Ligand-Protein Binding Affinity Prediction Using Meta-Learning

Alperen Bağ

Advisor: Arzucan Özgür

Department of Computer Engineering

Bogazici University

12.02.2021

CMPE 492 & Spec. Project In Computer Engineering

Mahdi Fazeli

Abstract

The ligand-target affinity prediction problem is a very important problem to solve in order to quicken the drug discovery process. This problem can be considered as a classification problem where the purpose is to determine whether a ligand and a protein bind each other or not, or as a regression problem where the purpose is to predict the strength of the binding as a numerical value. As a solution, we can benefit from recent developments in artificial intelligence and deep learning. So far, several works have tried to solve this problem using various machine learning methods. However, because of some downsides of the datasets used in this field, it is observed that machine learning models do not generalize well in the test data. The issues related to the datasets are discussed in detail in (3.1. Data).

In this work, we propose that we can benefit from the meta-learning algorithms to overcome the negative effect of these downsides and train a model that generalizes well outside the training distribution. Specifically, we applied the MAML algorithm in our work to train the affinity predictor model. MAML is a method that makes the model adaptive to the situations that it has not seen during the train. In particular, we use MAML++ method which was claimed to be more stable than MAML thanks to some useful tricks, hence we preferred MAML++. Our code is available in (https://github.com/alprnbg/DeepDTA-MAML).

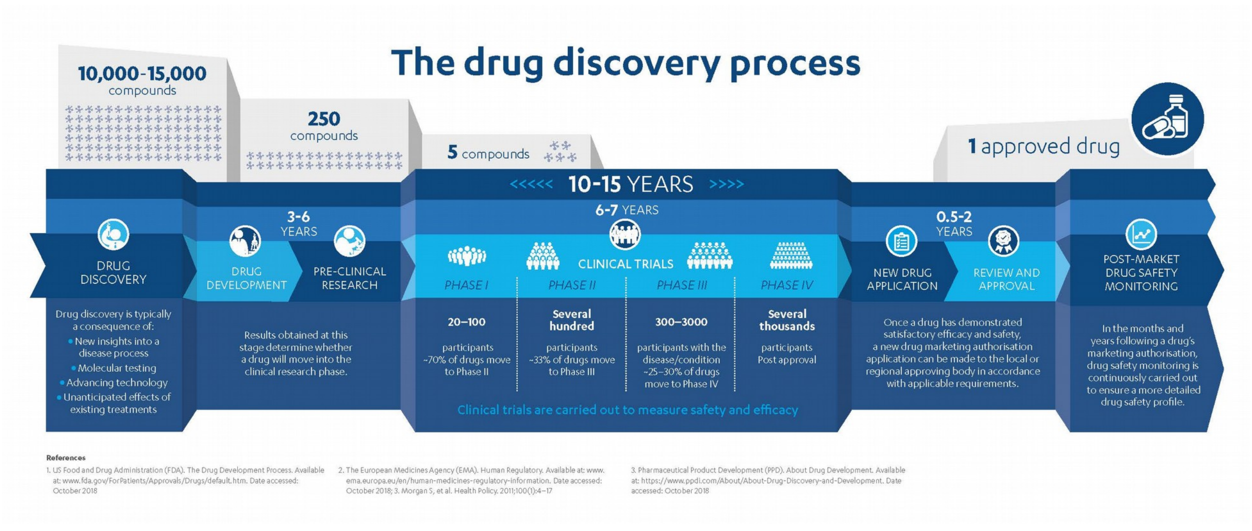
Table of Contents

List of Figures

List of Tables

1 Introduction

Drug discovery is a very crucial topic of science today. With the increase in population and wealth, many states and companies started to invest billions of dollars in drug discovery research to provide efficient and strong treats for many illnesses. However, drug discovery requires lots of money, lots of human-power, and lots of time, since many laboratory experiments are needed to design a powerful drug for a specific illness. According to a web article[1], it takes 12-15 years on average to design a new drug and get approval from FDA. Again according to the same source, this development process costs around 1$ billion.


The drug discovery process
10,000-15,000 compounds
250 compounds
5 compounds
1 approved drug
<<<<< 10-15 YEARS >>>>
3-6 YEARS
6-7 YEARS
0.5-2 YEARS
DRUG DISCOVERY
DRUG DEVELOPMENT
PRE-CLINICAL RESEARCH
CLINICAL TRIALS
PHASE I
PHASE II
PHASE III
PHASE IV
NEW DRUG APPLICATION
REVIEW AND APPROVAL
POST-MARKET DRUG SAFETY MONITORING
20–100 participants ~70% of drugs move to Phase II
Several hundred participants ~33% of drugs move to Phase III
300–3000 participants with the disease/condition ~25–30% of drugs move to Phase IV
Several thousands participants Post approval

Drug discovery is typically a consequence of:
- New insights into a disease process
- Molecular testing
- Advancing technology
- Unanticipated effects of existing treatments

Results obtained at this stage determine whether a drug will move into the clinical research phase.

Clinical trials are carried out to measure safety and efficacy

Once a drug has demonstrated satisfactory efficacy and safety, a new drug marketing authorisation application can be made to the local or regional approving body in accordance with applicable requirements.

In the months and years following a drug's marketing authorisation, drug safety monitoring is continuously carried out to ensure a more detailed drug safety profile.

References
1. US Food and Drug Administration (FDA). The Drug Development Process. Available at: www.fda.gov/ForPatients/Approvals/Drugs/default.htm. Date accessed: October 2018
2. The European Medicines Agency (EMA). Human Regulatory. Available at: www.ema.europa.eu/en/human-medicines-regulatory-information. Date accessed: October 2018; 3. Morgan S, et al. Health Policy. 2011;100(1):4–17
3. Pharmaceutical Product Development (PPD). About Drug Development. Available at: https://www.ppdi.com/About/About-Drug-Discovery-and-Development. Date accessed: October 2018

Illustration 1: Drug discovery stages [2]

A typical drug discovery process consists of 5 stages[2]. Discovery, development and preclinical research, clinical development, FDA review, and FDA post-market safety monitoring. We will explain only the first two stages in this section since clinical experiments and the FDA approval process are not related to this work. Discovery and development stage is the first step of designing a new drug. In this stage, first of all, a target should be found for the disease against which our drug will be used. This target can be a gene or a protein that plays an important role in that disease. This step might require in vitro experiments and some screening assays. In this work, target identification is not examined, that is to say, we assume that we already know our target protein. After the target identification and validation, lots of possible drug-candidate molecules are determined. In this stage, the number of candidates is around 10,000-15,000[2]. In the second stage, which is the development and preclinical research, various measurements and experiments are done to eliminate bad candidates. In this stage, the binding affinity between the candidate ligand and the target protein is also measured. A good drug should bind the target strongly so that it can disrupt the activity of the protein, thereby treating the disease. In this work, we want to make the binding affinity measurement process faster and scalable so that it will be possible to eliminate many bad candidates in the early stages of the drug discovery process, therefore the cost and the necessary time will reduce dramatically. This can lead to easier, faster, and better drug discovery procedures, thereby improving healthcare on a global scale.

In this work, we only focus on the binding affinity prediction problem. Other stages in the drug discovery process are beyond the scope of this work. In short, we propose a meta-learning model that predicts the binding affinity between a protein and a target quickly and cheaply. The details of this model are explained in (3.2. Model).

2 Background

Recently, there are a lot of works that try to predict binding affinity between proteins and ligands[3,4,5,6,7,8,14,15,16]. Some of them consider this problem as classification[14,15,16] and the others approach the problem as regression problem[3,4,5,6,7,8]. Although classifying interactions as whether protein-ligand pairs bind to each other or not, may give important information for drug-discovery, prediction of binding affinity values give more valuable information which is the strength of the binding. Hence, we also think of this problem as binding affinity value prediction and try to predict pK_d values in filtered BindingDB[17] data set which is also used in some other works[9]. pK_d is the negative logarithm of the dissociation constant which is a measure for the strength of the interaction between protein-target pairs. Moreover, protein family information may be important for this problem because the proteins with the same protein family tend to have similar characteristics and functionalities which may result in similar binding properties[10]. Furthermore, it has been stated that models that are specifically developed for a specific protein family can give better results than a general model[11]. However, usually, there is not a lot of labeled data for only one protein family so it is impractical to use this approach for neural networks. Hence, we make use of protein family information and modeled our problem such that making prediction for different protein families can be considered as different tasks. Since meta-learning approaches have gained popularity for multi-task learning, we used MAML[31]. Successes of MAML approaches in different kinds of fields such as computer vision and NLP have been shown[12,13,31]. Although it is usually used for classification problems, its success in regression problems is also shown[31].

3 Methodology

3. 1. Data

To train and evaluate our model, we used the BindingDB dataset[17] which contains measured binding affinities of many proteins and ligands. As of January 18, 2021, it has 2,108,548 interaction data for 8,197 unique proteins and 925,431 unique ligands. It was curated from various sources and it provides big data for the affinity prediction problem. However, like other binding affinity datasets[18-20], the BindingDB dataset is not a very suitable dataset for deep learning. The reason is that creating a binding dataset is not a simple task, it requires laboratory experiments and validation of the results. Besides, environmental conditions can affect the experiments, which leads to noisy datasets. Another issue is the bias in these datasets, which is examined in Appendix B. In our work, we observed that some proteins, ligands, that are small drug-candidate molecules, and protein families have a good number of interaction data, while some of them have very few data samples. Also, some proteins and ligands have either mostly positive or mostly negative interaction data. Positive means strong binding and negative means weak or no binding in this case. This is another problem preventing the model from generalizing well in a test dataset.

In the Chemboost paper[9], the authors extracted a subset from the BindingDB to overcome noisy data and the bias problem. They took the proteins with 6 or more interaction data and ligands with 3 or more interaction data. If there are multiple affinity values for the same interaction, which is possible since the BindingDB dataset was curated from different sources, they took the one with the highest affinity value. We followed the same filtering procedure to create our dataset so that we can compare our results with theirs. The statistics of the extracted dataset is in Appendix A.

We also created our test dataset using the same procedure as in the Chemboost paper[9]. That is, we created 4 different types of test sets whose names are “warm”, “cold protein”, “cold ligand” and “cold both”. In the “warm" test set, each protein and ligand in the test set appears also in the training set. In the “cold protein" test set, each ligand in the test set appears in the training set, while none of the proteins appear in the training set. The “cold ligand" test set can be considered similar to the “cold protein" test set, except ligands are never seen by the model during the training instead of proteins. In the “cold both" test set, the model tries to predict the binding affinity between a protein and a ligand that it has never seen during the training. In addition to these datasets, we also create a new test set called “cold protein family" that may or may not contain the ligand in the training set but does not contain any protein from protein families in this set. It should be noted that this dataset is not disjoint from the others, and it is extracted from “cold both" and “cold protein" datasets. The “cold both" test set is the most difficult test set in this setup and the model should generalize well to perform accurately in this test set.

3. 1. 1. Measurement of Binding Strength

In the BindingDB dataset and many other binding datasets, binding strengths are measured as K_d and K_i values, in general. Their names are the equilibrium dissociation constant and the inhibitor constant, respectively. The equilibrium reaction that K_d governs can be written as below.

$$PL \Leftrightarrow P + L$$

In this reaction, proteins and ligands bind to each other to create PL (protein-ligand complex) and PL dissociate into its components that are a protein (P) and a ligand (L). This equilibrium is governed by K_d. The higher K_d is, the more PL complexes dissociate into its components. Therefore, when the K_d is high, we can say that the binding affinity between the protein and the ligand is low, and vice versa.

The meaning of K_i constant is more complex and beyond this work. We did not use K_i values in our dataset, since it depends on some other factors other than the ligand and the protein[21]. Therefore, we filtered the BindingDB to include the interactions with a K_d value only. Finally, we converted K_d to pK_d using the equation below.

$$pK_d = -\log_{10}\left(\frac{K_d}{1e9}\right)$$

After this transformation, we obtained a more sensible distribution and now higher pK_d means stronger binding affinity. We can consider the interaction with $pK_d > 7$ as strong interaction, and the interaction with $pK_d < 7$ as weak interaction, according to the paper[22]. The pK_d histogram of the dataset is can be found in Appendix A.

3. 1. 2. Representation of Molecules

3. 1. 2. 1. Ligands

The Binding DB dataset represents ligands in SMILES[23] notation that is a widely accepted notation format for small molecules. SMILES is the abbreviation of “simplified molecular-input line-entry system” which was introduced by Weininger in 1988. The purpose of the SMILES notation is to represent ligands as a string instead of a 2D graph. To convert 2D graph notation to SMILES, we should start from a node, that is an atom in this case, and traverse the graph following the rules set by SMILES notation. This traverse will result in a string that can be processed by neural networks after some preprocessing steps. In illustration 2, (A) is the 2D graph notation and (D) is the SMILES string of Ciprofloxacin. However, a unique molecule can have multiple SMILES representations depending on the starting node of the traversal. To have unique SMILES representations, we used the canonical SMILES representation of ChEMBL[24]. Our model takes the SMILES representation of the input ligand as a string and an embedding layer outputs an embedding vector for each character in the SMILES notation. With this preprocessing, our model learns how to process SMILES notation.

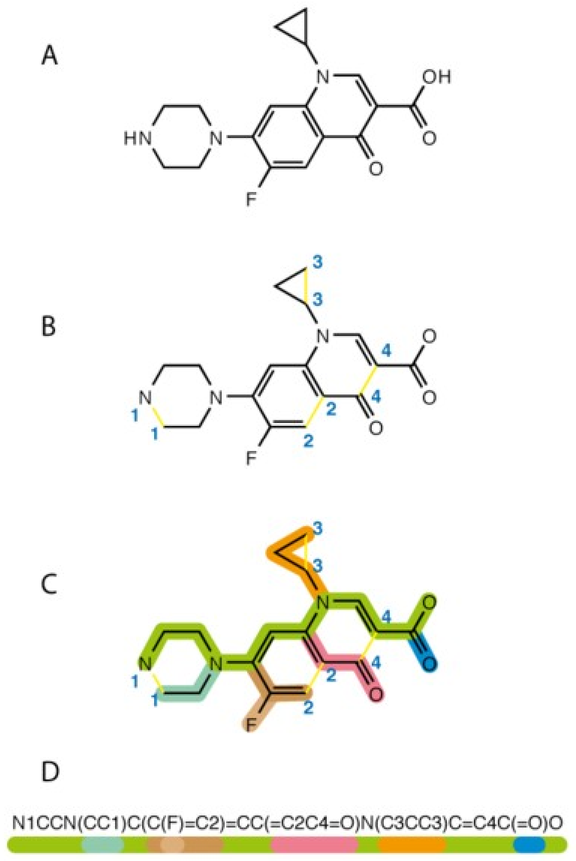
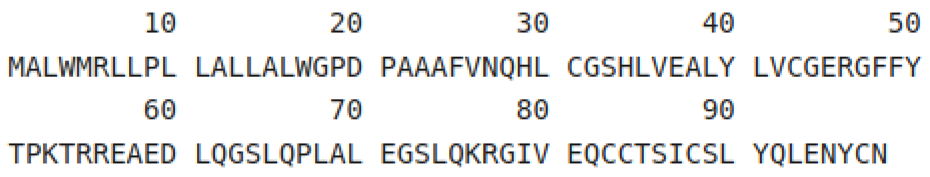


Illustration 2: The process of converting 2D graph notation to SMILES notation for Ciprofloxacin[25]

3. 1. 2. 2. Proteins

For proteins, the FASTA[26] format is a very common notation and the BindingDB also adapts this format for proteins. Basically, each nucleotide in a protein chain is represented by a character in FASTA format. Therefore, we can represent a protein as a string with FASTA notation. In illustration 3, 98-character long FASTA notation of the human insulin enzyme can be seen.



```
        10         20         30         40         50
MALWMRLLPL LALLALWGPD PAAAFVNQHL CGSHLVEALY LVCGERGFFY
        60         70         80         90
TPKTRREAED LQGSLQPLAL EGSLQKRGIV EQCCTSICSL YQLENYCN
```

Illustration 3: Human insulin enyzme (FASTA) [27]

3. 2. Model

3. 2. 1. DeepDTA

DeepDTA[4] is one of the first methods trying to utilize deep learning to solve the drug-target affinity prediction problem. Although it uses a basic convolutional neural network architecture that has two input branches and one output branch, it can be still considered as SOTA. Basically, it takes a protein and a ligand as inputs. The inputs are represented as a sequence of characters. The format of the inputs is explained in detail in (3.1.2. Representation of Molecules). At the beginning of the network, there are 2 identical branches, one for the protein and one for the ligand. Each branch is as a sequence of an embedding layer, three 1D-convolutional layers and a max-pooling layer. The embedding layers are used to convert given characters to a numerical value that can be processed by the convolutional layers afterward. The outputs of the branches are concatenated, and then the concatenated output is fed through fully-connected layers that produce the final result. The illustration of the DeepDTA architecture can be found in Appendix B.

We used a similar architecture to DeepDTA, but with some modifications to get better results. The first update to the model is the addition of batch normalization layers. Batch normalization[28] is a widely accepted technique that is used in deep learning models. It is practically proven that using batch normalization leads to better training most of the time. Therefore, after each convolutional layer, we inserted a batch normalization layer. The second update is the addition of the dropout mechanism[29]. Dropout is also a commonly accepted and used technique in the training of deep learning models and it leads models to generalize well in the test datasets. Basically, dropout layers select some activation nodes randomly and make them zero during the forward pass. With this technique, it is shown that dropout layer performs as a good regularizer and prevents over-fitting[29]. Hence, we inserted a dropout layer after each activation function. The last update to the model is that we changed ReLU activations with leaky ReLU activations[30]. With this change, we aim to reduce the probability of the vanishing gradient problem during the training. In our experiments, we used the modified model which is illustrated in Appendix B. We name this architecture DeepDTA v2.

3. 2. 2. Meta-Learning

We trained our model in the meta-learning framework. The reason is to have a model that can easily adapt to different types of proteins and perform well in the “cold both” and “cold protein” test sets.

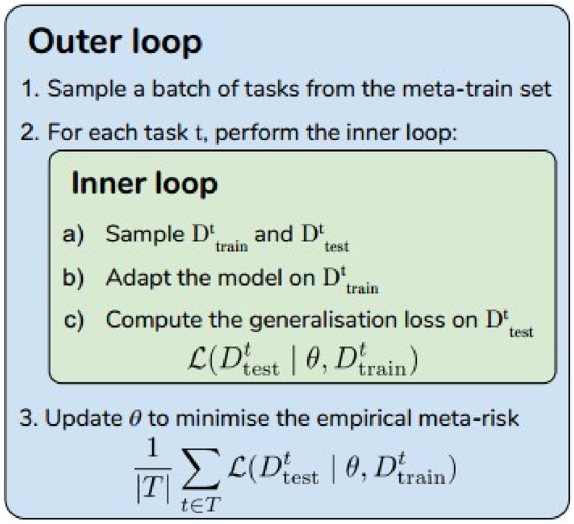
Firstly, meta-learning in the machine learning domain means learning how to learn. During the meta-learning, the model learns how to adapt to different tasks quickly. In other words, it learns how to learn. For example, we can think of a person who knows how to cycle. For that person, learning how to ride a motorcycle should not be a hard process, and after a few trials, they should easily adapt to the motorcycle. The reason is that cycling and motorcycling have a lot in common and this person can use their experience on cycling to learn how to ride a motorcycle. In our case, we consider each protein family as a new task and we want our model to adapt to the protein family of the input protein during the inference. To realize a meta-learning system for our model, we employ the improved version of the MAML[31] algorithm, that is MAML++[32]. First of all, we want to explain the fundamentals of the MAML algorithm and we will explain the additions that lead to the MAML++ algorithm later on.

3. 2. 2. 1. MAML

MAML[31] (Model Agnostic Meta-Learning) is one of the most known algorithms in the meta-learning field and it is easy to apply to any deep learning setup since it is model agnostic. In other words, we can use any deep learning model in MAML training. However, the training procedure is different from the classical deep learning training where we sample some data points in each iteration and calculate the loss for these samples, then update the model with respect to calculated gradients. In each iteration, the model processes samples whose number is equal to the batch size. In MAML, we have 2 loops in each iteration, the outer loop and the inner loop (see illustration 4). In the first step of the outer loop, we sample as many tasks, that is protein families in our case, as the batch size. In the second step, we iterate through each task denoted by t, which starts the inner loop optimization. In the inner loop, we first sample a support set D_{train}^{t} and a target set D_{test}^{t} for task t. Our model will adapt itself to task t with the help of the support set which contains data samples and ground truths for that task. This adaptation process can be considered as a very quick fine-tuning process which is denoted by step (b) in illustration 4. The adaptation process is a couple of training iterations in general. In illustration 5, the adaptation process is illustrated by the dashed lines. ∇L_1, ∇L_2, and ∇L_3 represent the losses coming from the adaptation process for task 1, task 2, and task 3 respectively. After the adaptation, the model makes predictions for the target set and the gradient calculation is done with respect to the loss coming from these predictions, which is step (c) in illustration 4. Finally, the gradients obtained from the target set predictions are used for the outer loop optimization, which is denoted by step 3. In this step, we update our theta to move it to a better place where it can easily adapt to new tasks after the adaptation process. The update of θ requires second-order derivatives which increase the training time compared to the classical training. During the inference time, we use a support set to make our model adapt to the desired task, then do the normal inference for the target set.

In our work, we defined each protein family as a task and want our model to adapt the desired protein family before making predictions. During the training, we sample 3 negative and 3 positive interactions as our support set. The target set contains only one sample. As we mentioned before, we assume[22] that the interaction is weak, that is negative, when pK_d < 7, and strong, that is positive, when pK_d > 7. Since each sample in the support and the target set should come from the same task, each protein in these sets belongs to the same protein family.

As mentioned, we did not use the original MAML algorithm since it suffers from some instability issues[32], we preferred to employ the MAML++ algorithm which was shown[32] to be more stable thanks to some tricks.



```
Outer loop
1. Sample a batch of tasks from the meta-train set
2. For each task t, perform the inner loop:
    Inner loop
    a) Sample $D^t_{train}$ and $D^t_{test}$
    b) Adapt the model on $D^t_{train}$
    c) Compute the generalisation loss on $D^t_{test}$
       $\mathcal{L}(D^t_{\text{test}} \mid \theta, D^t_{\text{train}})$
3. Update $\theta$ to minimise the empirical meta-risk
   $\frac{1}{|T|} \sum_{t \in T} \mathcal{L}(D^t_{\text{test}} \mid \theta, D^t_{\text{train}})$
```

Illustration 4: Pseudocode of the MAML training procedure [33]

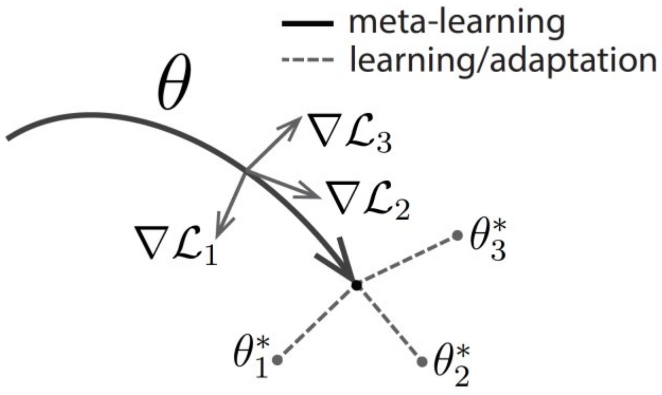


Illustration 5: Illustration of MAML[31]

3. 2. 2. 2. MAML++

In the paper[32], the authors observed that MAML training is very unstable and they proved this by training MAML models with different random seeds which affect the initialization of the model (see illustration 6). To solve this problem, they proposed a couple of tricks which we also utilized in our work.

Firstly, we used directly second-order derivatives instead of the first-order approximation proposed by the MAML paper[31]. In MAML++, the authors suggested using first-order derivates until the 50th epoch, then switch to second-order derivative for better convergence. In our work, we used second-order derivatives from the beginning which increased the training time but led to better optimization.

Secondly, we used LSLR (Learning Per-Layer Per-Step Learning Rates and Gradient Directions) which was proposed by the MAML++ paper[32]. Simply, this method learns the learning rate and gradient direction for each layer in the network during the adaptation process, that is inner-loop optimization. We initialize the learning rates as 1e-3, then let the model learn the learning rates itself. Lastly, we utilized “Cosine Annealing” for the outer loop optimization.

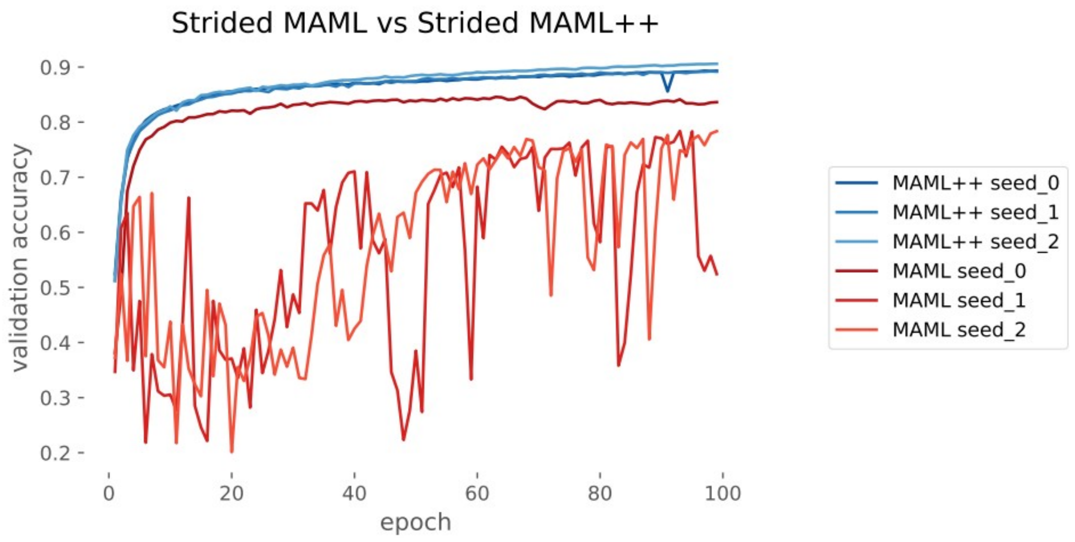


Illustration 6: MAML vs MAML++ [32]

3. 2. 3. Gradient Clipping

During the training, we observed high gradients than the normal for the dense layers. This eventually led to infinite losses and made optimization impossible. To overcome this issue, we apply gradient clipping to both inner-loop and outer-loop optimizations. We clipped each gradient between -1 and 1. After applying gradient clipping, we are able to train our model properly.

3. 3. Evaluation Metrics

For the evaluation of the models, we calculate the mean-squared error loss (MSE) and concordance index (CI)[34] on the test set. While MSE is simply the square of the difference between the ground truth and the predicted value, CI is defined as below. Lower MSE and higher CI are better in our case.

$$CI = \frac{1}{z} \sum_{\delta_x > \delta_y} h\left(b_x - b_y\right)$$

In this equation, δ_x and δ_y represents the ground truth binding affinity values for interaction x and y., while b_x and b_y are the predicted values for these interactions respectively. Lastly, Z is the normalization constant and h is the step function.

3. 4. Experimental Setup and Tools

For our experiments, we benefited from the original repository[35] of the MAML++ paper which was implemented using PyTorch[36]. We implemented our model and made necessary modifications to the code. Our code is open-source and available (https://github.com/alprnbg/DeepDTA-MAML).

We trained our model using SGD (stochastic gradient descent) optimizer with a learning rate of 1e-3 for the inner loop optimization, and Adam[37] optimizer with a learning rate of 1e-4 for the outer loop optimization. We apply cosine annealing to the outer loop optimization for stability[32]. We trained our model with a batch size of 8 and 100 epochs in total. Each epoch consists of 500 iterations. All of our experiments were done with GTX 1060 provided by BOUN TABILAB.

In our experiments, we trained the DeepDTA v2 in two different ways. Firstly, we trained the model in classical fashion, that is, without using MAML++. Secondly, we first trained it in classical fashion for a while and used the weights coming from this train as pre-trained weights for our proposed method. Former is denoted as "DeepDTA v2", while latter is denoted as "DeepDTA v2 (MAML++)". In both trains, we apply cross-validation with 5 splits. All given results in the next section (4. Results and discussion) is the average of that metric for all splits.

4 Results and discussion

As result, our models, DeepDTA v2 and DeepDTA v2 (MAML++), performed well than DeepDTA v1[4] in all cold test sets. However, the performance is slightly worse in the warm test set. This shows that our proposed modifications reduced the over-fitting and increase the generalizability of the model, which results in an improvement in the cold test performance. Since the warm test set resembles the train set and we wanted to make the model more generalizable, worse performance in the warm test was expected actually.

The mean, minimum, maximum and standard deviation of MSE and CI metrics for DeepDTA v2 and DeepDTA v2 (MAML++) can be seen in table 1. We can see from the table that DeepDTA v2 (MAML++) is better in the cold protein family, the cold both and the cold ligand test sets, while DeepDTA v2 is very slightly better in the cold protein. Therefore, we can conclude that applying meta-learning improved the performance for the cold test sets.

Model	Validation Set	MSE	CI
DeepDTA v2	Warm	Mean: **0.409** Min: 0.337, Max: 0.570, Std: 0.092	Mean: **0.864** Min: 0.832, Max: 0.888, Std: 0.022
DeepDTA v2 (MAML++, pretrained)	Warm	Mean: 0.476 Min: 0.413, Max: 0.563, Std: 0.058	Mean: 0.843 Min: 0.857, Max: 0.857, Std: 0.018
DeepDTA v2	Cold Protein	Mean: **0.795** Min: 0.644, Max: 0.921 Std: 0.102	Mean: **0.788** Min: 0.757, Max: 0.818, Std: 0.023
DeepDTA v2 (MAML++, pretrained)	Cold Protein	Mean: 0.805 Min: 0.656, Max: 0.974, Std: 0.139	Mean: 0.777 Min: 0.735, Max: 0.806, Std: 0.030
DeepDTA v2	Cold Ligand	Mean: 1.360 Min: 0.838, Max: 2.414, Std: 0.609	Mean: 0.696 Min: 0.550, Max: 0.773, Std: 0.087
DeepDTA v2 (MAML++, pretrained)	Cold Ligand	Mean: **1.161** Min: 0.891, Max: 1.458, Std: 0.216	Mean: **0.700** Min: 0.587, Max: 0.785, Std: 0.074
DeepDTA v2	Cold Both	Mean: 1.433 Min: 0.955, Max: 2.610, Std: 0.672	Mean: 0.622 Min: 0.476, Max: 0.753, Std: 0.099
DeepDTA v2 (MAML++, pretrained)	Cold Both	Mean: **1.300** Min: 0.955, Max: 1.664, Std: 0.271	Mean: **0.628** Min: 0.511, Max: 0.732, Std: 0.082
DeepDTA v2	Cold Protein Family	Mean: 1.409 Min: 0.725, Max: 1.895, Std: 0.462	Mean: 0.679 Min: 0.538, Max: 0.781, Std: 0.089
DeepDTA v2 (MAML++, pretrained)	Cold Protein Family	Mean: **1.342** Min: 0.441, Max: 2.504, Std: 0.931	Mean: **0.689** Min: 0.551, Max: 0.788, Std: 0.098

Table 1: MSE and CI statistics for DeepDTA v2 and DeepDTA v2 (MAML++)
(In each cell, “Mean”, “Min”, “Max” and “Std” values are measured across the 5 cross-validation splits)

In table 2, we provide the results from the Chemboost[9] paper. Again, each cell denotes the average MSE or CI value for all 5 cross-validation splits. As mentioned, we used the same splits as in Chemboost in our work, therefore results can be compared.

Model	Validation Set	MSE	CI
DeepDTA v1	Warm	0.345	0. 672*
Chemboost	Warm	0.361	0.880
DeepDTA v1	Cold Protein	0.810*	0.778*
Chemboost	Cold Protein	0.720	0.799
DeepDTA v1	Cold Ligand	1.350*	0.672*
Chemboost	Cold Ligand	1.157	0.730
DeepDTA v1	Cold Both	1.522*	0.614*
Chemboost	Cold Both	1.358*	0.665

Table 2: The averages of MSE and CI for DeepDTA v1 and Chemboost. (The values are taken from the Chemboost paper[9])

We put a star (*) next to the metrics where both or one of our models is better. As we can see, our models performed well than DeepDTA v1 in all cold sets and performed similarly to Chemboost in general. It is important to note that in the cold both test set, which is the most difficult set, DeepDTA v2 (MAML++) is the best model, which shows the advantage of meta-learning for the generalizability. Lastly, we can say adding batch normalization and dropout layers to DeepDTA v1 and applying a meta-learning procedure, that is MAML++ in our paper, improved the cold test set performance of the model in general. MAML++ method can also be applied to Chemboost in future works for better results.

5 Conclusions and recommendations

In conclusion, we have shown that meta-learning techniques can be used to predict drug-target binding affinity. Because of the lack of high-quality and numerous data in this field, data-efficient algorithms such as MAML should be analyzed and examined in future works. Since the success of the well-known deep learning models actually comes from the quality and amount of the dataset[38], we should not approach the problem of drug-target binding affinity as a regular deep learning problem, in our opinion.

In addition, we think that using the protein family information in this problem is a notable improvement since it is important information that gives away much about the protein behavior.

Finally, during our research, we observed that the representation of the molecules is a very important factor in this problem. Although we used the same representation as in the paper[4], we believe that the representation should be improved and the 3D structure should also be given the model somehow. In future work, we are considering to benefit from the recent AlphaFold[39] paper that successfully solves the protein folding problem using a more complex representation for the proteins.

6 Acknowledgments

I want to thank my project advisor Arzucan Özgür for her precious support for this work. Her knowledge and expertise in bio-informatics helped me a lot to continue this project. Besides, I want to thank Rıza Özçelik who encouraged me to make a project about bio-informatics and supported me during the course of this project. I also appreciate Elif Özkırımlı for her contributions, especially about chemical topics which I am not very familiar with. Last but not least, I thank Berk Atıl for his all contributions. It was a pleasure for me that we worked together on this project. We also want to continue to work on this work next semester as his term project.

7 References

[1] Pandey: Drug Discovery and Development Process [Internet]. Connecticut (CT): NorthEast BioLab; [updated 2020 Jun; cited 2021 Feb 10]. Available from: https://www.nebiolab.com/drug-discovery-and-development-process/

[2] Janssen: Addressing The Challenges Of Drug Discovery [Internet]. [place unknown]: Janssen; [[date unknown]; cited 2021 Feb 10]. Available from: https://www.janssen.com/emea/drug-discovery

[3] Shin B, Park S, Kang K, Ho JC. Self-attention based molecule representation for predicting drug-target interaction Doshi-Velez F, Fackler J, Jung K, Kale D, Ranganath R, Wallace B, Wiens J, editors. arXiv [cs.LG]. 2019:230–248.

[4] Öztürk H, Özgür A, Ozkirimli E. DeepDTA: deep drug-target binding affinity prediction. Bioinformatics (Oxford, England). 2018;34(17):i821–i829.

[5] Nguyen T, Le H, Quinn TP, Nguyen T, Le TD, Venkatesh S. GraphDTA: Predicting drug-target binding affinity with graph neural networks. Bioinformatics (Oxford, England). 2020 [accessed 2021 Feb 11]. doi:10.1093/bioinformatics/btaa921

[6] Öztürk H, Ozkirimli E, Özgür A. WideDTA: prediction of drug-target binding affinity. arXiv [q-bio.QM]. 2019. http://arxiv.org/abs/1902.04166

[7] Zhao Q, Xiao F, Yang M, Li Y, Wang J. AttentionDTA: prediction of drug–target binding affinity using attention model. In: 2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM). IEEE; 2019. p. 64–69.

[8] Jiang M, Li Z, Zhang S, Wang S, Wang X, Yuan Q, Wei Z. Drug–target affinity prediction using graph neural network and contact maps. RSC advances. 2020 [accessed 2021 Feb 11];10(35):20701–20712.

[9] Özçelik R, Öztürk H, Özgür A, Ozkirimli E. ChemBoost: A chemical language based approach for protein - ligand binding affinity prediction. Molecular informatics. 2020;(minf.202000212). http://dx.doi.org/10.1002/minf.202000212. doi:10.1002/minf.202000212

[10] Imrie F, Bradley AR, van der Schaar M, Deane CM. Protein family-specific models using deep neural networks and transfer learning improve virtual screening and highlight the need for more data. Journal of chemical information and modeling. 2018;58(11):2319–2330.

[11] Ross GA, Morris GM, Biggin PC. One size does not fit all: The limits of structure-based models in drug discovery. Journal of chemical theory and computation. 2013;9(9):4266–4274.

[12] Zelenko D, Aone C, Richardella A. Kernel methods for relation extraction. In: Proceedings of the ACL-02 conference on Empirical methods in natural language processing - EMNLP '02. Morristown, NJ, USA: Association for Computational Linguistics; 2002.

[13] Mi F, Huang M, Zhang J, Faltings B. Meta-learning for low-resource natural language generation in task-oriented dialogue systems. arXiv [cs.CL]. 2019. http://arxiv.org/abs/1905.05644

[14] Bleakley K, Yamanishi Y. Supervised prediction of drug-target interactions using bipartite local models. Bioinformatics (Oxford, England). 2009;25(18):2397–2403.

[15] Cao D-S, Zhang L-X, Tan G-S, Xiang Z, Zeng W-B, Xu Q-S, Chen AF. Computational prediction of Drug□Target interactions using chemical, biological, and network features. Molecular informatics. 2014;33(10):669–681.

[16] Öztürk H, Ozkirimli E, Özgür A. A comparative study of SMILES-based compound similarity functions for drug-target interaction prediction. BMC bioinformatics. 2016;17(1):128.
[17] Gilson MK, Liu T, Baitaluk M, Nicola G, Hwang L, Chong J. BindingDB in 2015: A public database for medicinal chemistry, computational chemistry and systems pharmacology. Nucleic acids research. 2016;44(D1):D1045-53. doi:10.1093/nar/gkv1072
[18] Sterling T, Irwin JJ. ZINC 15--ligand discovery for everyone. Journal of chemical information and modeling. 2015;55(11):2324–2337.
[19] Davis MI, Hunt JP, Herrgard S, Ciceri P, Wodicka LM, Pallares G, Hocker M, Treiber DK, Zarrinkar PP. Comprehensive analysis of kinase inhibitor selectivity. Nature biotechnology. 2011;29(11):1046–1051.
[20] Metz JT, Johnson EF, Soni NB, Merta PJ, Kifle L, Hajduk PJ. Navigating the kinome. Nature chemical biology. 2011;7(4):200–202.
[21] Sciencesnail, The difference between Ki, Kd, IC50, and EC50 values [Internet]. [location unknown]: Sciencesnail.com; [[date unknown]; accessed 2021 Feb 10]. Available from: https://www.sciencesnail.com/science/the-difference-between-ki-kd-ic50-and-ec50-values
[22] He T, Heidemeyer M, Ban F, Cherkasov A, Ester M. SimBoost: a read-across approach for predicting drug-target binding affinities using gradient boosting machines. Journal of cheminformatics. 2017;9(1):24.
[23] Weininger D. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. Journal of chemical information and modeling. 1988;28(1):31–36.
[24] Mendez D, Gaulton A, Bento AP, Chambers J, De Veij M, Félix E, Magariños MP, Mosquera JF, Mutowo P, Nowotka M, et al. ChEMBL: towards direct deposition of bioassay data. Nucleic acids research. 2019 [accessed 2021 Feb 10];47(D1):D930–D940.
[25] Wikipedia contributors. Simplified molecular-input line-entry system. Wikipedia, The Free Encyclopedia. 2021 Feb 8 [accessed 2021 Feb 10]. https://en.wikipedia.org/w/index.php?title=Simplified_molecular-input_line-entry_system&oldid=100552252
[26] Lipman DJ, Pearson WR. Rapid and sensitive protein similarity searches. Science (New York, N.Y.). 1985 [accessed 2021 Feb 10];227(4693):1435–1441.
[27] Uniprot, Insulin [Internet]. [location unknown]: Uniprot.org. [[date unknown]; accessed 2021 Feb 10]. Available from: https://www.uniprot.org/uniprot/A6XGL2
[28] Ioffe S, Szegedy C. Batch Normalization: Accelerating deep network training by reducing internal covariate shift Bach F, Blei D, editors. arXiv [cs.LG]. 2015:448–456. http://proceedings.mlr.press/v37/ioffe15.html
[29] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: A simple way to prevent neural networks from overfitting. Journal of machine learning research: JMLR. 2014 [accessed 2021 Feb 10];15(56):1929–1958.
[30] Xu B, Wang N, Chen T, Li M. Empirical evaluation of rectified activations in convolutional network. arXiv [cs.LG]. 2015 [accessed 2021 Feb 10]. http://arxiv.org/abs/1505.00853
[31] Finn C, Abbeel P, Levine S. Model-agnostic meta-learning for fast adaptation of deep networks. arXiv [cs.LG]. 2017. http://arxiv.org/abs/1703.03400
[32] Antoniou A, Edwards H, Storkey A. How to train your MAML. arXiv [cs.LG]. 2018. http://arxiv.org/abs/1810.09502

[33] Dura B. Meta-learning for drug discovery. 2020. https://bdura.me/assets/files/Meta-learning%20for%20drug%20discovery.pdf
[34] Gönen M, Heller G. Concordance probability and discriminatory power in proportional hazards regression. Biometrika. 2005;92(4):965–970.
[35] Antoniou A. HowToTrainYourMAMLPytorch. github.com. [accessed 2021 Feb 10]. https://github.com/AntreasAntoniou/HowToTrainYourMAMLPytorch
[36] PyTorch. Pytorch.org. [accessed 2021 Feb 10]. http://pytorch.org/
[37] Kingma DP, Ba J. Adam: A method for stochastic optimization. arXiv [cs.LG]. 2014. http://arxiv.org/abs/1412.6980
[38] Deng J, Dong W, Socher R, Li L-J, Li K, Fei-Fei L. ImageNet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition. IEEE; 2009. p. 248–255.
[39] Senior AW, Evans R, Jumper J, Kirkpatrick J, Sifre L, Green T, Qin C, Žídek A, Nelson AWR, Bridgland A, et al. Improved protein structure prediction using potentials from deep learning. Nature. 2020;577(7792):706–710.

Appendix A: Dataset Statistics

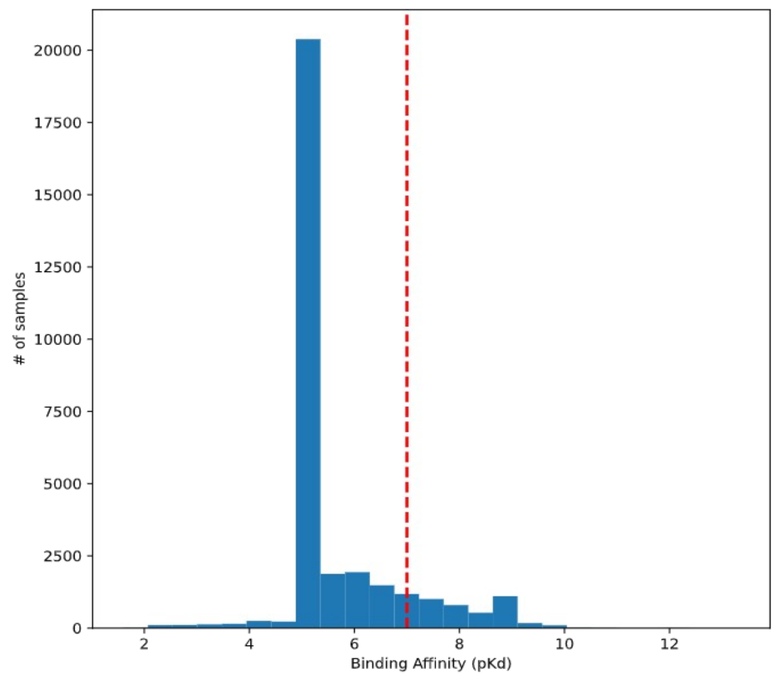


Illustration 7: Histogram of binding affinity values in our dataset.

The red line represents the threshold to classify weak and strong bindings. It was proposed by the paper22. From this plot, it is clear that the dataset is very biased towards weak bindings.

	Protein	Ligand	Protein Family
# of unique ...	490	924	86
Avg. # of interaction data	34.2	64.5	367.5
Min # of interaction data	3	6	6
Max # of interaction data	6	162	12746
Std of the # of samples	100.3	26.1	1377.2

Table 3: Some important statistics of the extracted dataset

It is obvious that the dataset is imbalanced. For example, there are 12746 samples for the Pkinase family (PF00069), while there are only 6 samples for the Chromo family (PF00385). As mentioned, this imbalance causes the models to not be able to generalize well. As we have shown in this work, using meta-learning solves this problem to some extend.

Appendix B: DeepDTA v1 and DeepDTA v2 Architectures

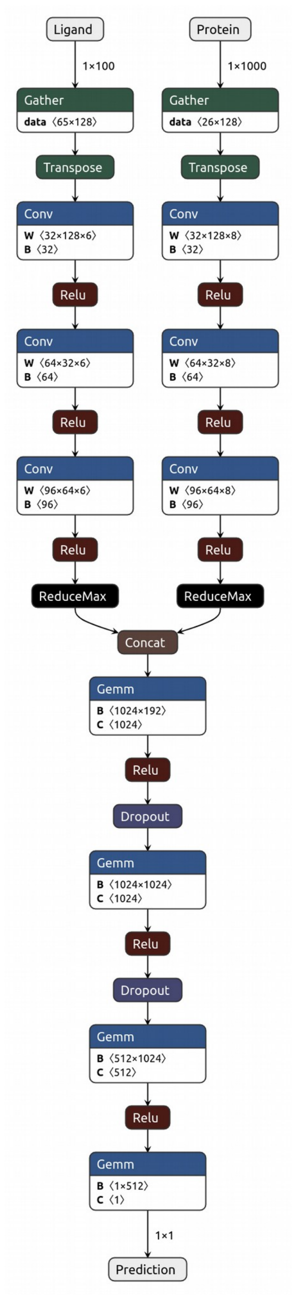


Illustration 8: DeepDTA v1 Architecture (Taken from the DeepDTA paper[4])

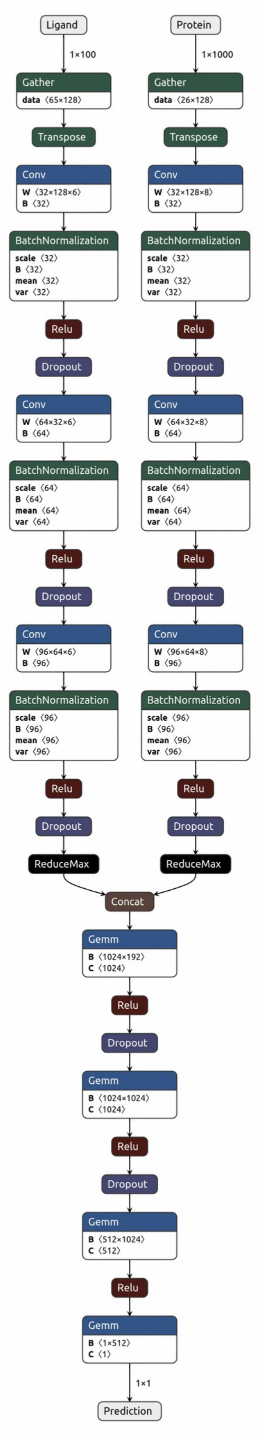


Illustration 9: DeepDTA v2 Architecture