To help visualize ToxCast better, we added a Figure 5.3 which is a scatter plot of the same data as Figure 5.2(d), with histograms on the margins.

Clearly, all datasets except Metz data are very concentrated at some values.

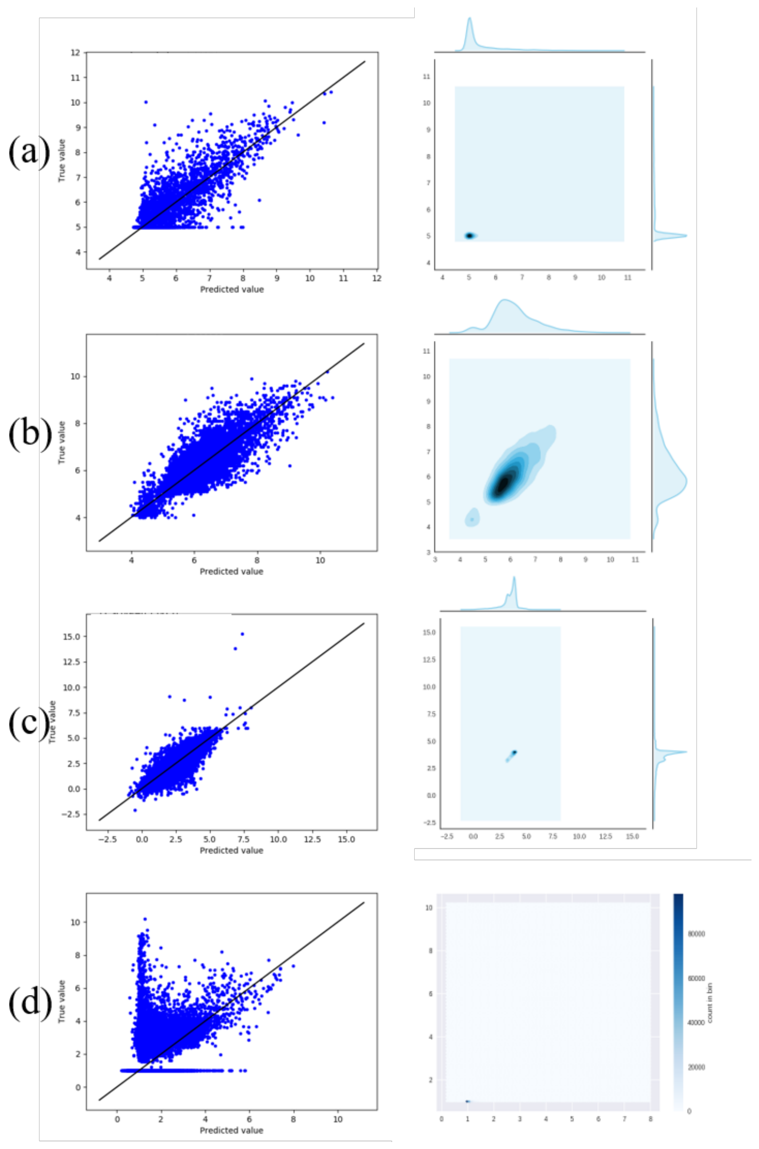

Figure 5.2: Scatter plot and contour plot of predicted VS true values across all datasets. The panels **a, b, c** and **d** correspond to **Davis, Metz, Kiba** and **ToxCast** datasets, respectively. The axes in the two plots of the same panel are the same, and both plots are generated from the same data. The diagonal lines in the scatter plots are the reference lines where *predicted* = *true value*.

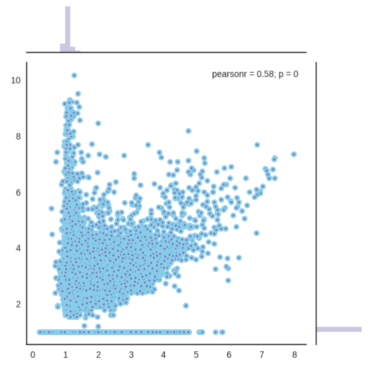

Figure 5.3: ToxCast data scatter plot with marginal histograms, generated from the same data as Figure 5.2(d)

Because the concentration of Davis and ToxCast datasets pose problems in visualizing the prediction performances on them, we decided to plot the scatter plots, contour plots and hexagon plots of the true active and true inactive data points separately for those datasets (Figs. 5.4 and 5.5). From Fig. 5.4 we can see the Davis dataset was predicted pretty well on both true active and inactive values, Fig. 5.4 (a) shows a nice pattern of correspondence between predicted VS true values on the active datapoints, while Fig. 5.4 (b) presents true inactive values, in which the hexagon plot shows a high concentration of predicted values close to the true values. As reflected in Fig. 5.5 (a), the model fitted on the ToxCast dataset was strongly influenced by inactive values, and the prediction performance for the true active datapoints was not very good, but the true inactive datapoints were predicted to concentrate around the true values (shown in the hexagon plot in Fig. 5.5 (b)), which might explain why its quantitative analysis results were decent.

To tackle the imbalanced dataset problem in ToxCast, we tried to train a model on oversampled dataset and measured its performances. In short, it performs worse than expected.

5.2.1 Analysis on the oversampled ToxCast dataset

We tried to oversample the ToxCast dataset to balance the number of active/inactive observations to boost the performance of the models. Oversampling is a technique that increases the samples of the minority class by randomly sampling the minority class samples in addition to the existing samples, such that the new dataset is more balanced, enabling the machine learning model to learn in a "healthier" way [64]. Compared to other ways to balance the dataset like undersampling, oversampling was found to be superior in convolutional neural networks [5]. Often the number of inactive samples equals the number of active ones after oversampling.

Because the 672 assays of the ToxCast dataset were divided into 61 measurements, each drug-target pair can have multiple non-null observations. Even for those drug-target pairs (only a small fraction, less than 20% of total pairs) with some active measurements, it is still

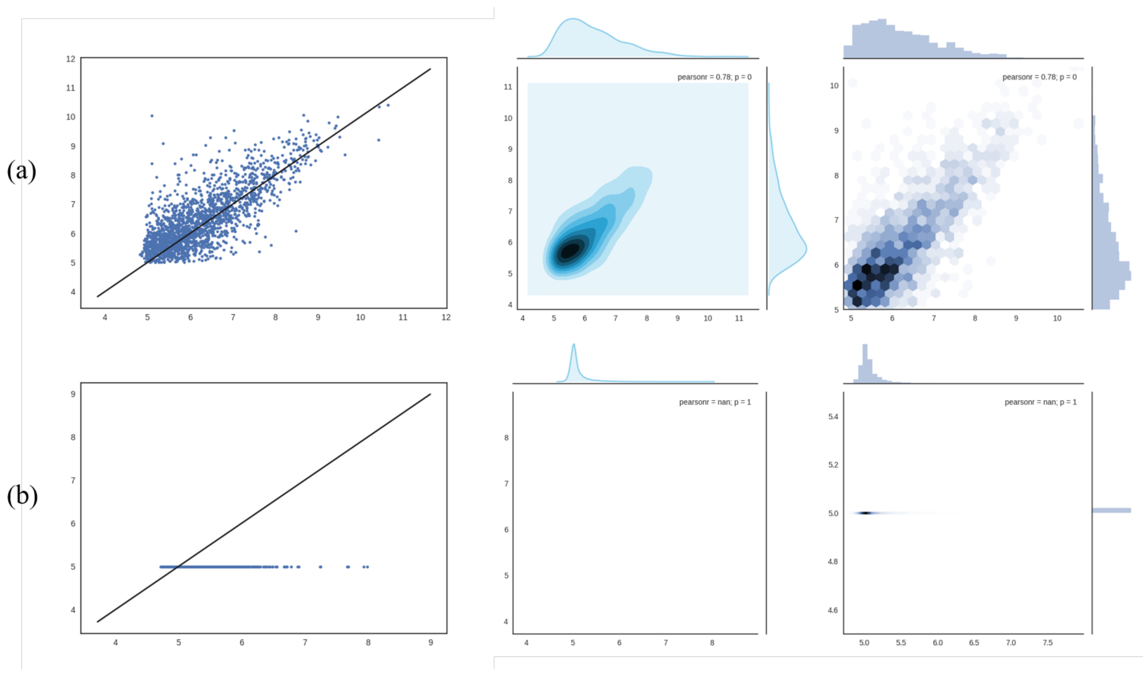

Figure 5.4: Plots for Davis dataset predicted value VS true value. Panel (a) corresponds to the true active values, while panel (b) corresponds to true inactive values. Similar to figure 5.2, all plots in the same panel are plotted from the same data.

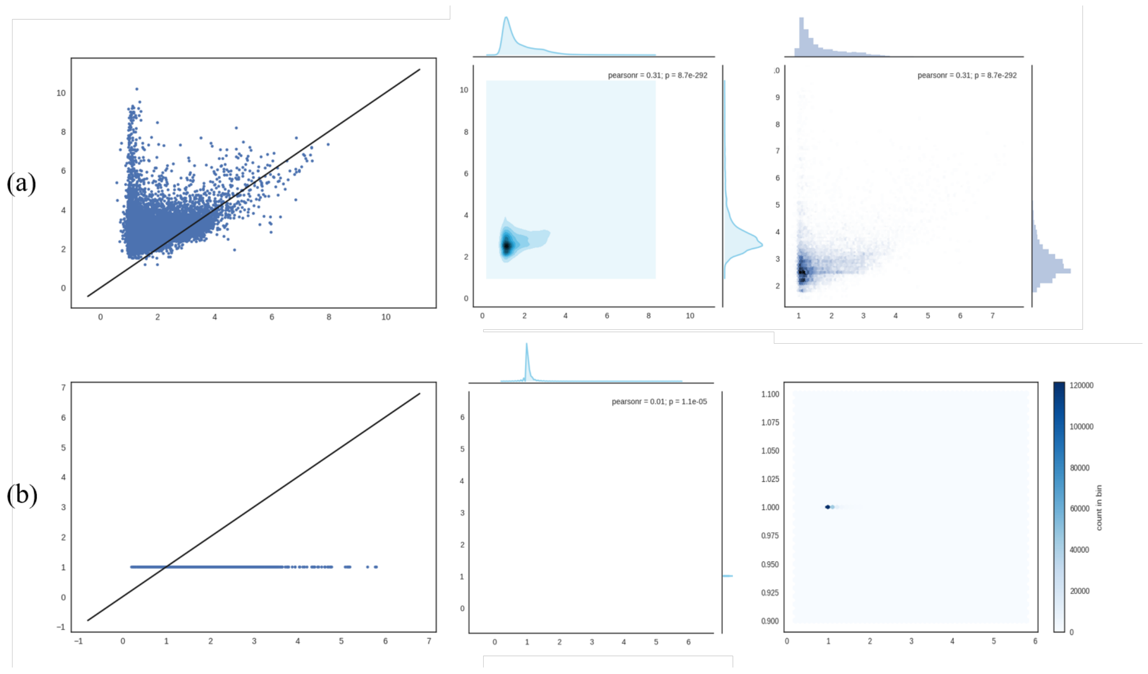

Figure 5.5: Similar to Figure 5.4, plots for the ToxCast dataset. Panel (b) uses a different hexagon plot from (a), because that form of hexagon plot on panel (b) does not show properly.

very likely that most of its non-null measurements are inactive, so making half of the total measurements across all drug-target pairs active is completely distorting the dataset and infeasible. Hence, we define “active pair” as a drug-target pair that has at least one active measurement, we also define “inactive pair” as a pair that has all its non-null measurements inactive, and we randomly sampled the active pairs such that the total number of active pairs equals inactive pairs.

We used the oversampled dataset for CV. Each training fold was split from oversampled dataset without further processing, while the validation folds have their duplicate drug-target pairs removed, so that we are essentially training on oversampled data and testing on the original data. To prevent each oversampled pair from appearing in both training and validation sets, we put all repeated instances of a pair into the same fold.

Tables 5.4 and 5.5 presents the 5-fold CV results of the ToxCast PADME models with oversampling, while for reference, alongside them are the CV results of ToxCast PADME models without oversampling (extracted from Tables 5.1 and 5.2). All the hyperparameters used for training DNN on oversampled datasets are the same as those already found for training on original datasets, so the results presented for oversampled data may not be optimal.

Table 5.4: CV Results of PADME models on ToxCast original and oversampled datasets, measured in RMSE (smaller is better). Boldfaced numbers indicate the better results between the models trained on original VS oversampled datasets.

		RMSE			
		PADME-ECFP		PADME-GraphConv	
Dataset	Cross Validation Splitting type	Original dataset	Oversampled dataset	Original dataset	Oversampled dataset
ToxCast	Warm(random)	**0.4056**	0.4887	**0.4078**	0.4663
	Cold Drug	**0.4448**	0.5159	**0.4450**	0.5057
	Cold Target	**0.4870**	0.5381	**0.4944**	0.6554

Table 5.5: Similar to the last table, but performance measured in Concordance Index (larger is better).

		Concordance Index			
		PADME-ECFP		PADME-GraphConv	
Dataset	Cross Validation Splitting type	Original dataset	Oversampled dataset	Original dataset	Oversampled dataset
ToxCast	Warm(random)	**0.7965**	0.7717	**0.7987**	0.7781
	Cold Drug	**0.7206**	0.6628	**0.7286**	0.6887
	Cold Target	0.6848	**0.7008**	**0.6905**	0.6735

The effect of oversampling is surprising. Clearly, oversampling overall has a negative influence on the prediction performance. We speculate that, in oversampling, the repetition of active pairs reduced the information diversity in some of the training folds, because some inactive pairs that could have been selected as training samples were “squeezed” into the validation set due to the oversampling of active pairs. A repeated (active) pair which must have all its repeated instances placed in one fold can potentially “squeeze” several unique pairs out of the fold, possibly this phenomenon cancels out the positive effects of oversampling. Because imbalanced dataset is a very common scenario and important problem in virtual screening, more studies of ToxCast regarding this issue might be needed.

5.2.2 Discussion of the results on the ToxCast dataset

So why does PADME perform well on Davis, Metz and KIBA datasets, but not so satisfactorily on the ToxCast dataset? We think it might be related to the nature of the ToxCast dataset itself. The ToxCast dataset not only contains a much larger variety of proteins (unlike the other 3 datasets which only contain kinase inhibitors), but it also has a much larger number of assays (measurement endpoints), which are often quite different from each other. Though we only selected the assays with single intended targets, many of those assays are cell-based (for example, OT_AR_ARSRC1_0480), which could introduce some more complexities in addition to the drug-target interaction, due to the intricacies of biochemical processes in cells. These challenges might be some reasons why, to our knowledge, previous non-docking researches[2] on drug-target interaction prediction containing protein information as input did not use this dataset [17, 39, 59, 38], though other kinds of researches did [7, 29, 26].

The challenges with the ToxCast dataset, including its large number of measurement endpoints and the imbalanced dataset problem, should be investigated in future work, as it is an important objective to build a more general-purpose DTI prediction model that handles a larger variety of input proteins, compounds and measurement endpoints. We believe that, based on our work, future researches will either make improvements on the ToxCast dataset or deem it as a great challenge for DTI prediction models involving protein information, and our results presented here, though not ideal, is of reference value to the community.

5.3 Case Studies

In addition to the quantitative and qualitative evaluations shown above, we performed some case studies to further validate the predictions of PADME by investigating the compounds predicted to interact strongly with selected target proteins.

[2]Docking is a molecular modelling technique, which simulates the 3D interaction between compounds (ligands) and proteins, to predict the position and orientation of the compound when it is bound to the protein.

We focused on the androgen receptor (AR), for which alterations of functions are associated with prostate [63] and breast cancers [34]. We are also very familiar with the protein, possessing the equipment and skills to test its interactions with different compounds in the wet-lab experiments.

We used all compounds in the datasets used in this thesis, together with all the compounds in US National Cancer Institute human tumour cell line anticancer drug screen data (NCI60), totalling more than 100000, and AR as the only target protein. NCI60 dataset records the in vitro drug response of cancer cell lines [47]. For prediction, we used PADME-ECFP and PADME-GraphConv trained on the whole ToxCast dataset, then took the average of their predictions, we call this averaged model PADME-Ensemble. The reason we chose the ToxCast dataset is that its endpoints are the most suitable for calculating AR binding affinity or antagonistic effects, and the ToxCast dataset has the most diverse set of compounds and proteins.

We conducted 2 different case studies: a simple one which predicts compounds' AR binding affinity, and a much more complicated one, predicting compounds' AR antagonist effects.

5.3.1 Compounds predicted with high binding affinity to AR

There are many different assays in ToxCast, some are cell-based, while some are cell-free. Cell-based assays are much more complicated than their cell-free counterparts, since the results of cell-based assays might involve some intricate biochemical reactions in the cells. Thus, we used the assay NVS_NR_hAR, a cell-free assay measuring the binding affinity between Human Androgen Receptor (AR) and ligands (please refer to the Section 5.3.2 for details), to examine the efficacy of PADME's predictions.

From the predictions of PADME-Ensemble, we selected the prediction results corresponding to NVS_NR_hAR, and then sorted the predicted values in a descending order. Due to the transformations we performed in Section 4.2, the larger the number (higher in the sorted list), the stronger the binding affinity. Next, we filtered out those compounds in the predicted list that have appeared in the ToxCast dataset, or had a large Tanimoto similarity (Jaccard distance) with some compounds in ToxCast, calculated using rdkit fingerprint. We then did a search on PubChem database [21] for the top 30 compounds predicted to bind strongly with AR.

The top 30 (and beyond) compounds all shared highly similar structure with androgen, so it is quite possible that most of them are able to bind strongly with AR. After a stringent search on PubChem, we confirmed that 4 of them are active, which is reflected in patents, bioassay results, or research papers. The other compounds in top 30 are also possibly active, but since there are no direct evidence from PubChem, we take the conservative approach and do not consider them here.

The 4 compounds' PubChem CID are: 88050176, 247304, 9921701, 220507. Fig. 5.6 shows their 2D images.

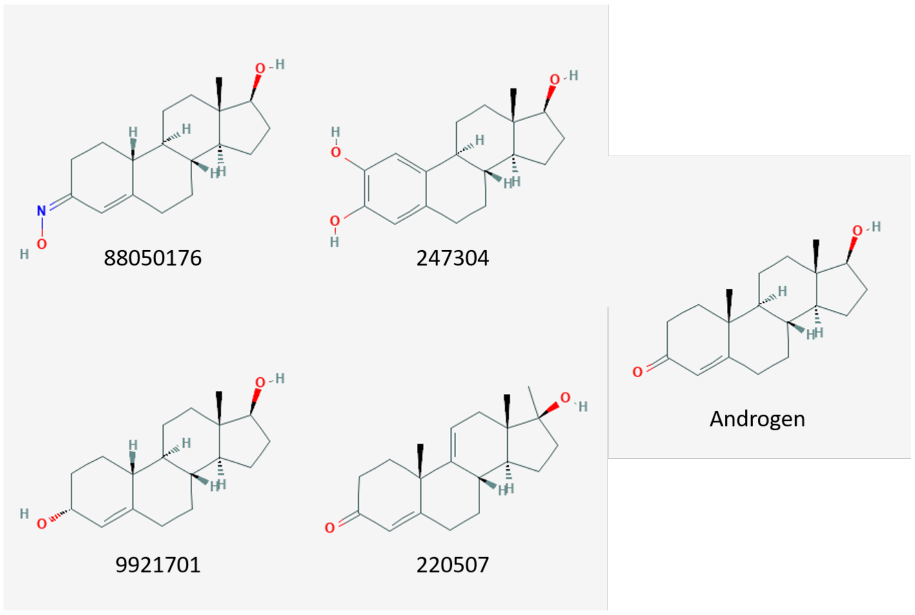

Figure 5.6: The 4 compounds from top 30 predictions that are confirmed to bind strongly with AR. The numbers are their corresponding PubChem CIDs. On the right side is the 2d representation of testosterone, the major androgen. The images are downloaded from PubChem website.

Obviously, the compounds are all very similar to androgen, since NVS_NR_hAR is a very simple assay, the model learns from the dataset that analogs of androgen tend to bind strongly with AR. This shows that the prediction results of PADME are effective in drug discovery.

5.3.2 Predicting compounds with strong AR antagonist effects

Based on this, we tried to take one step further to do a more interesting task: calculating the AR antagonist effect of compounds based on the predictions produced by PADME. Because there are 61 outputs in PADME models trained on ToxCast data, we had to propose a set of coefficients to calculate a composite AR antagonist score (details to be followed) from the averaged prediction results, for which we expect the compounds with higher scores would show stronger anticancer activity in AR-related cell lines in NCI60 dataset. We then ordered the AR antagonist scores in a descending order.

Compared to those predicted to bind strongly with AR, the predicted list of AR antagonistic compounds have much more diverse structures. However, their results are not well aligned with our expectations. This is not caused by PADME, which captures the patterns in the training data faithfully and shows it in the test set. It is the results on the true dataset that are different from our assumptions.

There are two major challenges in this process: the formula we used for calculating AR antagonist score is based on assumptions, the existence of cell-based assays is also a possible source of problem; our expectation that AR antagonistic compounds should perform selectively on some known AR-related cancer cell lines might deviate from the truth, or AR influences many types of cancers in different ways from what we knew, like suggested in [36]. Tackling these two challenges is a task that could require years or decades of work by the community.

Formula of AR antagonist score

The AR antagonist score was constructed as a linear combination of PADME predictions for assays involving AR:

$$\begin{aligned}
Antagonist\_score = {} & NVS\_NR\_hAR + (\frac{1}{2}TOX21\_AR\_LUC\_MDAKB2\_Antagonist \\
& + \frac{1}{2}TOX21\_AR\_BLA\_Antagonist\_ratio) \\
& - \{(\frac{1}{2}OT\_AR\_ARSRC1\_0480 + \frac{1}{2}OT\_AR\_ARSRC1\_0960) \\
& + ATG\_AR\_TRANS\_up + (\frac{1}{3}OT\_AR\_ARELUC\_AG\_1440 \\
& + \frac{1}{3}TOX21\_AR\_LUC\_MDAKB2\_Agonist \\
& + \frac{1}{3}TOX21\_AR\_BLA\_Agonist\_ratio)\}
\end{aligned} \tag{5.1}$$

Table 5.6: ToxCast assays description

Stage	Assays	Tissue/Cell lines
Receptor binding	NVS_NR_hAR	extracted gene-proteins from LnCAP in a cell-free assay
Cofactor recruitment	OT_AR_ARSRC1_0480	HEK293T, a human kidney cell line
	OT_AR_ARSRC1_0960	
Gene transcription	ATG_AR_TRANS_up	HepG2, a human liver cell line
Gene expression	OT_AR_ARELUC_AG_1440	CHO-K1, a Chinese hamster ovary cell line
	TOX21_AR_LUC_MDAKB2_Agonist	MDA-kb2, a human breast cell line
	TOX21_AR_LUC_MDAKB2_Antagonist	
	TOX21_AR_BLA_Agonist_ratio	HEK293T, a human kidney cell line
	TOX21_AR_BLA_Antagonist_ratio	

Some of the assays in Table 5.6 are cell-free assays, while some are cell-based assays. It might not have been a good idea to combine them together using simple arithmetic operations. This concern is justified in the subsequent steps.

Analysis of AR antagonist score on true dataset

We investigated whether the compounds predicted with strong AR antagonist effects could inhibit proliferation of related cancer cell lines, and whether this inhibition effect is specific to those AR-related cell lines.

Among the measurements in NCI60, we only considered the logGI50 values, which are real numbers. The smaller the logGI50 value, the more active the compound is in suppressing the growth of cancer cells. GI stands for "Growth Inhibition".

Based on previous researches showing the relationship between AR and breast cancer [34], we hypothesized that the AR antagonist (or agonist) scores should have a strong correlation with logGI50 values of the breast cancer cell lines and a lower correlation with logGI50 values of other cancer cell lines. We also assumed that the ordered list of compounds ranked by AR antagonist scores would agree well with the ordered compound list ranked by logGI50 values in breast cancer cell lines, while agreeing poorly with the compound list ranked by logGI50 values in other cancer cell lines.

To test the validity of our assumptions, we selected the compounds that appear in both ToxCast and NCI60 datasets, calculated the AR antagonist scores of those compounds using the observed values in the ToxCast dataset, plotted the antagonist scores against the negative logGI50 values (taking negative to make the two values positively correlated) of the corresponding compounds in NCI60 dataset, and also measured the ranking agreement between the AR antagonist scores and logGI50 values using quantitative methods. Some results are shown in Fig. 5.7 and Table 5.7.

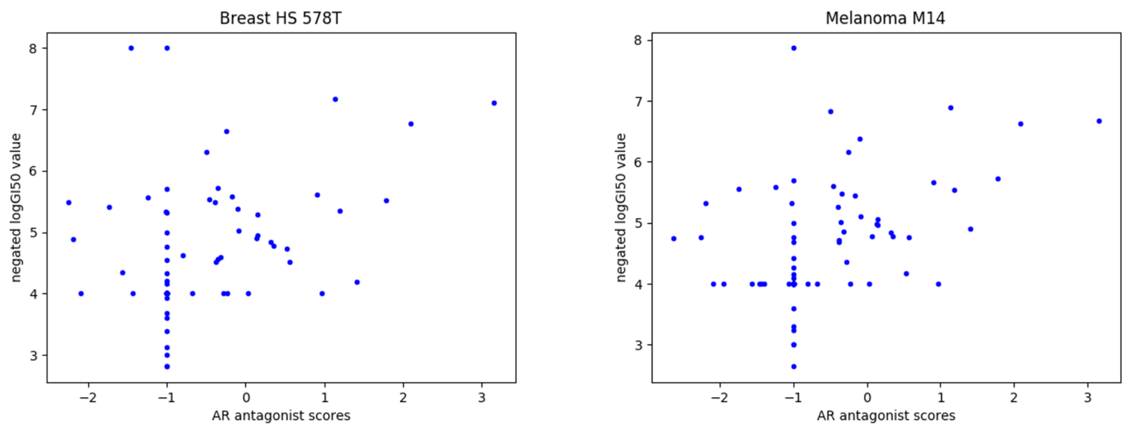

Figure 5.7: Scatter plot of AR antagonist scores VS negative logGI50 values for HS 578T cell line in breast cancer and M14 cell line in Melanoma. Each dot corresponds to a compound.

Table 5.7: Agreement between logGI50 values and AR antagonist scores in different cell lines.

Panel	Cell line	Number of valid compounds	Concordance Index	NDCG	Spearman's correlation	p-value for Spearman's correlation
Prostate	PC-3	68	0.673469	0.821579	0.512713	7.82e-06
Prostate	DU-145	68	0.657856	0.791487	0.441896	1.62e-04
Colon	HCT-116	85	0.675707	0.839728	0.476324	4.07e-06
Melanoma	M14	82	0.669832	0.855436	0.45758	1.55e-05
Breast	MDA-MB-468	31	0.603524	0.80477	0.29136	0.11177
Breast	HS 578T	68	0.60805	0.84215	0.30980	0.01014
Breast	BT-549	61	0.67681	0.82405	0.50380	3.49e-05
Breast	MCF7	66	0.64341	0.84066	0.40496	0.000744
Breast	T-47D	65	0.68801	0.77054	0.53885	3.65e-06
Breast	MDA-MB-231/ATCC	65	0.66009	0.78359	0.44498	0.000204

Because AR is strongly related to breast cancer, we assumed there should be a strong positive correlation between the AR antagonist scores and negative logGI50 values. Similarly, since Melanoma is not shown to be related to AR, we assumed there would be a much weaker (or even zero) correlation between AR antagonist scores and negative logGI50 values compared to breast cancer cell lines. However, Fig. 5.7 does not show a discernible difference between Melanoma and breast cancer. Plots of other breast cancer cell lines and other cancer types also gave us similar patterns.

In addition to visual inspection in Fig. 5.7, we used several metrics to measure the ranking agreement between the AR antagonist scores and logGI50 values in Table 5.7, in which Normalized Discounted Cumulative Gain (NDCG) is a metric often used in data mining to examine the quality of ranking [18]. For NDCG we used the classical logarithmic discount, and chose zero-adjusted negative logGI50 as the relevance score of the compound, such that it starts from 0 and increases with stronger relevance. NDCG values have the range [0, 1], the higher the value is, the better the ranking quality. For Concordance Index, we used the compound list ranked by AR antagonist scores as the predicted ranking, while the compound list ranked by logGI50 value corresponds to the true ranking. Spearman's correlation is calculated on ranks only, so it is also a metric measuring the agreement between two rankings.

The colon cancer, Melanoma and the presented prostate cancer cell lines are not known to be related to AR, so they are expected to have lower agreements between logGI50 values and AR scores. However, in Table 5.7, it can be seen that all metrics are quite similar across multiple cell lines. (More cell lines are tested on, but only a selected subset is presented in the thesis. All the cell lines yielded similar results.)

We also carried out the same experiments using AR agonist score that we calculated from the observed dataset. Likewise, there are no notable differences between cell lines. The details are out of the scope of this thesis.

As we can see, contrary to what we expected, the compounds' antagonist effects on AR are not more strongly related to their activity in breast cancer cell lines than to their activity in other cell lines. This phenomenon was also observed with our predicted AR antagonist scores, which shows PADME's faithfulness, but we don't present the details here, since they are less convincing than scores calculated from true values. This suggests that either the AR is related to a wide range of cancers, or the true relationship between AR and prostate cancer is far from a linear relationship, or the assumptions used in making the AR antagonist score formula causes the problem. We suggest this as an issue for future research.

Analysis of AR antagonist score on true dataset: t-tests

We wanted to examine our assumption that the compounds ranked high in (both predicted and observed) AR antagonist scores generally have higher activities in suppressing the growth of breast cancer cell lines. Since we already found the AR antagonist scores to have similar relationship with all cell lines, this assumption actually degenerates into a general toxicity problem. Nevertheless, we decided to take a look.

This time, we start with the predicted scores, to see whether those predicted high-ranking antagonistic compounds are truly inhibiting cancer cell lines.

We used only the Breast Cancer panel in the NCI60 dataset, in which we chose 5 out of 6 cell lines, leaving out the one with relatively few observations.

We report the results separately for each cell line. For each of them, we took the top 100, top 1000, top 15000 and all compounds according to the averaged AR antagonist score. We skipped those compounds that were absent from the cell line to ensure that the top-n set contains n compounds. Say we have a top-100 list for cell line X, the 100th compound in the list is not in the top 100 in the sorted AR antagonist score list, because there are compounds in AR score list that do not have observations for X.

We calculated the mean and standard deviation of the logGI50 values for those top-n compounds. Table 5.8 presents the results for some breast cancer cell lines. Similar patterns are also observed in other breast cancer cell lines.

Based on the values in Table 5.8, we conducted a series of one sample t-tests. For example, in Table 5.9, the entry corresponding to (100, 1000) is obtained by performing a t-test on top 100 compounds against the top 1000 compounds, in which H_0 is: the top 100 compounds are obtained from the top 1000 randomly. In more formal terms, the mean logGI50 values of the hypothetical group that the top 100 compounds in AR score in BT-549 dataset belong to, equals the mean of that of top 1000 compounds (-5.41672). H_1 is: the top 100 compounds in AR scores are truly more active than the top 1000 compounds. In more formal terms, the mean logGI50 values of the hypothetical group that the top 100

Table 5.8: Mean and standard deviation of logGI50 values in top-n compounds in predicted AR antagonist scores. We only used BT-549 cell line as an example, but similar patterns are observed across all breast cancer cell lines.

Panel	top n	cell line	mean value	standard deviation
Breast	32128	BT-549	-4.679217	0.87416
Breast	15000	BT-549	-4.87648	0.93134
Breast	1000	BT-549	-5.41672	1.40488
Breast	100	BT-549	-6.73576	1.45166

compounds in AR score belong to, is smaller than the mean of that of top 1000 compounds (-5.41672).

Table 5.9: Base top-k compared against other top-k's for BT-549. A one sample t-test is conducted for each filled entry, where H_0 is that both top-k's have identical means, while H_1 is that the base top-k has a smaller mean.

Cell line: BT-549	Top-k to compare against		
Base top-k	1000	15000	All(35159)
15000	—	—	t-score: -25.9409 p-value: 1.8e-145
1000	—	t-score: -12.1603 p-value: 3.96e-32	t-score: -16.6006 p-value: 3.84e-55
100	t-score: -9.04089 p-value: 6.75e-15	t-score: -12.7437 p-value: 6.76e-23	t-score: -14.0958 p-value: 1.05e-25

We can see that the null hypotheses are all strongly rejected, which shows a consistent trend that the compounds with a high predicted AR score tend to be more actively inhibiting the prostate cancer cell lines. Similar results are also obtained on other cell lines in breast cancer, but the tables are not presented here for brevity.

To determine whether such a trend also exists in true rather than predicted data, we performed a similar analysis on the observed values in ToxCast. Similar to the analysis for Table 5.8, we selected the compounds that appeared in both ToxCast and the NCI60 cell lines, calculated their AR antagonist scores using observed values in ToxCast, and did an analysis on top 10 and top 20 compounds. Some of the results are presented in Tables 5.10 and 5.11. While all the breast cancer cell lines in breast cancer have similar results as Table 5.11, which corresponds to BT-549, only Table 5.11 is presented as an example.

Clearly, we can observe that Tables 5.10 and 5.11 are similar to Tables 5.8 and 5.9, top compounds in true AR antagonist scores calculated from observed values also tend to be significantly more active in breast cancer cell lines than lower compounds, similar to their predicted counterparts. This indicates PADME captures the patterns present in the training data.

Table 5.10: Mean and standard deviation of logGI50 values in top-n compounds in true AR antagonist scores calculated from observed data in ToxCasts. We only used BT-549 cell line as an example, but similar patterns are observed across all breast cancer cell lines.

Panel	top n	cell line	mean value	standard deviation
Breast	61	BT-549	-4.69618	1.01828
Breast	20	BT-549	-5.3118	0.80113
Breast	10	BT-549	-5.5317	0.99625

Table 5.11: Base top-k compared against other top-k's for BT-549. A one sample t-test is conducted for each filled entry, where H_0 is that both top-k's have identical means, while H_1 is that the base top-k has a smaller mean.

Cell line: BT-549	Top-k to compare against	
Base top-k	20	68
20	—	t-score: -3.43656 p-value: 0.00138
10	t-score: -0.698 p-value: 0.251406	t-score: -2.65208 p-value: 0.01319

This kind of trend is similar in other cancer types in NCI60 data, so we can only say that the AR antagonist score we proposed shows the general toxicity of a compound.

Wet-lab experiments

Although the AR antagonist score might not work as well as expected, we still decided to test whether the compounds predicted with high AR antagonist effects are truly so. Because we cannot readily conduct the chemical database search (like in Section 5.3.1 where we were finding the compounds that bind strongly to AR) due to insufficient amount of information available, we decided to purchase the compounds and conduct wet-lab experiments on their AR antagonist performance. We chose 38 compounds from the top predictions that were available from the vendors including the National Cancer Institute [51], and 3 of them were confirmed to be active after going through eGFP and PSA (Prostate-Specific Antigen) tests. See Table 5.12 which lists the PSA assay results.

Table 5.12: The confirmed active compounds and their PSA assay results (smaller is better). As a reference, MDZ, a state-of-the-art drug, has PSA value of 0.6928.

ZINC ID	PSA assay result
ZINC8665890	2.566
ZINC3861637	4.860
ZINC4947964	6.113

The size of our candidate compound set used for this study is only around 100000, so the choices are limited. 3 actives out of 38 is still a pretty good performance.

Summary on AR antagonist scores

To summarize, we designed the formula for calculating AR antagonistic score based on the predictions given by PADME, the compounds predicted to have strong antagonist effects on AR indeed showed higher level of activities in NCI60 dataset breast cancer cell lines (and other cancer cell lines), suggesting that PADME has the potential to be applied in drug development. However, contrary to what we believed, the effect of AR seems not to be specific to breast cancer cell lines, so our AR antagonist score degenerates to a general toxicity indicator. This could be caused by the problem in designing the formula of AR antagonist score, not necessarily the problem of PADME; instead, PADME faithfully captures the pattern in the training dataset and shows it in the test dataset. Another possible source of problem is that our understanding of AR's effect on cancer cell lines is not complete enough, like what was proposed in [36].

5.3.3 Summary of Case Studies

In all, the obtained results indicate that PADME is capable of identifying compounds that have the desired simple interactions with the target protein. Although our more complicated AR antagonist effect prediction did not quite work as we hoped, the problem arose from the assumptions in designing the formula and hypothesizing AR's effect on cancer cell lines, not PADME itself.

Chapter 6

Discussion

PADME is a very general and versatile framework, compatible with a large variety of different protein and compound featurization methods. By combining different protein descriptors like PSSM and other molecule featurization schemes like Weave [20], many more variants of PADME can be constructed, whose performances can be compared against each other. In fact, we used the Weave implementation in DeepChem as a molecular featurization method and ran hyperparameter searching on it, but the result was worse than ECFP and GraphConv, and it consumed much more time and memory than the other two, so we did not pursue it any further.

[38] used CNN to learn a latent feature vector to represent the proteins based on its amino acid sequence information, instead of using fixed-rule protein descriptors as the input, and they also used CNN to learn a compound feature vector from SMILES string in a similar way, making the network structure more "symmetrical". They showed a performance similar to PADME, but they did not use cross-validation to get average performances, only running different models on the same test set, which was just 1/6 the size of the whole dataset, so we think there is still much room for improvement in that direction. Nonetheless, it is a very good step towards it.

We only used simple feedforward neural networks in our implementations of PADME from the Combined Input Vector to the output, but other types of Neural Networks might be able to generate better results, like Highway Networks [49], which allows the units in the network to take shortcuts, circumventing the large amounts of layers in some networks.

Pretraining also has the potential to improve our model, but we did not include it, because it might be difficult for the community to compare the real performance of PADME with other models.

Compared to previous models like SimBoost and KronRLS, PADME is not only outperforming them in terms of prediction accuracy, but is more scalable in terms of number of drugs and targets and number of prediction endpoints[1], because both SimBoost and KronRLS rely on similarity matrices and they are only single-task models. In the age of Big Data, this scalability will be a big advantage in virtual high-throughput screening.

We envision PADME and its future derived models to be useful in lots of tasks in medicinal chemistry, which might include toxicity prediction, computer-aided drug discovery, precision medicine, etc. For toxicity prediction, using PADME, scientists can better predict the side-effects of known drugs, or predict the toxicity of a drug under development; in computer-aided drug discovery, such models will greatly narrow down the scope of candidates in a virtual screening process, leaving only very few top candidates to be further simulated or tested; in precision medicine, we believe the model can give physicians better insight based on the protein expression profile of the patient.

[1]As elaborated in Section 3.6, the scalability in the number of drugs and targets might not always be the case, but the scalability in the number of prediction endpoints is.

Chapter 7

Conclusion

To tackle DTI prediction problem from a new angle, we devised the PADME framework that utilizes deep neural networks for this task. PADME incorporates both compound and target protein sequence information, so it can handle the cold-start problem, which most current deep learning-based models for DTI prediction cannot do. Using sequence information as the input makes the model simple and generally applicable. Predicting real-valued endpoints also makes it desirable for problems requiring finer granularity than binary classification.

PADME is the first method to incorporate MGC with protein descriptors into the DTI prediction task, and has been shown to consistently outperform state-of-the-art methods as well as Compound-Only DNN models. Surprisingly enough, PADME based on MGC (GraphConv in our case) does not outperform PADME based on ECFP, which could be due to the difficulty of finding the best set of hyperparameters for MGC. More work is needed to construct a better MGC model or find a better set of hyperparameters for the existing model. PADME is also more scalable than the state-of-the art models for DTI regression task, namely SimBoost and KronRLS, and this advantage might be significant in datasets with lots of compounds/targets and multiple measurement endpoints. Another contribution is the use of the ToxCast dataset in DTI prediction problems with protein information input, which we believe future research should investigate further in addition to the other benchmarking datasets. Our results on the ToxCast dataset suggests it is a greater challenge than we expected.

As a case study, we predicted the binding affinity between compounds and the androgen receptor (AR), a high proportion of the compounds predicted to bind strongly with AR are confirmed through database/literature search. This suggests that PADME has the potential to be applied in drug development, and will likely benefit domains like toxicity prediction, computer-aided drug discovery, precision medicine, etc.

With the compatibility of PADME to different drug molecule and target protein featurization methods, as well as its scalability compared to methods which rely on similarity matrices and have single outputs, we believe that future work could propose more PADME variants that advance the frontier of DTI prediction research.

Bibliography

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Khan Academy. Enzyme regulation. `https://www.khanacademy.org/science/biology/energy-and-enzymes/enzyme-regulation/a/enzyme-regulation`, 2016. Accessed: 2019-02-27.

[3] Han Altae-Tran, Bharath Ramsundar, Aneesh S Pappu, and Vijay Pande. Low data drug discovery with one-shot learning. *ACS central science*, 3(4):283–293, 2017.

[4] Désirée Baumann and Knut Baumann. Reliable estimation of prediction errors for QSAR models under model uncertainty using double cross-validation. *Journal of cheminformatics*, 6(1):47, 2014.

[5] Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106:249–259, 2018.

[6] Dong-Sheng Cao, Qing-Song Xu, and Yi-Zeng Liang. Propy: A tool to generate various modes of Chou's PseAAC. *Bioinformatics*, 29(7):960–962, 2013.

[7] YG Chushak, HW Shows, JM Gearhart, and HA Pangburn. In silico identification of protein targets for chemical neurotoxins using toxcast in vitro data and read-across within the QSAR toolbox. *Toxicology Research*, 7(3):423–431, 2018.

[8] UniProt Consortium et al. UniProt: the universal protein knowledgebase. *Nucleic acids research*, 46(5):2699, 2018.

[9] George E Dahl, Navdeep Jaitly, and Ruslan Salakhutdinov. Multi-task neural networks for QSAR predictions. *arXiv preprint arXiv:1406.1231*, 2014.

[10] Mark Davies, Michał Nowotka, George Papadatos, Nathan Dedman, Anna Gaulton, Francis Atkinson, Louisa Bellis, and John P Overington. ChEMBL web services:

streamlining access to drug discovery data and utilities. *Nucleic acids research*, 43(W1):W612–W620, 2015.

[11] M I Davis, J P Hunt, S Herrgard, P Ciceri, L M Wodicka, G Pallares, M Hocker, D K Treiber, and P P Zarrinkar. Comprehensive analysis of kinase inhibitor selectivity. *Nature Biotechnology*, 29(11):1046–1051, 2011.

[12] H Ding, I Takigawa, H Mamitsuka, and S Zhu. Similarity-based machine learning methods for predicting drug–target interactions: A brief review. *Briefings in Bioinformatics*, 15(5):734–747, 2014.

[13] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232, 2015.

[14] Anna Gaulton, Anne Hersey, Michał Nowotka, A Patrícia Bento, Jon Chambers, David Mendez, Prudence Mutowo, Francis Atkinson, Louisa J Bellis, Elena Cibrián-Uhalte, et al. The ChEMBL database in 2017. *Nucleic acids research*, 45(D1):D945–D954, 2016.

[15] Joseph Gomes, Bharath Ramsundar, Evan N Feinberg, and Vijay S Pande. Atomic convolutional networks for predicting protein-ligand binding affinity. *arXiv preprint arXiv:1703.10603*, 2017.

[16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[17] T He, M Heidemeyer, F Ban, A Cherkasov, and M Ester. Simboost: A read-across approach for predicting drug–target binding affinities using gradient boosting machines. *Journal of Cheminformatics*, 9(1):24, 2017.

[18] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

[19] Jose Jimenez. pyGPGO: Bayesian Optimization for Python.

[20] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: Moving beyond fingerprints. *Journal of computer-aided molecular design*, 30(8):595–608, 2016.

[21] Sunghwan Kim, Jie Chen, Tiejun Cheng, Asta Gindulyte, Jia He, Siqian He, Qingliang Li, Benjamin A Shoemaker, Paul A Thiessen, Bo Yu, et al. PubChem 2019 update: improved access to chemical data. *Nucleic acids research*, 47(D1):D1102–D1109, 2018.

[22] KinBase. The kinase database website. `http://kinase.com/web/current/kinbase/`. Accessed: 2018-02-02.

[23] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[24] Greg Landrum. RDKit: Open-source cheminformatics.

[25] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[26] Jie Liu, Kamel Mansouri, Richard S Judson, Matthew T Martin, Huixiao Hong, Minjun Chen, Xiaowei Xu, Russell S Thomas, and Imran Shah. Predicting hepatotoxicity using toxcast in vitro bioactivity and chemical structure. *Chemical research in toxicology*, 28(4):738–751, 2015.

[27] Junshui Ma, Robert P Sheridan, Andy Liaw, George E Dahl, and Vladimir Svetnik. Deep neural nets as a method for quantitative structure–activity relationships. *Journal of chemical information and modeling*, 55(2):263–274, 2015.

[28] Gerard Manning, David B Whyte, Ricardo Martinez, Tony Hunter, and Sucha Sudarsanam. The protein kinase complement of the human genome. *Science*, 298(5600):1912–1934, 2002.

[29] Kamel Mansouri, Ahmed Abdelaziz, Aleksandra Rybacka, Alessandra Roncaglioni, Alexander Tropsha, Alexandre Varnek, Alexey Zakharov, Andrew Worth, Ann M Richard, Christopher M Grulke, et al. CERAPP: Collaborative estrogen receptor activity prediction project. *Environmental health perspectives*, 124(7):1023, 2016.

[30] Miriam Mathea, Waldemar Klingspohn, and Knut Baumann. Chemoinformatic classification methods and their applicability domain. *Molecular informatics*, 35(5):160–180, 2016.

[31] Andreas Mayr, Günter Klambauer, Thomas Unterthiner, and Sepp Hochreiter. Deeptox: Toxicity prediction using deep learning. *Frontiers in Environmental Science*, 3:80, 2016.

[32] Andreas Mayr, Günter Klambauer, Thomas Unterthiner, Marvin Steijaert, Jörg K Wegner, Hugo Ceulemans, Djork-Arné Clevert, and Sepp Hochreiter. Large-scale comparison of machine learning methods for drug target prediction on ChEMBL. *Chemical Science*, 2018.

[33] James T Metz, Eric F Johnson, Niru B Soni, Philip J Merta, Lemma Kifle, and Philip J Hajduk. Navigating the kinome. *Nature chemical biology*, 7(4):200, 2011.

[34] Alain Mina, Rachel Yoder, and Priyanka Sharma. Targeting the androgen receptor in triple-negative breast cancer: Current perspectives. *OncoTargets and therapy*, 10:4675, 2017.

[35] Zaynab Mousavian, Sahand Khakabimamaghani, Kaveh Kavousi, and Ali Masoudi-Nejad. Drug–target interaction prediction from PSSM based evolutionary information. *Journal of pharmacological and toxicological methods*, 78:42–51, 2016.

[36] Javier Munoz, Jennifer J Wheler, and Razelle Kurzrock. Androgen receptors beyond prostate cancer: an old marker as a new target. *Oncotarget*, 6(2):592, 2015.

[37] NCBI. Ncbi batch entrez. `https://www.ncbi.nlm.nih.gov/sites/batchentrez`. Accessed: 2018-02-02.

[38] Hakime Öztürk, Elif Ozkirimli, and Arzucan Özgür. DeepDTA: Deep drug-target binding affinity prediction. *arXiv preprint arXiv:1801.10193*, 2018.

[39] Tapio Pahikkala, Antti Airola, Sami Pietilä, Sushil Shakyawar, Agnieszka Szwajda, Jing Tang, and Tero Aittokallio. Toward more realistic drug–target interaction predictions. *Briefings in bioinformatics*, 16(2):325–337, 2014.

[40] Tianyi Qiu, Jingxuan Qiu, Jun Feng, Dingfeng Wu, Yiyan Yang, Kailin Tang, Zhiwei Cao, and Ruixin Zhu. The recent progress in proteochemometric modelling: Focusing on target descriptors, cross-term descriptors and application scope. *Briefings in bioinformatics*, page bbw004, 2016.

[41] Bharath Ramsundar, Peter Eastman, Karl Leswing, Patrick Walters, and Vijay Pande. *Deep Learning for the Life Sciences.* O'Reilly Media, 2019. `https://www.amazon.com/Deep-Learning-Life-Sciences-Microscopy/dp/1492039837`.

[42] David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of chemical information and modeling*, 50(5):742–754, 2010.

[43] Kunal Roy, Supratik Kar, and Rudra Narayan Das. Chapter 7 - validation of QSAR models. In Kunal Roy, Supratik Kar, and Rudra Narayan Das, editors, *Understanding the Basics of QSAR for Applications in Pharmaceutical Sciences and Risk Assessment*, pages 231 – 289. Academic Press, Boston, 2015.

[44] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[45] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[46] Alok Sharma, James Lyons, Abdollah Dehzangi, and Kuldip K Paliwal. A feature extraction technique using bi-gram probabilities of position specific scoring matrix for protein fold recognition. *Journal of theoretical biology*, 320:41–46, 2013.

[47] Robert H Shoemaker. The NCI60 human tumour cell line anticancer drug screen. *Nature Reviews Cancer*, 6(10):813–823, 2006.

[48] Justin S Smith, Olexandr Isayev, and Adrian E Roitberg. Ani-1: An extensible neural network potential with dft accuracy at force field computational cost. *Chemical science*, 8(4):3192–3203, 2017.

[49] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

[50] Iurii Sushko, Sergii Novotarskyi, Robert Korner, Anil Kumar Pandey, Artem Cherkasov, Jiazhong Li, Paola Gramatica, Katja Hansen, Timon Schroeter, Klaus-Robert Muller, et al. Applicability domains for classification problems: benchmarking of distance to models for Ames mutagenicity set. *Journal of chemical information and modeling*, 50(12):2094–2111, 2010.

[51] Drug Synthesis and Chemistry Branch of National Cancer Institute. Developmental therapeutics program, division of cancer treatment and diagnosis. https://dtp.cancer.gov/. Accessed: 2019-01-17.

[52] Jing Tang, Agnieszka Szwajda, Sushil Shakyawar, Tao Xu, Petteri Hintsanen, Krister Wennerberg, and Tero Aittokallio. Making sense of large-scale kinase inhibitor bioactivity data sets: A comparative and integrative analysis. *Journal of Chemical Information and Modeling*, 54(3):735–743, 2014.

[53] Chemical Computing Group ULC. Molecular operating environment (MOE), 2013.08. https://www.chemcomp.com/MOE-Molecular_Operating_Environment.htm, 8 2013.

[54] Thomas Unterthiner, Andreas Mayr, Günter Klambauer, Marvin Steijaert, Jörg K Wegner, Hugo Ceulemans, and Sepp Hochreiter. Multi-task deep networks for drug target prediction. In *Neural Information Processing System*, pages 1–4, 2014.

[55] EPA US. Summary files from invitrodb_v2. https://www.epa.gov/chemical-research/toxicity-forecaster-toxcasttm-data, 10 2015. Accessed: 2018-03-22.

[56] EPA US. Toxicity forecaster (ToxCast) fact sheet. https://www.epa.gov/sites/production/files/2016-12/documents/tox_cast_fact_sheet_dec2016.pdf, 2016. Accessed: 2018-03-22.

[57] Gerard JP van Westen, Jörg K Wegner, Adriaan P IJzerman, Herman WT van Vlijmen, and A Bender. Proteochemometric modeling as a tool to design selective compounds and for extrapolating to novel targets. *MedChemComm*, 2(1):16–30, 2011.

[58] Izhar Wallach, Michael Dzamba, and Abraham Heifets. AtomNet: A deep convolutional neural network for bioactivity prediction in structure-based drug discovery. *arXiv preprint arXiv:1510.02855*, 2015.

[59] Ming Wen, Zhimin Zhang, Shaoyu Niu, Haozhi Sha, Ruihan Yang, Yonghuan Yun, and Hongmei Lu. Deep-learning-based drug–target interaction prediction. *Journal of proteome research*, 16(4):1401–1409, 2017.

[60] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. MoleculeNet: A benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

[61] 徐优俊 (Youjun Xu) and 裴剑锋 (Jianfeng Pei). 深度学习在化学信息学中的应用(the application of deep learning in cheminformatics). 大数据*(Big Data)*, 3(2):2017019, 2017.

[62] Y Yamanishi, M Araki, A Gutteridge, W Honda, and M Kanehisa. Prediction of drug-target interaction networks from the integration of chemical and genomic spaces. *Bioinformatics*, 24(13):232–240, 2008.

[63] Timothy A Yap, Alan D Smith, Roberta Ferraldeschi, Bissan Al-Lazikani, Paul Workman, and Johann S de Bono. Drug discovery in advanced prostate cancer: Translating biology into therapy. *Nature Reviews Drug Discovery*, 15(10):699, 2016.

[64] 周志华 (Zhihua Zhou). 机器学习*(Machine Learning)*. Qing hua da xue chu ban she (Tsinghua University Press), 2016.

Appendix A

Code and datasets

PADME_supplementary_dataset_files.zip: dataset used and the code for preprocessing.
The source code and some processed datasets are deposited at https://github.com/simonfqy/PADME.
Some bigger processed datasets could be obtained upon request.