2018

Scalable Feature Selection and Extraction with Applications in Kinase Polypharmacology

Derek Jones
University of Kentucky, derek.jones4@uky.edu
Author ORCID Identifier:
 https://orcid.org/0000-0002-9510-6662
Digital Object Identifier: https://doi.org/10.13023/ETD.2018.137

Click here to let us know how access to this document benefits you.

Recommended Citation

STUDENT AGREEMENT:

I represent that my thesis or dissertation and abstract are my original work. Proper attribution has been given to all outside sources. I understand that I am solely responsible for obtaining any needed copyright permissions. I have obtained needed written permission statement(s) from the owner(s) of each third-party copyrighted matter to be included in my work, allowing electronic distribution (if such use is not permitted by the fair use doctrine) which will be submitted to UKnowledge as Additional File.

I hereby grant to The University of Kentucky and its agents the irrevocable, non-exclusive, and royalty-free license to archive and make accessible my work in whole or in part in all forms of media, now or hereafter known. I agree that the document mentioned above may be made available immediately for worldwide access unless an embargo applies.

I retain all other ownership rights to the copyright of my work. I also retain the right to use in future works (such as articles or books) all or part of my work. I understand that I am free to register the copyright to my work.

REVIEW, APPROVAL AND ACCEPTANCE

The document mentioned above has been reviewed and accepted by the student's advisor, on behalf of the advisory committee, and by the Director of Graduate Studies (DGS), on behalf of the program; we verify that this is the final, approved version of the student's thesis including all changes required by the advisory committee. The undersigned agree to abide by the statements above.

Derek Jones, Student

Dr. Sally Ellingson, Major Professor

Dr. Miroslaw Truszczynski, Director of Graduate Studies

Scalable Feature Selection and Extraction with Applications in Kinase Polypharmacology

---

THESIS

A thesis submitted in partial

fulfillment of the requirements for

the degree of Master of Science in

the College of Engineering at the

University of Kentucky

By

Derek Jones

Lexington, Kentucky

Co-Director: Dr. Sally Ellingson, Assistant Professor of Biomedical Informatics

Co-Director: Dr. Nathan Jacobs, Associate Professor of Computer Science

Lexington, Kentucky 2018

ABSTRACT OF THESIS

Scalable Feature Selection and Extraction with Applications in Kinase
Polypharmacology

In order to reduce the time associated with and the costs of drug discovery, machine learning is being used to automate much of the work in this process. However the size and complex nature of molecular data makes the application of machine learning especially challenging. Much work must go into the process of engineering features that are then used to train machine learning models, costing considerable amounts of time and requiring the knowledge of domain experts to be most effective. The purpose of this work is to demonstrate data driven approaches to perform the feature selection and extraction steps in order to decrease the amount of expert knowledge required to model interactions between proteins and drug molecules.

KEYWORDS: Machine Learning, Deep Learning, Chemoinformatics, Bioinformatics, Drug Discovery

Author's signature: _____ Derek Jones

Date: _____ May 4, 2018

Scalable Feature Selection and Extraction with Applications in Kinase

Polypharmacology

By

Derek Jones

Co-Directors of Thesis: <u>Dr. Sally Ellingson and Nathan Jacobs</u>

Director of Graduate Studies: <u>Miroslaw Truszczynski</u>

Date: <u>May 4, 2018</u>

I dedicate this work to my entire family, and my brother Brandon who has traveled much of this road with me.

ACKNOWLEDGMENTS

I would like to thank Dr. Sally Ellingson for her help in advising me on the bioinformatics aspects of this work as well as for her support for my research. I would like to thank Dr. Nathan Jacobs for his help in advising me on the machine learning aspects of this work and for his overall support during this process. I would like to thank Jeevith Bopaiah for his help in many aspects of the research that is presented in this work. I would like to thank the other students whom I have worked with and come to befriend in my research group for their mentorship and advice. I would like to thank Dr. Licong Cui for her assistance in providing domain expertise to ensure the quality of this work and its relevance. I would like to thank Dr. Judy Goldsmith for her mentorship and friendship, as she played a pivotal role in my decision to begin graduate study. I would lastly like to thank Dr. Miroslaw Truszczynski for his advice and helpfulness in supporting my research career over the course of my time in the Department of Computer Science at the University of Kentucky.

Table of Contents

LIST OF FIGURES

Chapter 1

Introduction

1.1 Challenges in Drug Discovery

Drug Discovery is known to be an expensive and time-consuming process. On average, it takes nearly a decade, with research and development costs exceeding $1.4 billion, to develop a *single* successful candidate that is able to gain FDA approval [8, 15]. Not surprisingly, the level of investment in pharmaceutical r&d has increased in order to deliver higher quality treatments as well as to bring down the cost of these treatments to the consumer. However despite these increased levels of investment in pharmaceutical r&d, the progress that has been made in achieving these goals leaves much to be desired. In fact, the number of new FDA approved drugs has roughly halved every 9 years since 1950 [34], suggesting that new approaches must be taken to address this problem.

Drug Discovery is difficult for a number of reasons. For one, the number of *active*, or binding, compounds are greatly outnumbered by *inactive*, or non-binding, compounds. As the size of the molecular space is estimated to be between 10^{23} - 10^{60}, the problem of "discovering" an active compound can be likened to finding a needle in the haystack [29]. Secondly, many existing methods for identifying active

binding compounds, such as molecular docking simulations, are not highly accurate or able to reliably select active binding compounds, and restrict the throughput of a drug discovery pipeline. Thirdly, gaining access to rich datasets is difficult as many pharmaceutical companies do not publicly share much of the data they generate from their own research, making progress in developing practical computational drug discovery methods more difficult to achieve.

1.2 Improving Drug Discovery with Machine Learning

In recent years, the availability of larger amounts of realistic data [27] as well as advancements in machine learning and *deep learning* have given rise to the hope of making substantial improvements in the throughput and efficiency in the drug discovery process. The focus of this work is to demonstrate how these modern techniques can be used in conjunction with a focus on efficiently scaling to the available computational resources to address some of the aforementioned challenges. The problem of interest in this work is the development of methods to predict the likelihood of a drug-like molecule binding to a given target protein (figure 1.1). In Chapter 2, we present a feature selection method that uses an ensemble of weak learners in parallel to not only make the prediction for this task, but to also learn which features are most informative in a data-driven manner. In Chapter 3, we present a method that is able to learn feature representations directly from molecular structures as input, removing the need for explicit feature extraction steps during preprocessing. Furthermore, the method implements a distributed optimization procedure that again scales to the amount of available compute power. In Chapter 4, future directions for research are discussed that examine ways in which the molecular search space can be traversed more efficiently to develop better drug-like molecule libraries. Two

overarching themes throughout this work is the development of methods that are able to leverage the available compute resources, however large or small, with the ability to learn without human intervention, using the available data to perform the feature selection and extraction steps.
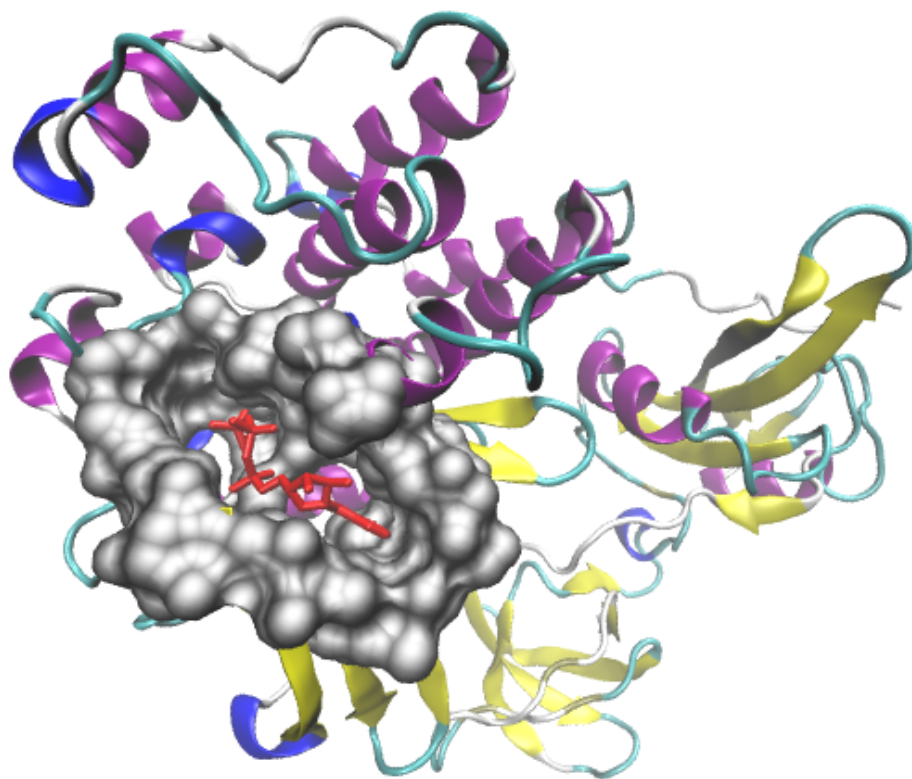
Figure 1.1: An example of a drug molecule binding to a protein and the binding pocket

Chapter 2

Data-Driven Feature Selection in Binding Affinity Models

2.1 Introduction

Protein kinases represent a large number of proteins in our body with essential functions. Because of this, any disruption in normal kinase activity may lead to a disease state. Additionally, due to high sequence and structural identity, selectively inhibiting a kinase is difficult. This means a drug intended to target one kinase will likely also target multiple other kinases. If these other kinases are normally expressed and not implicated in the given disease it could lead to toxic off-target effects. Pharmaceutical companies test drug interactions with many different kinases in the beginning of the drug discovery process. They do this as early as possible before lots of time and money has gone into drug development [4]. Drugs failing late in the pharmaceutical pipeline can be very costly, driving up the cost of drugs that do make it to market when they have to recuperate the cost for the failed drugs. It can also be fatal when they fail during clinical trial, because animal testing does not always give a good indication of serious side-effects [32]. Therefore, our interest in accurate computa-

tional models to study kinases is to develop better and safer cancer therapies, using efficient computational predictions that reduce the time and cost of bringing a drug to market.

We propose to use machine learning techniques to increase the accuracy of computational drug discovery in order to make better predictions as early as possible. We have seen in our own work that a small number of calculated features similar to ones used in this study can identify active compounds for a given protein with greater than 99% accuracy. These same drug features have been used in machine learning models in combination with docking scores to rescore interactions with one candidate drug to multiple proteins [14]. The individual components of a molecular docking scoring function can be used as features in a machine learning model to greatly improve the accuracy of identifying active compounds in models specific for one protein [19]. From a different perspective, protein features have been used in machine learning models to predict the druggability of a protein [18]. The goal of this work is to combine all these components in one model that would vastly improve the accuracy of predicting the effects of new proteins and classes of drugs. The specific goal of this chapter is to present machine learning models that can accurately predict the drug interaction for a class of functionally related proteins (kinases), an important class of proteins for drug discovery as already stated.

2.2   Methods

Our goal is to estimate the probability that a kinase-drug pair is active (binding) or decoy (not binding), a binary classification task. We propose to use a random forest classification method to address this task. A key focus of our effort is in investigating which features are most informative for this task. To support this effort, we created a large dataset of kinase-drug pairs and computed a wide variety of different features.

The data used in this study comes from the kinase subset of the Directory of Useful Decoys - enhanced (DUD-e) [27]. It is important to note that the ratio of active to decoy compounds in DUD-e is approximately 1:50.

Data Collection

- **Protein Descriptors** The human canonical sequences were collected for each protein from UniProt [3]. The sequences were submitted to three different webservers to collect features: ExPasy [11], Porter, PaleAle 4.0 [24], and PROFEAT Protein Feature Server [39]. These three tools were used to ensure we collect all features used in the DrugMiner [18] project. Additional features that these tools calculate are also collected. ExPasy calculates many features, such as the length, weight, half-life, isoelectric point, extinction coefficient assuming all pairs of Cys residues form cysteines, extinction coefficient assuming all Cys residues are reduced, instability index, aliphatic index, Grand average of hydropathicity (GRAVY), and the frequency of single amino acids, amino acid types (tiny, small, aliphatic, nonpolar, aromatic, polar, charged, basic, acidic, hydrophobic, hydrophilic, positive, and negative), and atom types. Porter calculates the predicted secondary structure based on the amino acid sequence and classifies each amino acid as helical, beta strand, or coil. PaleAle calculates the predicted relative solvent accessibility based on the amino acid sequence and classifies each amino acid as completely buried (0-4% exposed), partly buried (4-25% exposed), partly exposed (25-50% exposed), or completely exposed (50+% exposed). PROFEAT calculates features using many different tools including features based on the dipeptide composition of the protein sequence.

- **Pocket Descriptors** Inner point features are collected using PRANK [20], software used to predict and rank binding sites. PRANK first calculates feature vectors for heavy solvent exposed atoms (AFVs), including residue and atomic

level features. Then feature vectors are calculated for inner pocket points (IFVs) by summing all AFVs within an 8 Å radius using a distance weight function and then appending features specific to the inner pocket point, such as the number of H-bond donors and acceptors in its local neighborhood. The IFV from the inner pocket point with the closest distance to the center of the docking box calculated for molecular docking is used.

- **Drug Descriptors** Drug features are calculated using the Dragon Software [36]. Dragon can calculate over 5 thousand molecular descriptors, including the simplest atom types, functional groups and fragment counts, topological and geometrical descriptors, and three-dimensional descriptors. It also includes several property estimations like logP and drug-like alerts like Lipinski's alert. In this study 3-dimensional descriptors are left out because the input structures for Dragon are the predocking structures and not those predicted by molecular docking.

- **Binding Descriptors** The receptor files from DUD-e that were optimized for docking are used in this study. The dimension and center of the docking boxes are calculated using a VMD [16] tcl script to draw a box around the co-crystallized ligand included in the DUD-e dataset and it is extended by 5 Å in each direction. Compounds are prepared for docking using modified ADT scripts and a wrapper script for automation. Docking was performed using VinaMPI [10], which allows the distribution of a large number of Autodock Vina [38] docking jobs on MPI-enabled high-performance computers. The results of the docking jobs were submitted to Autodock Vina using the "–score-only" option to collect the individual terms calculated in the scoring function. This includes terms for gauss1, gauss2, repulsion, hydrophobic, and hydrogen interactions. The values for the first model and averages of each term for all

models are kept.

Feature Selection & Classification

The only form of preprocessing we performed was eliminating features with too many missing values. Specifically, we eliminated 21 features computed by the Dragon software package that had more than 5% missing values. The eliminated features had between 23.1%-99.9% missingness. There were 167 additional Dragon features that had less than 5% missing values and we imputed these values by using the column average. The final full dataset contains 5,410 features and 361,786 examples. After initial preprocessing, we train a classifier for various subsets of features and perform feature selection.

Random forests [5] are known to produce robust classifiers that are less prone to overfitting than ordinary decision trees. For a brief review, random forests contain a number of decision trees, a parameter that is chosen prior to training, each of which take random samples from the training data and random subsets of features to grow decision trees that are often limited in depth to create "weak" learners that underfit the testing data. By combining the "weak" learners that specialize in different regions of the feature space, random forests are able to learn complex functions that are robust to label imbalance or overfitting, two properties that are of great importance in our classification problem. The decision trees that make up a random forest compute orthogonal splits in feature space that attempt to maximize separation between the positive and negative classes minimize what is known as the GINI Impurity. From training a random forest, one can compute feature importances by measuring the average after-split impurity of the feature across all trees in the forest.

The feature selection method we use is similar to those used by [23] and [2] who also apply iterative feature selection method for classification to learn important sets of features. In our feature selection method, we input an initial set of features F for

which we use to train a random forest classifier. The input data, after preprocessing, is partitioned into training and testing sets using an 80/20 stratified split, with the test set containing the same proportion of positives to negatives as the training set. We fix several parameters of the random forest classifier by using an out-of-bag score to protect against overfitting, balanced class weighting when computing impurities for the forests which inversely adjust the weights according to class frequency to get measures of the F1-score that better reflect the random forest's performance in correctly predicting the active class, bootstrap sampling which allows training examples to be used in the building of more than one tree, and the GINI impurity criterion for which to compute the split quality when building the tree. We then perform model selection by sampling from distributions of hyperparameters, shown to be as effective as exhaustive parameter grid searching by [21], for the random forest including the number of trees to include (30-100 trees) in the forest, the minimum number of samples required to create a leaf node (1-100 samples), and the maximum number of features $f \in F$ to sample from F for each decision tree ($\sqrt{|F|}$ and $log_2(|F|)$). Given the distributions over hyperparameter values, we sample 100 possible settings of hyperparameters each iteration, evaluating the performance of each candidate model using k-fold cross validation, with $k = 3$. We define the best model trained on the feature set to be the one which maximizes the weighted F1-score on the testing data. We then compute the mean importance, more specifically the mean decrease in impurity ($\frac{1}{|F|}$), for the set of features F, and retain all features that have above mean importance. This strategy is employed in order to remove features with near 0 importance that contribute negligible information to the classification model and do not have a significant affect on performance. After computing the set of features to keep, F becomes the set of features identified as relevant, reducing the dimensionality of the input data. The iteration process continues until either a maximum number of iterations have completed or if there are no remaining features.

Algorithm 1 Feature Selection

1: **procedure** SELECTION FOREST(F)
2:     **while** $F \neq \emptyset$ and $step < max\_steps$ **do**
3:         $X, y = load\_data(features = features\_to\_keep)$
4:         $X_{train},\ X_{test},\ y_{train},\ y_{test} = train\_test\_split(X, y)$
5:         $best\_forest = RandomizedGridSearch(RandomForest,$
            $X_{train}, y_{train}).best\_estimator$
6:         $feature\_importances = best\_forest.importances$
7:         $features\_to\_keep = feature\_importances > \frac{1}{|feature\_importances|}$
8:         $F = features\_to\_keep$

We use principal component analysis (PCA) to visualize the various feature representations. For our purpose of visualization, we reduce the dimensionality to 2 principal components.

Test set

We used a fixed seed when creating the test set in order to make sure all models are tested with the same data. The test and training sets are stratified by kinase, keeping the same proportion of active and decoy compounds for each.

Table 2.1 gives the representation of each kinase in the test set. The whole dataset 'total' column gives the total number of active and decoy compounds for the given kinase in the whole dataset. The whole dataset 'ratio 0:1' column gives the ratio of negative to positive class for the entire given kinase's dataset. The remaining columns are particular to the test set. The '0' column gives the number of decoys (negative class) in the test set. The '1' column gives the number of actives (positive class) in the test set. The 'percent 0' column gives the percentage of the given kinase's dataset that is in the negative class test set. The 'percent 1' column gives the percentage of the given kinase's dataset that is in the positive class test set. The 'total %' column gives the percentage of the given kinase's dataset that is in the test set. The 'ratio 0:1' gives the ratio of negative to positive test cases for the given kinase in the test set.

10

Table 2.1: Representation of each kinase in the test set

| kinase | whole dataset | | test dataset | | | | | |
	total	ratio 0:1	0	1	percent 0	percent 1	total %	ratio 0:1
abl1	11,180	37	2,105	60	0.188	0.005	0.194	35
akt1	16,999	39	3,298	73	0.194	0.004	0.198	45
akt2	7,142	37	1,462	34	0.205	0.005	0.209	43
braf	10,349	40	2,036	46	0.197	0.004	0.201	44
cdk2	29,126	35	5,604	158	0.192	0.005	0.198	35
csf1r	12,720	43	2,563	54	0.201	0.004	0.206	47
egfr	36,274	43	6,936	158	0.191	0.004	0.196	44
fak1	5,516	47	1,085	14	0.197	0.003	0.199	78
fgfr1	736	2	183	116	0.249	0.158	0.406	2
igf1r	9,633	42	1,958	42	0.203	0.004	0.208	47
jak2	6,743	43	1,343	32	0.199	0.005	0.204	42
kit	10,861	42	2,144	52	0.197	0.005	0.202	41
kpcb	9,092	36	1,732	42	0.190	0.005	0.195	41
lck	28,539	41	5,569	136	0.195	0.005	0.200	41
mapk2	6,450	30	1,240	40	0.192	0.006	0.198	31
met	11,677	47	2,278	47	0.195	0.004	0.199	48
mk01	4,767	33	889	31	0.186	0.007	0.193	29
mk10	6,900	36	1,358	50	0.197	0.007	0.204	27
mk14	37,347	40	7,265	184	0.195	0.005	0.199	39
mp2k1	8,483	34	1,693	41	0.200	0.005	0.204	41
plk1	7,034	44	1,402	32	0.199	0.005	0.204	44
rock1	6,580	31	1,306	41	0.198	0.006	0.205	32
src	35,790	42	6,980	167	0.195	0.005	0.200	42
tgfr1	8,958	31	1,704	63	0.190	0.007	0.197	27
vgfr2	25,900	41	5,119	123	0.198	0.005	0.202	42
wee1	6,371	46	1,245	25	0.195	0.004	0.199	50

Evaluation

In this study, we compare the performance of machine learning models using different feature sets and also compare the performance to the computed docking score. Docking scores are typically used for ranking compounds from most likely to least likely to bind and there is no standard that defines an exact docking score that determines a binding prediction. In order to compare binding predictions from the docking score alone to the machine learning models, the maximum Youden's index (or J value) is calculated for each model. The best J value is calculated from the docking score receiver operator characteristic (ROC) curve and used as a cut-off to define true pos-

itive (TP), true negative (TN), false positive (FP), and false negative (FN) values for the docking results. The different feature sets are described below and we compare 6 models using the feature sets given in Table 2.2. All the metric presented in the Results are defined in Table 2.3.

Table 2.2: Evaluation Models

Model	Feature Set	Model	Feature Set	Model	Feature Set
1	FS1	3	FS1 + FS3	5	FS1 + FS2 + FS3 + FS4
2	FS4	4	FS1 + FS3 + FS4	6	all features

Table 2.3: Metrics used in this study

Name	Definition	Formula
Youden's index	Performance of dichotomous test. The value 1 indicates a perfect test and -1 indicates a useless test.	$\frac{TP}{TP+FN} + \frac{TN}{TN+FP} + 1$
F1	Harmonic mean of precision and recall	$\frac{2TP}{2TP+FP+FN}$
Precision	Positive predictive value	$\frac{TP}{TP+FP}$
Recall	True positive rate	$\frac{TP}{TP+FN}$

Table 2.4: Youden's Index.

Kinase	Youden's Index	Best docking score	Kinase	Youden's Index	Best docking score
abl1	0.35	-9.1	lck	0.29	-8.9
akt1	0.02	-8	mapk2	0.45	-8
akt2	0.22	-8.5	met	0.51	-8.9
braf	0.53	-9.6	mk01	0.6	-9.1
cdk2	0.33	-8.2	mk10	0.4	-8.7
csf1r	0.23	-8.9	mk14	0.28	-8.5
egfr	0.15	-8.7	mp2k1	0.13	-7.8
fak1	0.52	-8.4	plk1	0.2	-8.6
fgfr1	0.01	-8	rock1	0.4	-7.7
igf1r	0.46	-8.4	src	0.17	-8.1
jak2	0.37	-9.3	tgfr1	0.56	-9.6
kit	0.24	-8.5	vgfr2	0.36	-9
kpcb	0.33	-8.5	wee1	0.76	-10
Overall	0.23	-8.6			

- **Feature Set 1 (FS1):** This set is selected using the entire dataset and using the active or decoy binary labels. This is to collect the most important features for making the classification in which we are interested (i.e. active vs decoy).

- **Feature Set 2 (FS2):** This set is selected using only protein and pocket features and using the kinase as a label. We do this to ensure we have protein features to test whether or not they help identify which kinase compounds bind to and not just identify kinase inhibitors in general.

- **Feature Set 3 (FS3):** This set is selected using the drug features with the kinase as a label. This is also used to help with kinase selectivity.

- **Feature Set 4 (FS4):** This set contains all docking features, which includes terms for gauss1, gauss2, repulsion, hydrophobic, and hydrogen interactions for the first docked model produced using molecular docking and an average over all models (the default value of 9 models was kept when running Vina). There is also a feature for the final docking score.

2.3   Results

Youden's Index for Docking Scores

The maximum Youden's index (or J value) is calculated and used to define TP, FP, TN, and FN values using docking scores. The best J values and docking score cut-off for each kinase and on the dataset overall all are given in Table 2.4.

PCA of Feature Sets

We performed a PCA of each Feature Set (FS) described in the Methods section. Figures 2.1 and 2.2 plot the first two components for FS1-FS4. FS1 contains 776 features all which are drug features. FS2 contains three protein features that are most important in determining the kinase. These are *[G3.1.1.1.19], [G4.1.23.3], [G4.3.17.2]* and come from the PROFEAT webserver [39]. These correspond to an autocorrelation descriptors based on the distribution of amino acid types along the
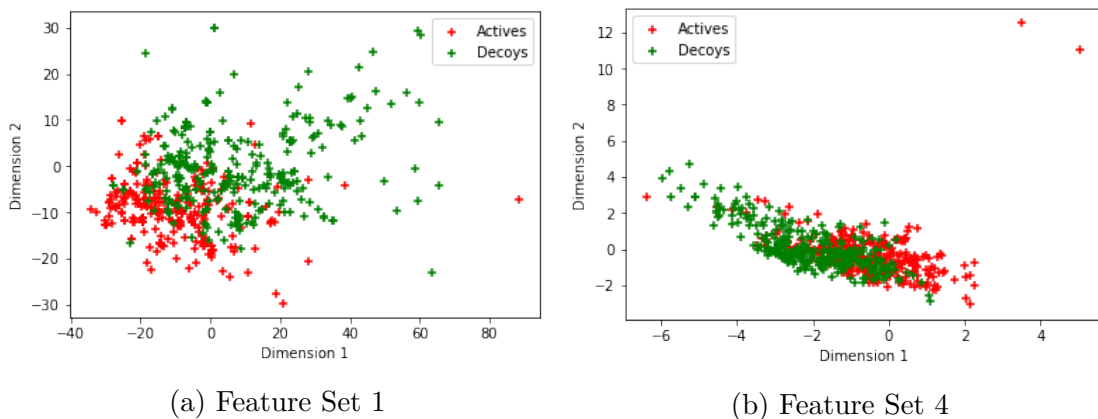
(a) Feature Set 1   (b) Feature Set 4

Figure 2.1: PCA of FS1 and FS4

(a) Feature Set 2   (b) Feature Set 3
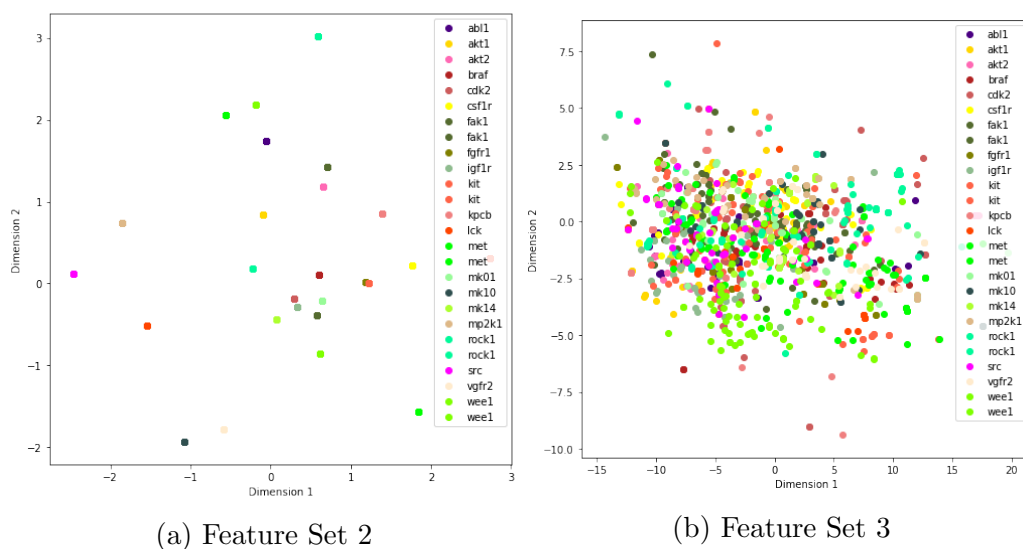
Figure 2.2: PCA of FS2 and FS3

protein sequence, a protein-ligand binding site propensity descriptor, and a protein-DNA interface propensity descriptor, respectively. FS3 gives us 191 features which has an overlap of 134 features with FS1, so this set has 57 new features in it. FS4 has 11 features.

Evaluation of Models Containing Different Feature Sets

Metrics (which are defined in Table 2.3) for each model are given in Table 2.5 and a comparison to the docking metrics obtained using the test set is also given. All

models do very well at identifying the decoy compounds or negative class. However, given the ratio of decoys to actives, a model could always predict decoy and give good results. Therefore, our focus is on the positive class metrics. Interestingly Model 2 which is all docking features and would hopefully be highly predictive, has the worst performance in all metrics. Model 2 still performs better than docking alone by F1-score. Model 5 gives the best F1-score which is a key metric as it is a balance between the precision and recall. It also gives the best recall of 0.92. Additionally, this is the model that includes protein features and we are interested in the added benefit of including them.

In Table 2.6 we further analyze the per kinase performance. All metrics here are using Model 5 and again the metrics based on docking scores alone are also given. The "Per kinase" columns are metrics on each individual kinase from the analysis given in Table 2.5. The "Leave-one-out" columns are additional models using the same feature sets as Model 5, but in which one kinase is left out for testing while all the other kinases are in the training set.

2.4   Discussion

We can see from the PCA (figures 2.1, 2.2) that FS1 (drug features selected based on active or decoy classification) has a fairly good separation of the two classes and alone gives good predictions with an F1-score of 0.87. FS4 (docking features) has some separation between the classes but also a sizable overlap and only an F1-score of 0.28 (Model 2) for classifying the compounds. FS3 (drug features selected based on kinase classification) do not do a great job at kinase classification and do not improve the model over using just FS1 (Model 3 vs Model 1). Only three protein features were selected for their ability to classify the kinase (FS2) and this model, Model 5, gives the best F1 score.

Even though the drug features are by far the most informative features in these models, they cannot account for kinase selectivity. When including features that are informative at classifying the kinase, we had a slight increase in F1 score. Even though kinase inhibitors are promiscuous and kinases have a high sequence and structural similarity, this provides hope that protein features can be informative in universal (multi-protein) models for drug binding.

We can see that the per kinase performance is much better when each kinase is represented in the training set by comparing the "Per kinase" and "Leave-one-out" columns in Table 2.6. However, many kinases still perform very well in the "leave-one-out" analysis. We believe that as we add more kinases to the model the "leave-one-out" analysis will improve. Recent results using a diverse set of proteins also show promise to improve models.

To exemplify the usefulness of our method, we have identified a compound that would never be identified in a docking virtual screen as an active compound (receiving a score of +95.19, when the most negative scores predict binding), that has been saved using this machine learning model. An example of such a compound is CHEMBL448926, an *ackt1* active compound. This compound is an actual *ackt1* inhibitor patented by Merck and Co Inc. (Patent ID US7544677) and directed to chemotherapeutic compositions. The reason this compound may be lost during docking is it a potent allosteric inhibitor [35].

Docking is often used as a tool to enrich a subset of data and therefore early enrichment is a common important metric. For example, if a large virtual drug set has 2% unknown active compounds in it then hopefully the top subset of scored drugs by docking will have maybe 10% active compounds in it. This would allow researchers to select a smaller set of drugs for experimental testing and have a greater success rate than randomly selecting a subset for testing. We can see here that the precision is always low for docking, therefore to recover the same amount of active compounds

using docking you would always have a much higher false positive rate making the number of compounds needed for experimental validation to be much higher for the same success rate.

With the test case presented here evaluated with Model 5, which includes protein features and gives the best F1 score, 97% of the test data is classified as non-binding. Therefore, 97% of the data can immediately be discarded and you would lose less than 10% of the binding compounds at this prediction stage. Experimentally testing the predicted active compounds would give an 83% success rate at identifying active compounds. Testing the same number of compounds based on docking score alone would have less than a 27% success rate. Part of the problem here is that docking scores are not a good indicator of binding when looking at multiple proteins. The range of docking scores varies per protein. This demonstrates a huge advantage to our machine learning approach for a multi-protein model.

Table 2.5: Comparison of performance for both classes on the testing set.

Model	Class	Precision	Recall	F1-Score	Class	Precision	Recall	F1-Score
1	0	1.00	1.00	1.00	1	0.83	0.92	0.87
2	0	0.98	0.98	0.98	1	0.26	0.30	0.28
3	0	1.00	1.00	1.00	1	0.83	0.92	0.87
4	0	1.00	1.00	1.00	1	0.83	0.91	0.87
5	0	1.00	1.00	1.00	1	0.84	0.92	0.88
6	0	1.00	1.00	1.00	1	0.85	0.89	0.87
Docking	0	0.99	0.58	0.73	1	0.04	0.67	0.07

Table 2.6: Evaluation metrics per kinase for the positive class

	Per Kinase			Leave-one-out			Docking		
Kinase	Prec.	Recall	F1	Prec.	Recall	F1	Prec.	Recall	F1
abl1	0.92	0.95	0.93	0.86	0.95	0.9	0.06	0.72	0.11
akt1	0.86	0.96	0.91	0.84	0.81	0.82	0.03	0.29	0.05
akt2	0.94	0.94	0.94	0.83	0.8	0.81	0.04	0.59	0.07
braf	0.85	0.98	0.91	0.76	0.85	0.8	0.06	0.80	0.11
cdk2	0.85	0.77	0.81	0.6	0.31	0.41	0.04	0.78	0.08
csf1r	0.82	0.83	0.83	0.62	0.59	0.6	0.03	0.63	0.06
egfr	0.85	0.91	0.88	0.76	0.75	0.75	0.03	0.81	0.05
fak1	0.75	0.86	0.8	0.8	0.77	0.78	0.03	0.86	0.06
igf1r	0.93	1.00	0.97	0.9	0.92	0.91	0.05	0.86	0.09
jak2	0.97	0.94	0.95	0.87	0.77	0.82	0.06	0.5	0.10
kit	0.83	0.92	0.87	0.74	0.88	0.81	0.05	0.58	0.08
kpcb	0.70	0.93	0.80	0.45	0.32	0.37	0.04	0.86	0.08
lck	0.94	0.93	0.94	0.8	0.83	0.82	0.05	0.49	0.09
mapk2	0.93	0.97	0.95	0.72	0.37	0.49	0.06	0.65	0.10
met	0.87	0.96	0.91	0.75	0.77	0.76	0.05	0.81	0.09
mk01	0.91	0.97	0.94	0.78	0.69	0.73	0.16	0.71	0.26
mk10	0.88	0.86	0.87	0.67	0.51	0.58	0.06	0.82	0.11
mk14	0.90	0.85	0.88	0.76	0.5	0.6	0.05	0.56	0.09
mp2k1	0.80	0.95	0.87	0.59	0.48	0.53	0.03	0.93	0.05
plk1	0.82	0.84	0.83	0.73	0.5	0.59	0.03	0.91	0.06
rock1	0.88	0.85	0.86	0.63	0.32	0.42	0.05	0.88	0.10
src	0.93	0.97	0.95	0.86	0.87	0.86	0.03	0.88	0.06
tgfr1	0.94	0.98	0.96	0.89	0.83	0.86	0.09	0.84	0.17
vgfr2	0.93	0.90	0.92	0.8	0.79	0.8	0.05	0.65	0.10
wee1	0.93	1.00	0.96	0.75	0.58	0.65	0.14	0.8	0.24

Limitations

While our results highlight the potential of our approach, there are several limitations of our evaluation that warrant further investigation. The main limitation is that we do not know whether a compound that is active for one kinase is not active for another. There is some overlap between the 26 different active sets but it is not much. Since selectively inhibiting a kinase is difficult it should be experimentally validated before marking a compound that is active for one kinase as not for another one. It is also difficult to tell if a given active for one kinase is in the decoy set for another kinase because the active and decoy compounds in DUD-e come from different databases, CHEMBL [12] and ZINC [17], respectively. Having an all-to-all set of connections where we know whether every drug in our dataset binds or does not bind to every protein in our dataset may uncover important features for this selectivity. Also, due to concerns with the *fgfr1* dataset (i.e. the proportion of actives to decoys does not match what is expected from DUD-e), we have excluded *fgfr1* compounds from our test set.

Future Work

Some potential future directions include (1) evaluating different ways of using the pocket features that may correlate better with predictions, (2) incorporating information on multiple possible binding sites in the model, and (3) incorporating diverse proteins in the dataset.

2.5   Conclusion

We successfully created a model of several kinases that makes good binding predictions. We found that the features we collected greatly increased binding predictions when used in a machine learning model over docking scores alone. A model using

features selected based on which kinase they belong to gave the best F1 score which balances precision and recall. We calculate a nearly 60% increase in success rate for discovering active compounds over docking.

2.6 Acknowledgements

Chapter 3

Distributed Learning of Molecular Feature Representations

3.1 Introduction

It is common for modern screening pipeline methods in computational drug discovery to require the use of feature extraction steps to compute meaningful vector representations of drug molecules that can later be used as inputs in predictive models for protein-drug interactions. This may be undesirable in at least two ways. First, these features are gathered independently of the end task, meaning that they are not able to be directly optimized for the task of interest in a manner that understands the relationships between the available data and the task of interest. Secondly, these extraction methods require some degree of domain expertise in determining which features are appropriate for the task and further, which parameters should be used to compute these features.

In this chapter we demonstrate how a deep learning methodology can be used to learn feature representations of drug-like molecules in a data-driven manner that can alleviate much of the burden in the data preparation. We show how the learned fea-

ture representations are able to provide accurate predictions of molecular properties for the drug molecules in our dataset as well as discriminate between protein-kinase inhibitors.

3.2 Related Work

One commonly used method for representing molecular structures is by encoding them using the smiles (*simplified molecular-input line-entry system*) format. A smiles string encodes the atoms and bonds, as well as the types of the bonds, of a molecule. A given smiles representation uniquely identifies a molecule, and for a given molecule their may exist a number of valid permutations of these unique identifiers. As this format is a string representation of a molecule, it is necessary to compute a vector representation of a molecules smiles representation for use as input to a machine learning system.

Drug molecules can also be understood as graph structured data. Each node in the molecular graph G represents an atom, and each edge $e_{uv} \in G$ can be thought of as a bond between two atoms u and v, with the edge e_{uv} labeled by the type of bond between the atoms u, v. Leveraging this graph representation G_m of a molecule m, one can extract a vector representation describing G_m in some context.

We briefly review some applications in which these representations have been used in machine learning, highlighting some of their limitations and thus motivating the need for data driven modeling to be used in practice.

Molecular Descriptors

Molecular descriptors are the output of some computational process "which transforms chemical information encoded within a symbolic representation of a molecule into a useful number. . ." [37]. Several solutions exist for the calculation of molecular

descriptors that can be used as input to machine learning models for tasks to predict molecular properties and activities [6, 26, 36]. A popular option is the Dragon software suite which is able to calculate 5,270 unique molecular descriptors that include features such as simplest atom types, functional groups and fragment counts, topological and geometrical descriptors, three-dimensional descriptors, as well as estimates of various properties relevant to computational drug discovery such as solubility (logP) and drug-like indices. Determining which of these features may be appropriate for a task either requires domain expertise, trial and error, or the simplest solution in using all of the available features and using some feature selection process to filter out those that are less informative according to some threshold. While using a feature selection process does help to remove the burden of domain expertise, a disadvantage to this approach is that unnecessary dimensionality may be introduced, restricting the possible approaches one may use to learn a task in a timely manner as a consequence of the *curse of dimensionality.*

Numerous studies have been performed in the context of drug-protein binding interactions that have made use of molecular descriptors as feature representations of molecules [7, 14, 18, 19, 22, 30]. In addition to traditional machine learning methods, Deep Neural Networks (DNNs) trained on molecular descriptors as feature representations have been successfully applied to problems in drug discovery. In the work by [7], Multi-Task (DNNs) were successfully applied to predict the targets of multiple PubChem assays using 3764 molecular descriptors gathered from the Dragon software suite. The authors compared their methods to several benchmarking methods, including a single-task DNN, and show that in the majority of cases their method exceeds the baseline performance in terms of the pearson correlation coefficient (i.e. R^2). In further work on applying DNNs to drug discovery, a competition launched by Merck & Co. on the Kaggle data science platform was used to generate further interest in applying modern machine learning techniques to the prediction of

molecular properties relevant to drug discovery. The data provided in the Merck & Co. molecular activity challenge consisted of molecular descriptors along with a set of activities as labels for each distinct molecule. The winning team subsequently published their results with the assistance of Merck & Co. in which they detail a crucial component behind their successful ensemble learning method, a DNN [22], showing that the DNN in most cases is able to outperform the RF across a number of hyperparameter settings on each of 15 selected datasets. While the results of early applications of DNNs were impressive in their own right given the time context, the use of molecular descriptors imposes a prior belief that all relevant or task-specific information is contained within these sets of descriptors, an obvious limitation that should be addressed in future methodologies.

Molecular Fingerprinting

Rather than explicitly computing features for a molecule such as a drug-like property or index, it is possible to instead compute a vector representation that identifies the molecule in a vector space with some intrinsic meaning. Algorithms for doing this calculation are known as *molecular fingerprinting* methods. Examples of these include morgan fingerprints [25] and extended connectivity circular fingerprints (ECFP) [33]. These algorithms compute a unique binary valued vector that identifies a given molecule based upon the atoms that it contains and their features. The state of the art fingerprinting algorithm, ECFP, computes a representation that can be used to understand properties such as similarity between molecules which can be helpful in predicting specific properties of a possibly unknown molecule as well as possible identification of active binding molecules to a target protein. However, in terms of extracting a representation for a machine learning task, a limitation of these methods is that they are computed independently of the task of interest, potentially limiting the performance of machine learning models trained using these representations.

Neural Fingerprinting

Recently, a deep learning "neural" fingerprinting method was proposed in the work by [9]. The authors alter the ECFP algorithm by introducing a differentiable function to compute molecular fingerprints, allowing the method to be optimized for specific tasks, an advantage over previous methods. Subsequently, a number of variations of this type of network using *graph convolutions* have been introduced by various groups. The work by [13] summarizes each of these techniques, and provides a unified definition of these methods named *Message-Passing Neural Networks*, or MPNNs, which can be roughly defined in two steps:

- *Message-Passing Phase*

$$m_v^{t+1} = \sum_{w \in N(v)} M_t(h_v^t, h_w^t, e_{vw}) \tag{3.1}$$

$$h_v^{t+1} = U_t(h_v^t, m_v^{t+1}) \tag{3.2}$$

   where M_t is a specified message passing function, U_t is a specified update function, h_v^t is the hidden state and m_v^t is the message received by node v at time t. This step is computed for all nodes $v \in G$ and is repeated for each node a specified number of iterations given as T. The result is the flow of local information from each node $v \in G$ across the molecular graph.

- *Readout Phase*

$$\hat{y} = R(\{h_v^T \mid v \in G\}) \tag{3.3}$$

   where R is a specified "readout" function of the hidden node states at the end of message passing at time T and $\hat{y}$ is the predicted target value of the network.

MPNNs are powerful in that they are able to learn on graphs that vary in topology, making them flexible for applications in which the inputs express this variety.

While the hidden states h_v^t can be updated simultaneously for all nodes in a graph G_m at time t, the message passing phases are sequential operations in that h_v^{t+1} is not able to be computed until the previous value h_v^t is known, so one must iterate T times to complete the message passing. When learning on large amounts of training data, this may become a troublesome property and so we explore a distributed optimization algorithm, HOGWILD! [31], as a possible method to address this concern in a practical application.

3.3   Methods

Dataset

The dataset used in the subsequent experiments is derived from the Directory of Useful Decoys-Extended (i.e. DUD-E), a molecular docking database. The molecules in this study are identified as active or decoy binding for the kinase subset of DUD-E and we use their smiles representations during preprocessing. From the smiles representations, the deepchem python library is used to construct the molecular graphs using the atom and bond feature extraction code, which encode various properties of the atoms in a given molecule as well as the bond types [1]. The atom features are used as the initial value of $h_v^{t=0}$ of size 70 and the bond features are used as the value of e_{vw} of size 6. The dataset was split into 3 partitions that are consistent across each experiment. The training and testing sets were generated using an 80/20 stratified split that keeps the number of positives consistent across each set. The training set was then further divided into a training and hold-out validation set using a 90/10 split, again using a similar stratified scheme, keeping the proportion of positives consistent across each split. In total there are 361,045 examples in the dataset, 259,952 of these in the training split, 28,884 in the validation split, and 72,209 in the testing split.

Table 3.1: Description of dataset partitioning

partition	# examples	% active
Train	259,952	2.495
Validation	28,884	2.662
Test	72,209	2.504

Network Details

Our implementation specifies the message passing phase as the result of two functions M_t and U_t, which compute the updated message vector m_v^{t+1} and the update hidden state vector h_v^{t+1}. m_v^{t+1} is specified as the following:

$$m_v^{t+1} = M_t(h_v^t, h_w^t, e_{vw}) = h_v^t \parallel h_w^t \parallel e_{vw} \tag{3.4}$$

where $\parallel$ denotes the concatenation operation. Furthermore, we specify h_v^{t+1} as:

$$h_v^{t+1} = U_t(h_v^t, m_v^{t+1}) = \text{ReLU}(m_v^{t+1}) \tag{3.5}$$

where $ReLU(x) = max(0, x)$. Finally, we specify the readout function R as:

$$R(\{h_v^T \mid v \in G\}) = \sum_{v \in G} \text{ReLU}(h_v^T \parallel h_v^T) \tag{3.6}$$

using consistent definitions for the ReLU non-linearity and concatenation. In the case of classification, the output of the network $\hat{y}$ is defined as:

$$\hat{y} = \text{softmax}(O(R(\{h_v^T \mid v \in G\}))) \tag{3.7}$$

and in the case of regression we define the output $\hat{y}$ as:

$$\hat{y} = O(R(\{h_v^T \mid v \in G\})) \tag{3.8}$$

where O is a linear transformation. We explore a number of configurations for the network parameters, using a randomized hyperparameter search to efficiently explore the hypothesis space. All code implemented using the PyTorch deep learning framework and deepchem molecular machine learning library [28], [1].

Network Training

To reduce the time for the MPNN to fit our dataset, we use the HOGWILD! distributed stochastic gradient descent (SGD) optimization algorithm [31]. HOGWILD! is an asynchronous implementation of SGD that trains a model in a distributed fashion by using multiple processes to update the parameters of a single shared copy of the model. The updates are lock-free, in that the processes are free to update the shared copy after computing the gradient for their respective batches and that there is no central authority that must approve of the updates. This alleviates issues that may come with a synchronous algorithm, whose overall computation time per mini-batch would depend on the computation time of the slowest worker. While it is possible that the processes overwrite each others work during the training of the network, it has been shown under the assumption that updates to the shared parameters are sparse, that one can achieve a near linear increase in convergence. We provide a visualization of this optimization algorithm in 3.1.

To give an overview of the HOGWILD! implementation, we initialize n training processes where each process gets a copy of the data, a reference to the shared memory model, and a unique random seed. Then each process begins its own SGD, treating the shared memory parameters as their own. The processes make updates and get the new values of the model parameters asynchronously. Thus after the initial copy, it is possible that the individual training processes are operating with different, but related, copies of the model parameters, introducing noise to the network training which can be understood as a form of regularization.
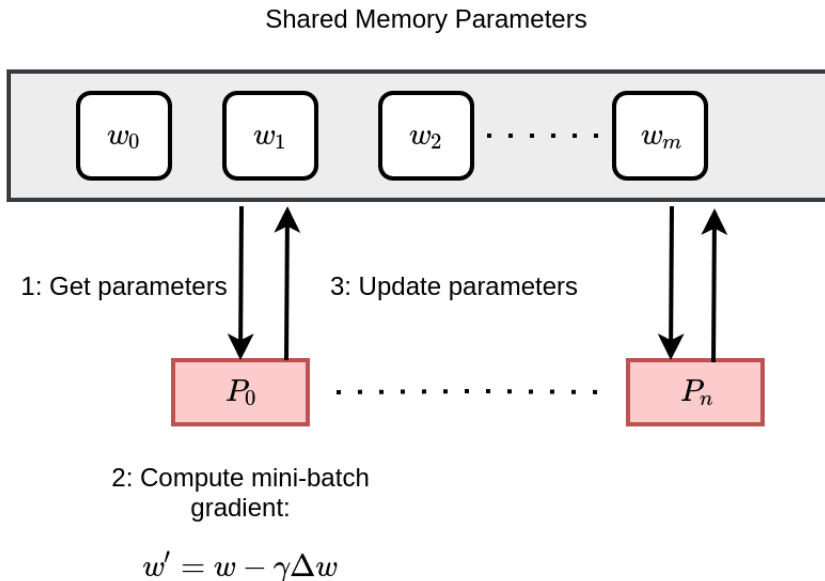
Figure 3.1: Visualization of the Hogwild! training algorithm

3.4   Results

Experiment 1: Predicting Molecular Properties

We first evaluate our method by predicting molecular properties that have been computed by the Dragon software suite [36]. The goal of this experiment was to understand how well the method could learn these properties in order to determine whether more complicated tasks would even be possible. To evaluate these results, we use the *pearson correlation coefficient*, or R^2 metric, which measures how well a predictive model's output correlates to the true target output.

Each network was trained for a separate task using a single training process with batch size of 300, using the ADAM optimizer with learning rate of $1e-3$, readout function R which takes as input a vector of size 128 and computes an output vector of size 140, and output layer O which takes as input a vector of size 128 and computes an output scalar. The target values for each task were standardized using the mean and standard deviation of the given feature column in the training set. The results on each of the separate task are measured given in figure 3.2.
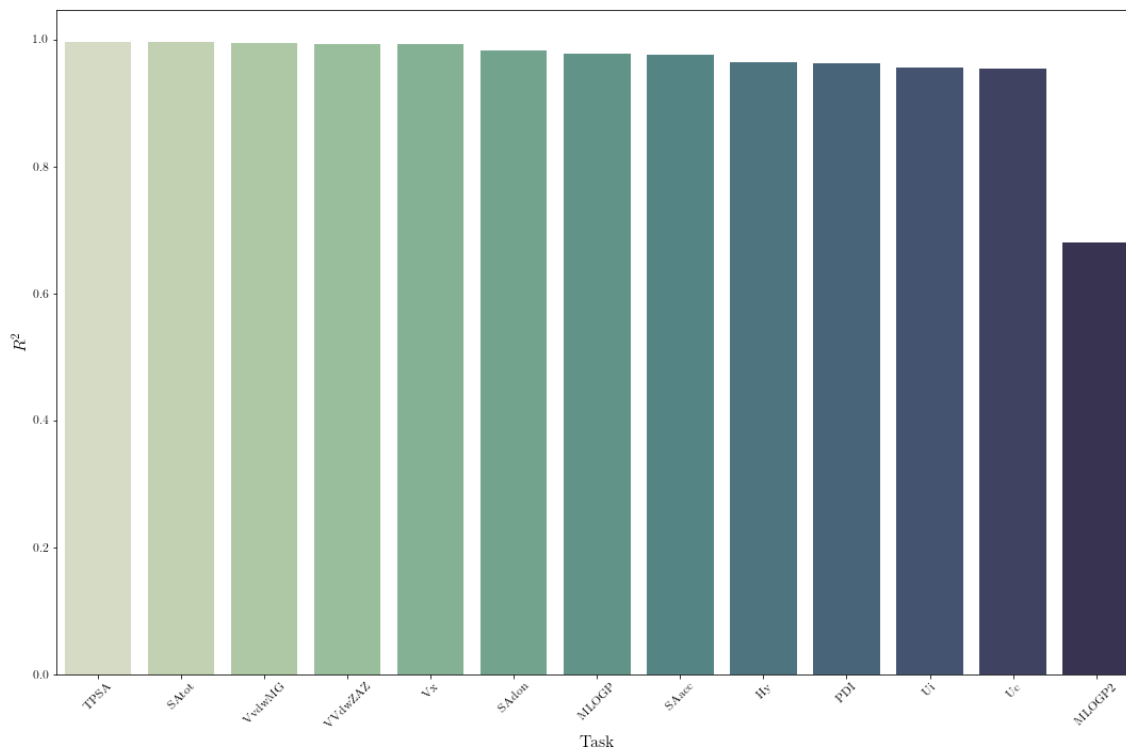
Figure 3.2: R^2 values for molecular property predictions on testing set

Experiment 2: Classification of Kinase Inhibitors

This task attempts to determine whether or not a given drug molecule inhibits a protein kinase target, a binary classification task. This classification is similar to what was done in the previous chapter where we use only drug features in our model 1 to identify the kinase inhibitors. We evaluate the performance of this classification using the precision, recall, and f1-score for each class to understand how well the model is able to correctly predict the active compounds in our dataset.

We use a random hyperparameter search to choose the number of training processes, the size of the output vector computed by readout function R, the size of the input vector for the output layer O, the number of message passing steps T, the batch size, and learning rate to use for training. We sample 100 configurations based upon this, then evaluate each model (for each separate training process) on the test set, choosing the model that gives the best f1-score on the testing set from which we

choose the optimal set of hyperparameters. From this, we then train 5 models with the optimal hyperparameter settings and vary the number of training processes. We then evaluate each of these on our testing set and report the mean metrics (over all training processes) in table 3.2.

Table 3.2: Comparison of kinase inhibitor classification between the MPNN method and Model 1 from the feature selection method.

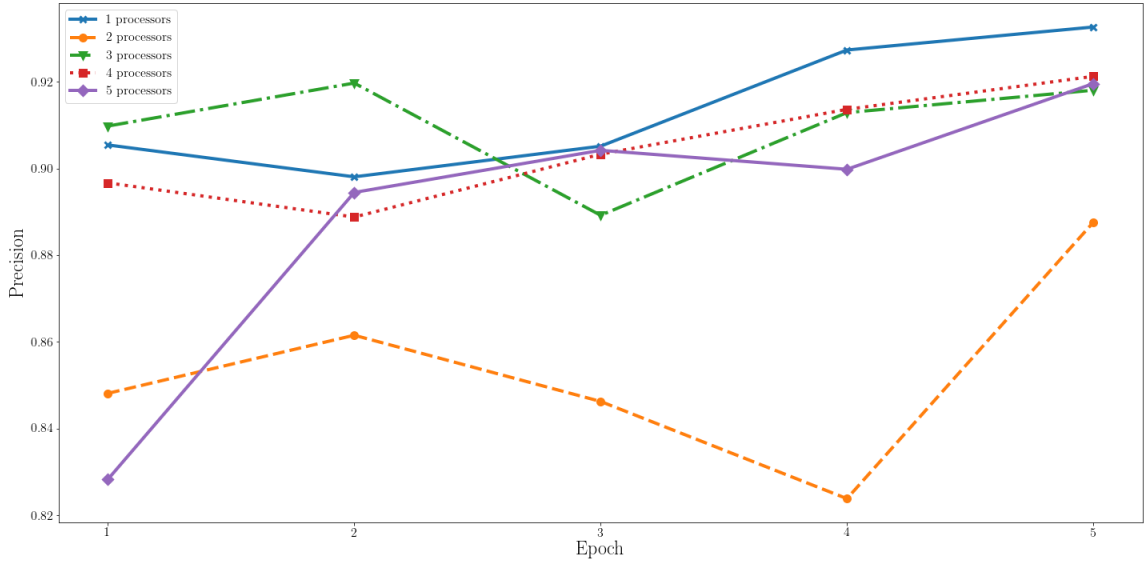| n **processors**	**Class**	**Prec.**	**Recall**	**F1-score**	**Class**	**Prec.**	**Recall**	**F1**
1	0	0.99	1.00	1.00	1	**0.93**	0.68	0.79
2	0	1.00	1.00	1.00	1	0.88	0.88	**0.88**
3	0	0.99	1.00	1.00	1	0.92	0.78	0.84
4	0	1.00	1.00	1.00	1	0.90	0.84	0.87
5	0	1.00	1.00	1.00	1	0.91	0.85	**0.88**
Model 1	0	1.00	1.00	1.00	1	0.83	**0.92**	0.87

Figure 3.3: Mean precision score on validation for each number of training processes, per epoch
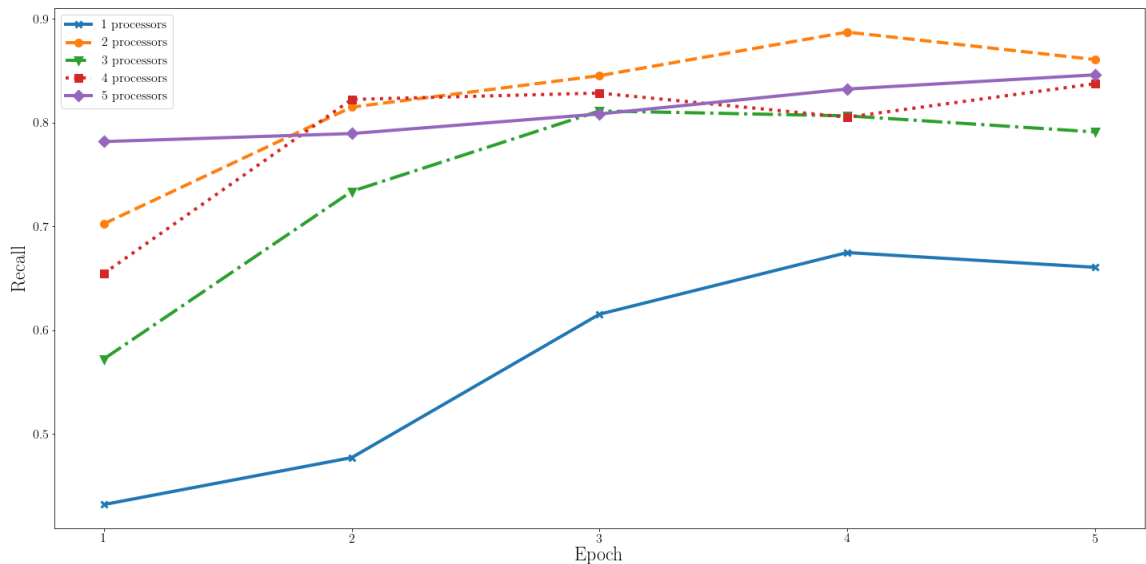
31

Figure 3.4: Mean recall score on validation set for each number of training processes, per epoch
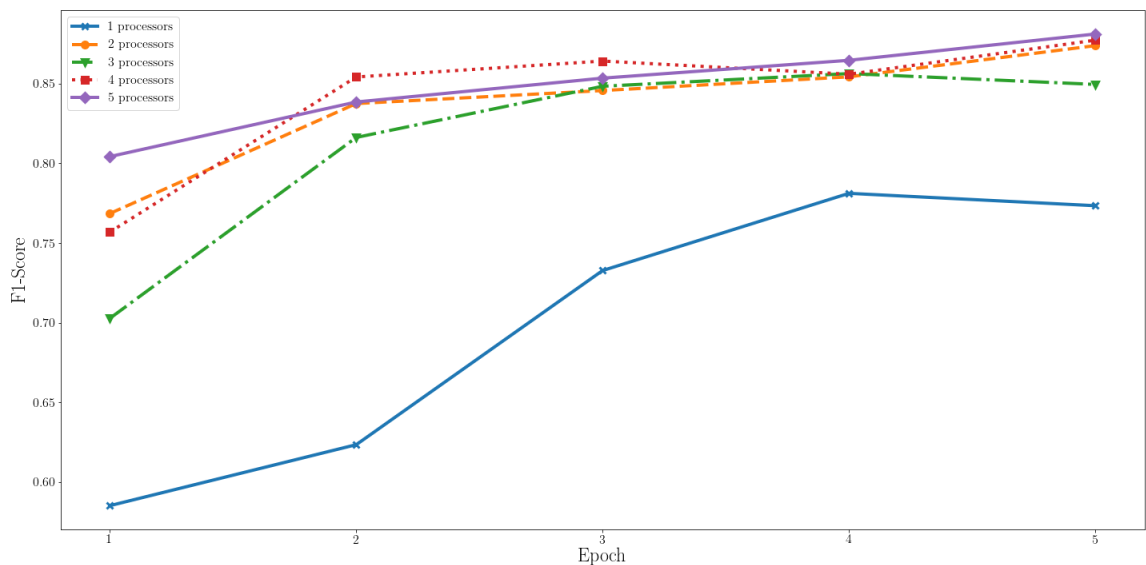

Figure 3.5: Mean f1-score on validation set for each number of training processes, per epoch

3.5   Discussion

Our results show that the MPNN model can be used to classify the kinase inhibitors with performance on par with our previous method that uses pre-computed features. It can also be observed that the use of at least 2 training processes appears to result

in a significant improvement in recall and f1 scores on the test set. This improvement is also achieved earlier in the network training, as seen in figures 3.3, 3.4, and 3.5. However, it does appear that beyond the use of 2 training processes, the improvements are not as substantial. This may be related to the number of model parameters and the sparsity of the updates from each of the training processes.

Chapter 4

Conclusions and Future Directions

In this thesis, we have presented two approaches for the modeling of interactions between drug-like molecules and protein kinase targets. Both of these approaches prioritize the task of extracting information directly from the available data rather than relying on expert knowledge. Additionally, these approaches are distributed in nature, allowing them to scale to the available resources, however large or small those may be. Most importantly, we have shown that these approaches are robust to the class imbalance within our dataset, giving confidence that these algorithms would be successful in identifying active compounds which could potentially lead to adverse reactions.

Future directions for research include the modeling of more complex tasks such as the binding affinities between the protein targets and drug molecules. In addition to this, it would also be interesting to learn multi-output models that quantify the uncertainty associated with each prediction, especially interesting in the case of regression models. Further work with larger datasets and distributed optimization algorithms would be another worthwhile direction to pursue.

Bibliography

1. Democratizing deep-learning for drug discovery, quantum chemistry, materials science and biology. `https://github.com/deepchem/deepchem`, 2016.

2. Ali Anaissi, Paul J Kennedy, Madhu Goyal, and Daniel R Catchpoole. A balanced iterative random forest for gene selection from microarray data. *BMC Bioinformatics*, 14:261, August 2013.

3. Amos Bairoch, Rolf Apweiler, Cathy H Wu, Winona C Barker, Brigitte Boeckmann, Serenella Ferro, Elisabeth Gasteiger, Hongzhan Huang, Rodrigo Lopez, Michele Magrane, et al. The universal protein resource (uniprot). *Nucleic acids research*, 33(suppl_1):D154–D159, 2005.

4. Joanne Bowes, Andrew J Brown, Jacques Hamon, Wolfgang Jarolimek, Arun Sridhar, Gareth Waldron, and Steven Whitebread. Reducing safety-related drug attrition: the use of in vitro pharmacological profiling. *Nature reviews. Drug discovery*, 11(12):909, 2012.

5. Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, 1 October 2001.

6. Dong-Sheng Cao, Yi-Zeng Liang, Jun Yan, Gui-Shan Tan, Qing-Song Xu, and Shao Liu. PyDPI: freely available python package for chemoinformatics, bioinformatics, and chemogenomics studies. *J. Chem. Inf. Model.*, 53(11):3086–3096, November 2013.

7. George E Dahl, Navdeep Jaitly, and Ruslan Salakhutdinov. Multi-task neural networks for QSAR predictions. June 2014.

8. Joseph A DiMasi, Henry G Grabowski, and Ronald W Hansen. Innovation in the pharmaceutical industry: New estimates of R&D costs. *J. Health Econ.*, 47:20–33, May 2016.

9. David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alan Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In C Cortes, N D Lawrence, D D Lee, M Sugiyama, and R Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2224–2232. Curran Associates, Inc., 2015.

10. Sally R Ellingson, Jeremy C Smith, and Jerome Baudry. VinaMPI: Facilitating multiple receptor high-throughput virtual docking on high-performance computers. *J. Comput. Chem.*, 34(25):2212–2221, September 2013.

11. Elisabeth Gasteiger, Christine Hoogland, Alexandre Gattiker, S'everine Duvaud, Marc R Wilkins, Ron D Appel, and Amos Bairoch. Protein identification and analysis tools on the ExPASy server. In John M Walker, editor, *The Proteomics Protocols Handbook*, pages 571–607. Humana Press, Totowa, NJ, 2005.

12. Anna Gaulton, Louisa J Bellis, A Patricia Bento, Jon Chambers, Mark Davies, Anne Hersey, Yvonne Light, Shaun McGlinchey, David Michalovich, Bissan Al-Lazikani, et al. Chembl: a large-scale bioactivity database for drug discovery. *Nucleic acids research*, 40(D1):D1100–D1107, 2011.

13. Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. April 2017.

14. Kun-Yi Hsin, Samik Ghosh, and Hiroaki Kitano. Combining machine learning systems and multiple docking simulation packages to improve docking prediction reliability for network pharmacology. *PloS one*, 8(12):e83922, 2013.

15. J P Hughes, S Rees, S B Kalindjian, and K L Philpott. Principles of early drug discovery. *Br. J. Pharmacol.*, 162(6):1239–1249, March 2011.

16. William Humphrey, Andrew Dalke, and Klaus Schulten. Vmd: visual molecular dynamics. *Journal of molecular graphics*, 14(1):33–38, 1996.

17. John J Irwin, Teague Sterling, Michael M Mysinger, Erin S Bolstad, and Ryan G Coleman. Zinc: a free tool to discover chemistry for biology. *Journal of chemical information and modeling*, 52(7):1757–1768, 2012.

18. Ali Akbar Jamali, Reza Ferdousi, Saeed Razzaghi, Jiuyong Li, Reza Safdari, and Esmaeil Ebrahimie. DrugMiner: comparative analysis of machine learning algorithms for prediction of potential druggable proteins. *Drug Discov. Today*, 21(5):718–724, May 2016.

19. Sarah L Kinnings, Nina Liu, Peter J Tonge, Richard M Jackson, Lei Xie, and Philip E Bourne. A machine learning-based method to improve docking scoring functions and its application to drug repurposing. *J. Chem. Inf. Model.*, 51(2):408–419, February 2011.

20. Radoslav Krivák and David Hoksza. Improving protein-ligand binding site prediction accuracy by classification of inner pocket points using local features. *J. Cheminform.*, 7:12, 1 April 2015.

21. Gilles Louppe, Louis Wehenkel, Antonio Sutera, and Pierre Geurts. Understanding variable importances in forests of randomized trees. In C J C Burges, L Bottou, M Welling, Z Ghahramani, and K Q Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 431–439. Curran Associates, Inc., 2013.

22. Junshui Ma, Robert P Sheridan, Andy Liaw, George E Dahl, and Vladimir Svetnik. Deep neural nets as a method for quantitative structure-activity relationships. *J. Chem. Inf. Model.*, 55(2):263–274, February 2015.

23. Bjoern H Menze, B Michael Kelm, Ralf Masuch, Uwe Himmelreich, Peter Bachert, Wolfgang Petrich, and Fred A Hamprecht. A comparison of random forest and its gini importance with standard chemometric methods for the feature selection and classification of spectral data. *BMC Bioinformatics*, 10:213, July 2009.

24. Claudio Mirabello and Gianluca Pollastri. Porter, PaleAle 4.0: high-accuracy prediction of protein secondary structure and relative solvent accessibility. *Bioinformatics*, 29(16):2056–2058, August 2013.

25. H L Morgan. The generation of a unique machine description for chemical Structures-A technique developed at chemical abstracts service. *J. Chem. Doc.*, 5(2):107–113, May 1965.

26. Hirotomo Moriwaki, Yu-Shi Tian, Norihito Kawashita, and Tatsuya Takagi. Mordred: a molecular descriptor calculator. *J. Cheminform.*, 10(1):4, February 2018.

27. Michael M Mysinger, Michael Carchia, John J Irwin, and Brian K Shoichet. Directory of useful decoys, enhanced (DUD-E): better ligands and decoys for better benchmarking. *J. Med. Chem.*, 55(14):6582–6594, July 2012.

28. Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

29. P G Polishchuk, T I Madzhidov, and A Varnek. Estimation of the size of drug-like chemical space based on GDB-17 data. *J. Comput. Aided Mol. Des.*, 27(8):675–679, August 2013.

30. Bharath Ramsundar, Bowen Liu, Zhenqin Wu, Andreas Verras, Matthew Tudor, Robert P Sheridan, and Vijay Pande. Is multitask deep learning practical for pharma? *J. Chem. Inf. Model.*, 57(8):2068–2076, August 2017.

31. Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild: A Lock-Free approach to parallelizing stochastic gradient descent. In J Shawe-Taylor, R S Zemel, P L Bartlett, F Pereira, and K Q Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 693–701. Curran Associates, Inc., 2011.

32. Ian Roberts, Irene Kwan, Phillip Evans, and Steven Haig. Does animal experimentation inform human healthcare? observations from a systematic review of international animal experiments on fluid resuscitation. *BMJ: British Medical Journal*, 324(7335):474, 2002.

33. David Rogers and Mathew Hahn. Extended-Connectivity fingerprints. *J. Chem. Inf. Model.*, 50(5):742–754, May 2010.

34. Jack W Scannell, Alex Blanckley, Helen Boldon, and Brian Warrington. Diagnosing the decline in pharmaceutical R&D efficiency. *Nat. Rev. Drug Discov.*, 11(3):191–200, March 2012.

35. Tony Siu, Jun Liang, Jeannie Arruda, Yiwei Li, Raymond E Jones, Deborah Defeo-Jones, Stanley F Barnett, and Ronald G Robinson. Discovery of potent and cell-active allosteric dual akt 1 and 2 inhibitors. *Bioorganic & medicinal chemistry letters*, 18(14):4186–4190, 2008.

36. Kode srl. *Dragon (software for molecular descriptor calculation) version 7.0.6*, 2016. `https://chm.kode-solutions.net`.

37. Roberto Todeschini and Viviana Consonni. *Handbook of Molecular Descriptors*. John Wiley & Sons, July 2008.

38. Oleg Trott and Arthur J Olson. Autodock vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multi-threading. *Journal of computational chemistry*, 31(2):455–461, 2010.

39. Peng Zhang, Lin Tao, Xian Zeng, Chu Qin, Shangying Chen, Feng Zhu, Zerong Li, Yuyang Jiang, Weiping Chen, and Yu-Zong Chen. A protein network descriptor server and its use in studying protein, disease, metabolic and drug targeted networks. *Brief. Bioinform.*, August 2016.

Vita

Derek Jones

Education

2011–2016   B.S. in Computer Science   University of Kentucky

Dual Major in Mathematical Economics

Appointments

Summer Student Intern   Lawrence Berkeley National Laboratory

06/2017–08/2017   *Lexington, KY*

Research Assistant   Markey Cancer Center, University of Kentucky

01/2017–present   *Lexington, KY*

Teaching Assistant   Department of Computer Science, University of Kentucky

08/2016-05/2017   *Lexington, KY*

Multimedia Production Consultant   Media Depot, University of Kentucky

08/2015–08/2016   *Lexington, KY*

Publications

Conference Proceedings

1. Derek Jones, Jeevith Bopaiah, Fatemah Alghamedy, Nathan Jacobs, Heidi L Weiss, W A de Jong, and Sally R Ellingson. Polypharmacology within the full kinome: a machine learning approach. In *AMIA 2018 Informatics Summit*, 2018.

2. Fatemah Alghamedy, Jeevith Bopaiah, Derek Jones, Xiaofei Zhang, Heidi L Weiss, and Sally R Ellingson. Incorporating protein dynamics through ensemble docking in machine learning models to predict drug binding. In *AMIA 2018 Informatics Summit*, 2018.

Poster Sessions

1. Derek Jones, Sally R Ellingson, and W A de Jong. How low can you go? feature selection for drug discovery. In *Commonwealth Computational Summit*, 2017.

Honors and Awards

- Supercomputing (SC) 2017 Student Volunteer Travel Award, 2017

- CRA Computing Sciences Research Pathways Fellowship (LBNL), $7,500, 2017

- Lyman T. Johnson Diversity Fellowship, $7,500, 2017

- AAAI 2017 Scholarship, $350, 2017

Volunteering Experience

- 2017, Supercomputing (SC) 2017 Student Volunteer

- 2017, AAAI 2017 Student Volunteer