In-silico prediction of protein-ligand binding affinity using deep learning

Undergraduate Thesis

Submitted in partial fulfillment of the requirements of
BITS F422T Thesis

By

Amitesh Badkul
ID No. 2018B2A30764H

Under the supervision of:

Dr. Lei Xie
&
Dr. Sudha Radhika

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI, HYDERABAD CAMPUS,
May 2023

"In the beginning, there was simplicity."

Richard Dawkins, *The Selfish Gene*

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI, HYDERABAD CAMPUS,

Abstract

Bachelor of Engineering (Hons.) in Electrical and Electronics Engineering

In-silico prediction of protein-ligand binding affinity using deep learning

by Amitesh Badkul

In recent years, the field of drug discovery has encountered significant challenges that call for the exploration of efficient computational techniques. Motivated by the potential of deep learning to address these challenges, this research seeks to leverage its capabilities for predicting binding affinities—a crucial element in drug design.

To achieve this goal, our study delves into both single-task and multi-task learning approaches. Multi-task learning stands out as a promising method, offering advantages in capturing shared representations across tasks. Detailed experiments reveal the impact of factors such as batch size, learning rate, and dropout rates on model performance. Notably, employing larger batch sizes and lower learning rates leads to improved model outcomes. Additionally, strategically freezing layers during regression tasks and adjusting the number of classes enhance model generalization.

Further investigations compare various deep learning architectures for protein sequence embeddings. The ESM-2 model, trained on extensive datasets for direct processing of protein sequences, demonstrates superiority over the DISAE-based model. Concerning feature extraction from protein sequences, the ResNet model, with its adept convolutional layers for capturing localized protein features, outperforms other architectures like LSTM, BiLSTM, and Transformer-based models.

In essence, this research highlights the potential of deep learning in transforming binding affinity predictions. The insights gained have the potential to streamline future drug discovery efforts.

Acknowledgements

I would like to express my sincere gratitude to my primary supervisor, Dr. Lei Xie, Professor in the Department of Computer Science at Hunter College, CUNY. His invaluable guidance and expertise in the fields of biology and deep learning have been instrumental in achieving the results presented in this work. I am truly grateful for his unwavering support and mentorship throughout the duration of the project.

I would also like to extend my appreciation to Dr. Sudha Radhika, Assistant Professor in the Department of Electrical and Electronics Engineering at BITS Pilani Hyderabad, for her continuous support and for introducing me to the wonderful world of Deep Learning.

Furthermore, I am thankful to Tian Cai, a Ph.D. candidate in the Department of Computer Science at Hunter College, for providing me with valuable resources and engaging discussions on deep learning models in bioinformatics, which have been immensely beneficial to my research.

I am also indebted to my lab mates, Shuo Zhang and Yoyo Wu, for their assistance in various aspects of the project, which has greatly enriched its outcome.

I would like to express my heartfelt gratitude to my family, my parents Dr. Alok Badkul and Dr. Madhulika Jain, and my sisters Avani and Anika, and my roommates Kshitij and Mayuresh, for their unwavering support and encouragement throughout this journey. Additionally, I am grateful to my friends, including Disha, Akshat, Aadarsh, Aditi, Anant, Ameya, Anjur, Ishita, Subhransu, Rishav, Pravar, and Nitya, for their support and thought-provoking discussions.

Their collective contributions have significantly enhanced my research experience, and I am truly appreciative of their support and meaningful interactions.

Contents

List of Figures

List of Tables

Chapter 1

Biological Relevance of Proteins

1.1 Proteins

Proteins are essential to the structure and function of cells in living organisms. The powerhouses of the cell proteins are crucial for many biological processes. They execute a wide array of functions in the cell, including catalysis of chemical reactions, the transmission of signals between cells, providing structural support to the cell, and serving as transporters for molecules across cell membranes. They are macromolecules formed by peptide bonds that connect the numerous amino acids that make up each one of them. Proteins can be categorized based on their structure, function, and sequencing. A protein's secondary, tertiary, and quaternary structures refer to the three-dimensional folding of the protein, while the primary structure refers to the sequential alignment of amino acids. A protein's secondary structure is defined as the local folding of the amino acid chain, which typically results in alpha-helices or beta-sheets. The three-dimensional folding of the protein, controlled by the interactions between various amino acid chain segments, is called the tertiary structure. The organization of many protein subunits into a functional complex is known as the quaternary structure [7].

Most drugs act by targeting specific proteins inside the cell in the body, either by inhibiting their activity or by enhancing it. For example, antibiotics such as penicillin target bacterial enzymes involved in cell wall synthesis. In contrast, anticancer drugs such as taxanes inhibit the function of microtubules involved in cell division from stopping the unending and rapid division and growth of cancer cells. Understanding the structure and function of proteins is vital in drug discovery, as it allows researchers to identify potential drug targets and design drugs that can selectively bind to them. Protein structure determination techniques, including X-ray crystallography, NMR spectroscopy, and cryo-electron microscopy, have enabled detailed structural 3D information about proteins [7]. Computational methods such as molecular modeling [16] and virtual screening [29, 12] can also be used to predict the binding of small molecules to proteins, which can help

identify potential drug candidates. Molecular modeling employs computational procedures to create three-dimensional models of protein structures and predict the interactions between small drug-like molecules and proteins. This allows researchers to visualize how a potential drug may bind to a target protein and assess its effectiveness in the treatment. Virtual screening is a computational process encompassing a vast database of small compounds likely to bind with the protein target being investigated. For example, AutoDock Vina [28] is a widely used software due to its speed and accuracy. It is used in drug discovery and virtual screening studies for molecular docking, a computational method to predict the binding affinity of proteins and chemicals. The software utilizes a scoring function that considers the energy of interaction between the protein and chemical and their shapes.

1.2 Drug Discovery

Drug Discovery refers to the identification of potential new drugs which integrate experimental, translational, clinical models, and computational. It is a multi-step process that involves identifying and developing new drugs. The pipeline for drug discovery is a convoluted and time-consuming task that involves -

1. **Discovery and Development** - It involves the identification of potential drug targets, which are either molecules or proteins involved in an illness. Target identification is achieved through various methods, including but not limited to genetic and proteomic investigations, virtual screening of large chemical libraries, and studying disease mechanisms and pathways. The next step involves identifying compounds that can interact with the target. It is referred to as **hit identification**. It consists of screening large libraries of compounds to identify those that show interaction activity against the target. This is typically done using high-throughput screening methods after selecting the potential target compounds from the hit identification process. The next stage consists of lead optimization. This encompasses modifying the hit compounds' pre-existing structure to enhance their activity, selectivity, and safety. This process is referred to as **lead optimization**.

2. **Pre-clinical Research** - After the lead optimization process, the selected compounds undergo preclinical development, which involves extensive and comprehensive animal testing to evaluate their **safety**, **toxicity**, **efficacy**, and **pharmacokinetics**.

3. **Clinical Research** - The final stage of drug discovery is clinical trials, which involve testing the drug candidate in **human subjects** to evaluate its safety and effectiveness. **Clinical trials** are conducted in three phases, each involving more subjects and additional **rigorous testing**.

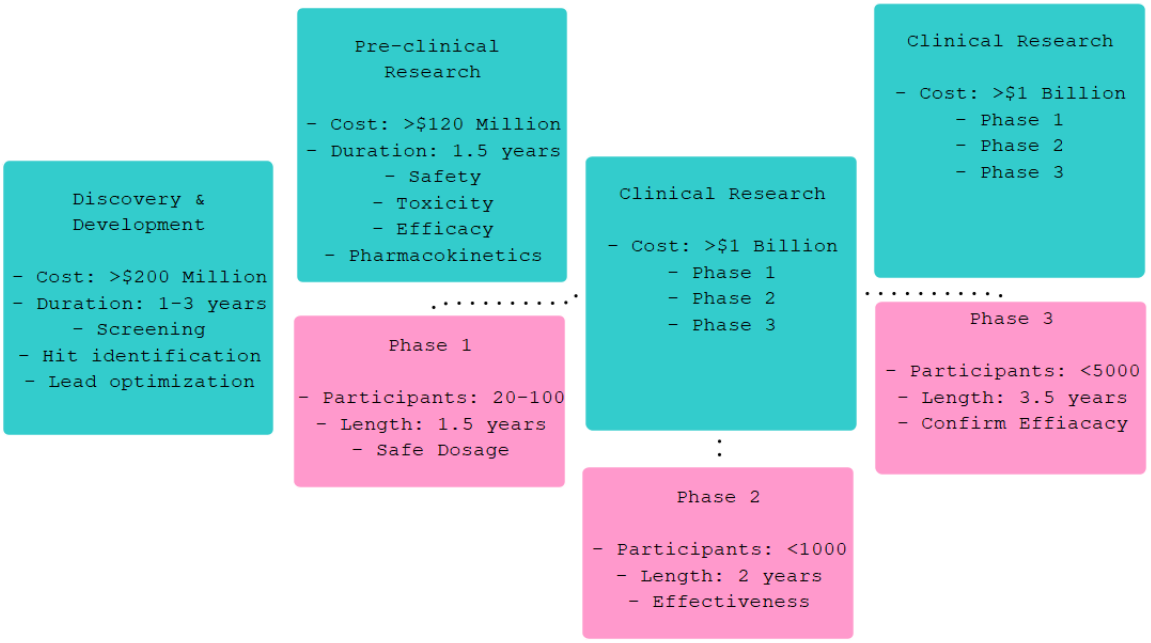

FIGURE 1.1: General Drug Discovery Process

(a) **Phase 1**: This phase involves a small number of healthy volunteers who are given the drug candidate to test its safety and toxicity. The goal of this phase is to determine the **safe dosage** range and identify any potential side effects.

(b) **Phase 2**: This phase involves a larger group of patients with the disease or condition the drug is intended to treat. The goal of this phase is to evaluate the **drug's efficacy** in treating the disease, as well as to continue to assess its safety. Patients are typically randomized into different groups, with each group receiving either the drug candidate or a placebo.

(c) **Phase 3**: This phase typically involves an even larger group of patients and is the last stage of the clinical trials. It is meant to validate the **effectiveness** and **safety** of the proposed drug concerning the human body. Patients are typically randomized into different groups, each receiving either the drug candidate or the standard of care treatment.

4. **Food and Drug Administration (FDA) Approval** - Upon successfully passing the proposed drug through the various stages of preclinical and clinical trials, the U.S. FDA exhaustively examines and investigates the proposed drug before releasing it into the market. The rigorous FDA review process involves multiple **evaluation stages**, including **preclinical** and **clinical data submission**, **manufacturing facilities**, and **operations evaluation**.

Traditional Drug Discovery	DL-Based Drug Discovery
A long and expensive process that takes 10-15 years and costs billions of dollars.	A quicker and less costly process, taking 4-5 years and costing only millions of dollars.
It depends on a trial and error approach, screening thousands of compounds to narrow it down to one compound. It is a target driven process.	Deep Learning (DL) algorithms utilize vast amounts of data to determine patterns and predict potential drug candidates. It is a data driven process.
Restricted by the human ability to investigate and analyze complex data.	DL algorithms can analyze large amounts of data more quickly and accurately than humans.
It has a low success rate, with only a slim percentage of drugs making it through clinical trials.	DL-based drug discovery can increase the success rate by identifying promising drug candidates earlier in the process.
Incredibly restrained due to the complicated nature of cellular interactions and cellular pathways.	Identify chemicals that could bind to 'undruggable targets,' that is, proteins whose structures aren't defined.

TABLE 1.1: Traditional Drug Discovery vs DL-Based Drug Discovery

1.3 Interaction of Chemical and Proteins: Quantification

The interaction between proteins and chemicals is crucial in many biological processes [15, 8], including metabolism, cellular signaling, and gene expression. Quantifying this interaction and understanding its mechanisms is vital to the drug discovery process.

1.3.1 Quantification

The measurement of the interaction between proteins and chemicals [13] is achieved through various methods, such as:

1. **Binding affinity (K_d)**: Binding affinity refers to the strength of interaction between two compounds, molecules, in our case between the protein and chemical, and it is expressed as the dissociation constant (K_d), which is the concentration of the chemical required to dissociate 50% of the protein-ligand complex. The lower the K_d value, the stronger the binding and interaction between the protein-ligand complex.

2. **Gibbs free energy change (ΔG)**: The Gibbs free energy change [2] represents the thermodynamic stability of the protein-chemical complex system. Negative ΔG values indicate favorable binding.

3. **Half Maximal Inhibitory Concentration (IC_{50})**: IC_{50} is a measure of the effectiveness of a chemical or drug in blocking a specific target's activity. The IC_{50} measurement indicates the drug concentration required to inhibit 50% of the target's activity, typically expressed in molarity units.

1.3.2 Calculation of Binding Affinity, Gibbs Free Energy Change and Half Maximal Inhibitory Concentration

Let protein 'P' and ligand 'L' interact with each other and 'PL' be the protein-ligand complex obtained after interaction.

$$\mathrm{P} + \mathrm{L} \underset{k_{off}}{\overset{\mathrm{k_{on}}}{\rightleftharpoons}} \mathrm{PL} \tag{1.1}$$

In a state of equilibrium, the reaction is written as follows, where [P], [L], and [PL] is the concentration of protein, ligand and protein-ligand complex at equilibrium:

$$\mathrm{k_{on}[P][L] = k_{off}[PL]} \tag{1.2}$$

Rearranging the equation as shown in Equation 1.3, where K_b is the binding constant and K_d is the dissociation constant:

$$\mathrm{K_b} = \frac{\mathrm{k_{on}}}{\mathrm{k_{off}}} = \frac{\mathrm{[PL]}}{\mathrm{[P][L]}} = \frac{1}{\mathrm{K_d}} \tag{1.3}$$

IC_{50} and K_i (Inhibition) are two terms that are defined within the same context. In the case of competitive binding, the Michaelis-Menten [17] kinetics and K_i can be employed to estimate the value of IC_{50}.

$$IC_{50} = K_i(1 + \frac{[S]}{K_m}) \tag{1.4}$$

The binding free energy (ΔG) of the protein chemical complex system can be determined at any given moment during an association, regardless of whether it is at standard-state conditions, using the following equations, where R is the universal gas constant ($1.987\ cal \cdot K^{-1} \cdot mol^{-1}$), T is the temperature in Kelvin and Q is the reaction quotient:

$$\Delta\mathrm{G} = \Delta\mathrm{G}^\circ + \mathrm{RTln(Q)} \tag{1.5}$$

1.4 Outline

The motivation for exploring deep learning models for drug discovery is now clear. The remainder of the thesis follows - understanding the deep learning framework for binding affinity prediction.

It describes approaches like single-task learning and multi-task learning-based predictions for binding affinity calculation.

Chapter 2 describes multi-task learning and explains the motivation, advantages, and significance of multi-task learning in a general and drug discovery-based setting.

Chapter 3 presents an extensive and thorough explanation of the implementation of the binding affinity prediction task as a multi-task learning problem. It further discusses the results obtained and lists out the further work.

Chapter 4 highlights the usage of the Evolutionary Scale Modeling (ESM) model [21]and the different architecture used along with it for chemical protein binding affinity prediction. Along with this it introduces the deep learning-based regression framework.

Chapter 2

Multi-Task Learning

2.1 Introduction

Multi-task learning [27] is a machine learning procedure that involves training a model to perform multiple tasks simultaneously instead of training separate models for each task to improve the model and obtain better generalization on new unseen data. The idea behind this approach is that the model can better generalize to new and unseen data by learning numerous related tasks together. Generally, any model that involves optimizing multiple loss functions implies using multi-task learning instead of the traditional single-task approach. There are two categories of multi-task learning: hard parameter sharing and soft parameter sharing, they are described in the section below.

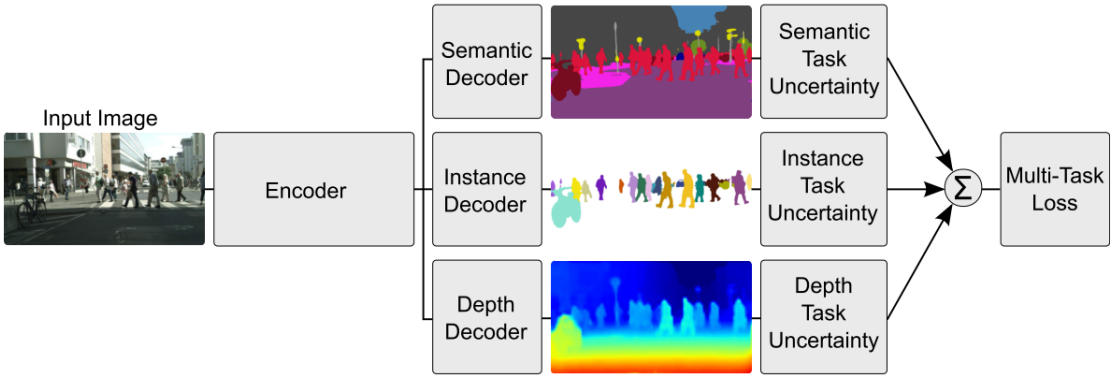

FIGURE 2.1: Example of Mutli-Task Learning (with Uncertainty Weights implementation) (from cite this paper)

2.2 Hard & Soft Parameter Sharing

The model shares the same parameters across all tasks in **hard parameter sharing**. This approach is practical when the tasks are highly correlated. In contrast, each task has its own set of parameters in **soft parameter sharing**, but the parameters are regularized, such as L2 and trace norms, to be similar. This approach is helpful when the tasks and the data have a fair amount of similarity and some dissimilarities in their input data. For example, the **multi-task question answering network** (MQAN) [24], a natural language processing model that is trained on ten similar tasks - question answering, summarization, machine translation, inference, semantic labeling, semantic analysis, parsing, and pronoun resolution is a hard parameter sharing model due to the closely related nature of the tasks. On the other hand, a soft parameter-sharing network is utilized in tasks like **cross-lingual model** training where the data isn't entirely related to each other [14].

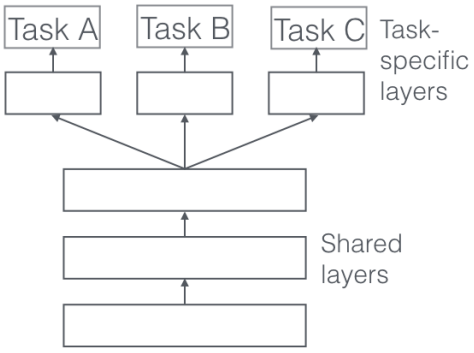

FIGURE 2.2: Hard-Parameter Sharing (from (https://www.ruder.io/multi-task))

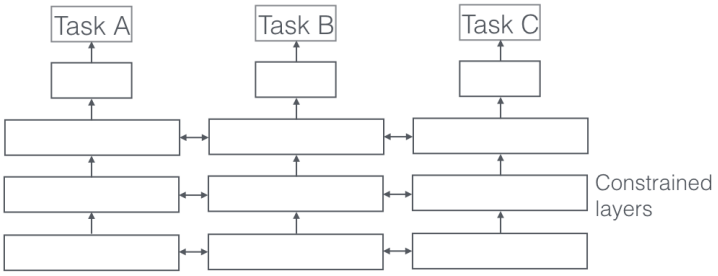

FIGURE 2.3: Soft-Parameter Sharing (from (https://www.ruder.io/multi-task))

2.3 Why does it work?

Multi-task learning works [27] because of the following reasons:

1. **Implicit Data Augmentation**: Implicit data augmentation is a process to increase the effective batch size for training deep learning and machine learning models. Multi-task learning achieves implicit data augmentation, which introduces noise in the learning process due to numerous tasks and their respective data. Most single-task models are highly prone to overfitting, which can be avoided by using multi-task learning. This leads to better generalizability by averaging the noise patterns across all the tasks and reducing the risk of overfitting.

2. **Adequate Representation**: The presence of multiple tasks constrains the model to learn suitable and superior representations, and hence it implicitly understands to filter out irrelevant information and features.

3. **Eavsdropping**: In some cases, certain features may be challenging to learn for the model for a specific task but straightforward to learn for another. MTL can enable "eavesdropping," which lets the model learn the features through numerous tasks.

4. **Reducing Bias**: Furthermore, MTL can help reduce representation bias by biasing the model towards representations preferred by other tasks. This can improve the model's ability to generalize to new tasks and recent unseen data, provided the tasks are from the same environment since the model is trained on multiple tasks, which inherently decreases the Rademacher complexity.

2.4 Task Balancing Methods

The amount of importance given to the various tasks has a significant impact on the generalizing power of the final trained model [31]. Therefore, it is important to consider the various techniques through which the impact of each task is taken in consideration. Task balancing methods deal with the problem of gradients and imbalanced datasets for the various tasks for multi-task learning. The losses and gradients of the multiple tasks significantly impact the final model. There are two main categories of task-balancing methods:

1. **Loss Balancing Methods**: Loss balancing methods seek to adequately measure the contribution of every task to the total loss function by allocating weights to them.

   (a) **Uncertainity Weights (UW) [18]**: The uncertainty weights method uses uncertainty to weigh the loss connected with each task. The reasoning is that the deep learning model should focus more on the task with higher uncertainty, which means the task is more difficult to model. The method calculates the uncertainty linked with each task along with its model parameters, quantifying uncertainty. The losses of the respective tasks are then weighted based on the uncertainties obtained.

   (b) **Dynamic weight average (DWA) [22]: The method of uncertainty weight averaging uses the attention mechanism. The weights for the losses of each task are decided based on the relevance and significance of the tasks as calculated by the attention mechanism. The losses have different scales, and some tasks overpower the training process, leading to an inferior performance on the remaining tasks.**

2. **Gradient Balancing Methods**: Gradient balancing methods involve adjusting the gradients of every task for the entire model.

   (a) **Gradient Normalization**: The authors propose, GradNorm [10], a method that addresses this problem by normalizing the gradients of the model. The normalization is based on the Lasso L2 norm. After normalization, the normalized gradients are utilized to update the layers' weights in the model. Due to gradient imbalance from the training process, the model often fails to generalize and learn representations properly.

   (b) **Projecting conflicting gradient**: The introduced method of Projecting conflicting gradients (PCGrad) [30] tackles this issue by selectively modifying the gradients of the model during training. The technique involves updating the gradients in direct relation to the tasks.

Therefore, task-balancing methods are vital for improving the performance and stability of multi-task learning models by attenuating the effect of gradients and losses of each task.

Chapter 3

Deep Learning-based classification for Prediction of Chemical Protein Interaction

3.1 Introduction

In this Chapter, we propose and investigate two main methods as part of predicting protein-chemical interaction based on -

1. Single-Task Learning (STL)
2. Multi-Task Learning (MTL)

Both of these models (STL & MTL) include the classification of chemical protein interaction and the binding affinity prediction. We examine protein-chemical interaction - qualitatively, which involves predicting the given protein and chemical pair interact or not, and quantitatively, which consists in predicting the strength of a protein and chemical interaction based on their inhibitory constant (K_i), which is explained in Chapter 1. However, both tasks are trained on the same data with different settings. The Chapter further discusses the data, the dataset in various classification and regression settings, it's cleaning, and preprocessing. Then we introduce our novel architecture for both tasks and discuss the results obtained.

3.2 Dataset

The dataset for the task of chemical-protein interaction is obtained from European Molecular Biology Laboratory (EMBL), referred to as the ChEMBL [25] dataset. In our case, it is the

Chemical	Protein	K_i
$CC(C)C[C@H](NC(=O)OCC1c2ccccc2-c2ccccO$	$Q9NRA2$	10200.00

TABLE 3.1: An example of the dataset

ChEMBL 32 dataset. The ChEMBL dataset provides the binding affinity of the chemical and protein pair in the form of the K_i value. The chemicals in the dataset are written in Simplified molecular-input line-entry system (SMILES) format, and the proteins are represented using their respective UniProt ID. UniProt [11] is a protein sequence dataset that provides a unique identifier, referred to as the UniProt ID, for each protein in the dataset. The UniProt ID serves as a primary accession number for any protein in the UniProt database. Uniprot IDs are a widely used identification method for referring to proteins. K_i is measured in micro-molar (μM). The data consists of $413,105$ protein chemical pairs and their respective interaction. The table 3.1 depicts the original dataset.

Along with the UniProt ID, the protein is identified using Pfam [26] (Protein family), a database of protein families, domains, and functional units that provides information on protein sequence, structure, and function.

3.2.1 Cleaning

The Ki values in the dataset ranged from $3.13 \times 10^{22} \pm 1.68 \times 10^{25}$, and the distribution of the values was too wide for any computational model to easily mathematically model. It is well-known that normal distributions are easier for any computational model, such as a deep learning model in our case model. Therefore, to simplify the task for the deep learning model, we apply log transformation to the dataset and obtain a normal distribution. Fig. 3.1 and 3.2 illustrate the effect of the transformation. To ensure there is no error when applying the log transform, the K_i values equal to zero are removed from the dataset, after which the total number of data points reduces by 1.8% only.

3.2.2 Regression

For the regression task, the newly obtained $log(K_i)$ is the dependent variable predicted using the regression task.

3.2.3 Classification

For the binary classification task, the 100 μM threshold is used to separate the interacting and non-interacting chemical protein pairs. For values greater than the threshold, there isn't any

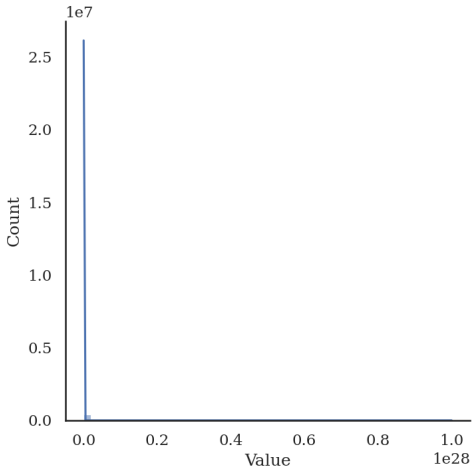

FIGURE 3.1: Original K_i distribution

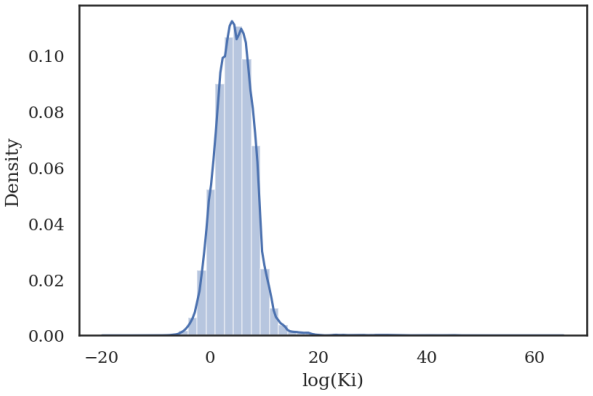

FIGURE 3.2: Distribution after log transform

interaction occurring. 100 μM is used as a threshold to ensure that the positive or interacting and negative or non-interacting classes have equal data points for dataset balance and more accessible learning for the deep learning models. Fig. 3.3 shows the balanced dataset obtained for binary classification after cleaning.

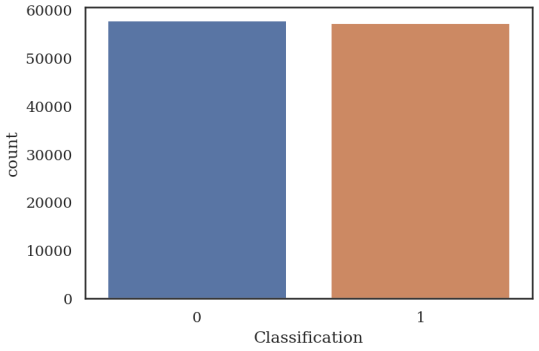

FIGURE 3.3: Balanced Dataset

The dataset was divided into 'n' classes along with binary classification to observe the model's performance when the task changes to multi-class classification. Each of these 'n' classes is balanced with an equal amount of data to make learning for the model more straightforward, as shown in Fig. 3.4.

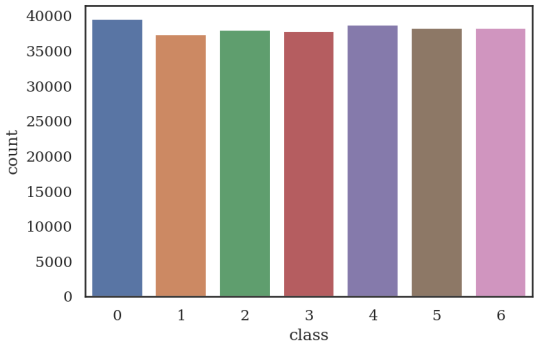

FIGURE 3.4: Equal Distribution for 7 Class division on the dataset

3.3 Models

3.3.1 Model Architecture

Firstly, we propose the binary classification of the chemical-protein interaction using a novel deep learning architecture, which utilizes the state-of-the-art distilled sequence alignment embedding (DISAE) [6] for obtaining the protein representation from its sequence. A Graph Neural Network [32] based model to obtain a chemical representation. And finally, an attention mechanism-based layer is to learn the significant interactions between the chemical and protein pair. Now this deep learning architecture was first modified such that instead of classification, it performs regression of the K_i values, however since neural networks are well known for classification tasks as compared to regression tasks therefore, the performance obtained by the regression model was very inferior considered to the classification task. Hence the idea of a deep learning-based regression architecture was immediately discarded. Since regression-based tasks are easily mathematically modeled using simpler machine learning architectures like random forest regression [3], xgboost regression [9], support vector regression [1], and others. The main idea now was to obtain features for the regression models obtained from the intermediate layer of the classification model itself. The STL section further discusses the architecture and the layer from which the features are extracted.

Fig. 3.5 illustrates the general working of the model, where the features for the regression task are obtained from the interaction pooler layer because it consists of the aggregate information of the protein, chemical, and their interaction. In the case of STL, the loss function is simply the classification loss, whereas, in the case of the MTL model, the loss function incorporates the regressor loss as well. Equation (3.1) is the mathematical representation of the loss function under STL and MTL circumstances. Let classification loss and regression loss be denoted as L_c & L_r, and the total loss be L_{tot}. And the weights associated based on the various weighting strategies described above are w_c & w_r. Therefore the total loss in both cases is represented as follows:

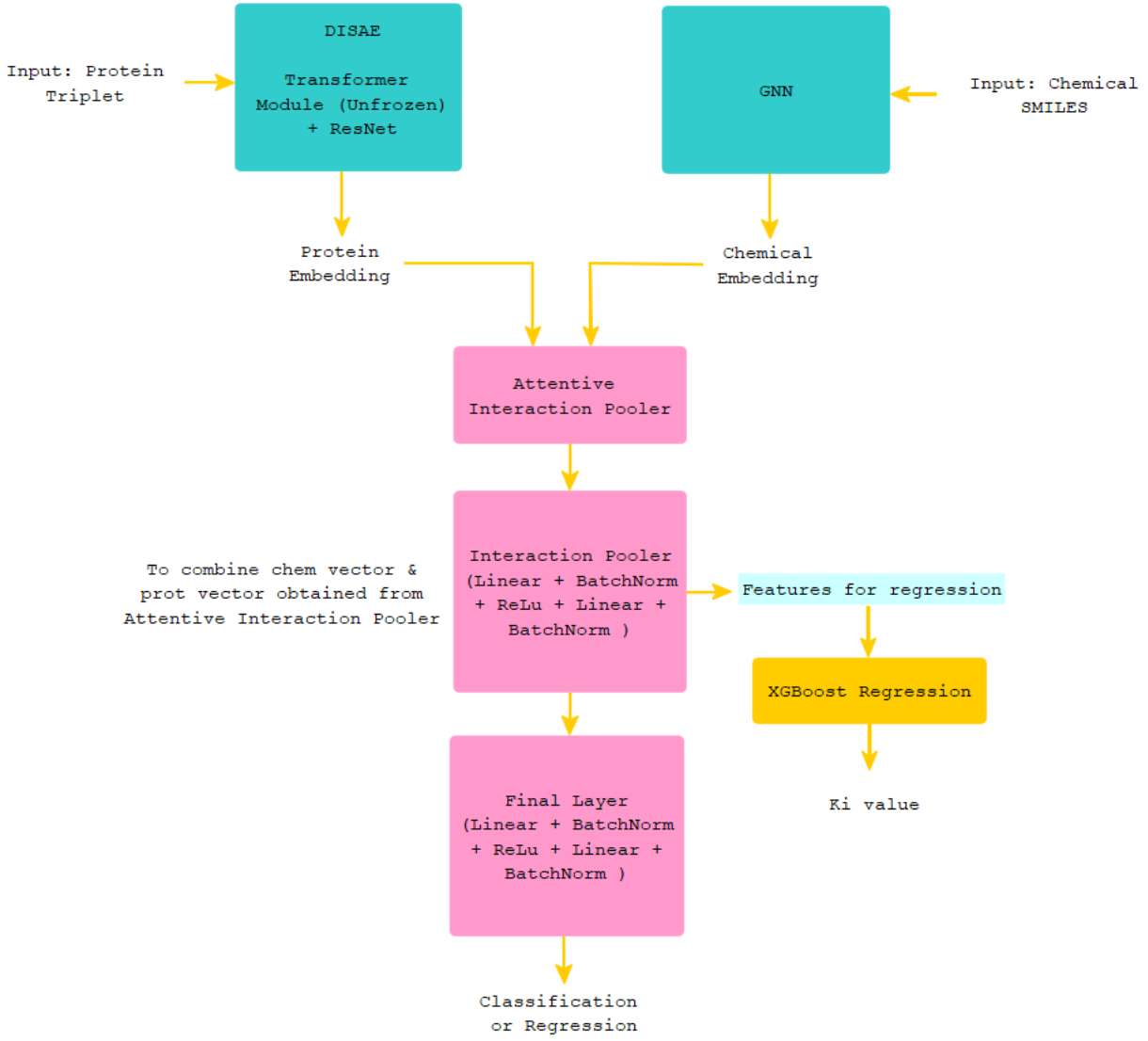

FIGURE 3.5: Model Architecture

$$L_{tot} = \begin{cases} L_c & STL \\ w_c * L_c + w_r * L_r & MTL \end{cases} \tag{3.1}$$

For the MTL, to obtain the weights w_c & w_r, we utilize the following loss balancing methods:

1. Equal Weights
2. Random Weights
3. Uncertainty Weights

Now I will describe the individual components in the model architecture.

3.3.2 DIstilled Sequence Alignment Embedding (DISAE)

DIstilled Sequence Alignment Embedding (DISAE) is a deep learning model that predicts chemical ligand binding to orphan proteins without observable and detectable association with annotated proteins. The figure 3.6 depicts the DISAE model. The DISAE model consists of two phases:

1. Unsupervised Learning: For learning protein representations using only sequence data.
2. Supervised Learning: To predict if a chemical and a protein interact using the known Chemical-Protein Interactions (CPIs) collected from chemical genomics databases.

In the first phase, DISAE uses a self-supervised masked language modeling (MLM) approach for learning the protein representations using the ALBERT (A Light BERT) architecture. The original protein sequence is distilled into an ordered list of triplets by excluding evolutionarily unimportant positions from a Multiple Sequence Alignment (MSA). MLM involves randomly masking 15% of amino acids in a protein sequence, training the model to predict the masked amino acids based on the surrounding context, and using the self-attention in the Transformer module of ALBERT. This process of distillation of the sequence improves the efficiency of sequence pretraining and reduces noise in the input sequence. The MLM component in DISAE is based on the same principle as the MLM component in the ALBERT language model, which was developed for natural language processing tasks. However, DISAE has been adapted to work with protein sequences and uses different masking methods optimized for this data type. By training on a large dataset of protein sequences, the model learns to encode the structural and functional characteristics of different proteins from its sequence.

In the second stage, a supervised learning model is trained to predict CPIs. The model takes the following inputs - the representation of chemical structures from neural fingerprints and the pretrained protein sequence embedding from the first phase. The deep learning model for the CPI prediction comprises three components: protein embedding by DISAE, chemical compound embedding, and attention pooling with multilayer perceptron (MLP) to model CPIs. Once processed through ALBERT, each protein is represented as a vector of dimension 210 by 312, with the sub-vector of length 312 representing each distilled triplet. The protein embedding matrix is compressed using ResNet during the chemical protein interaction prediction. The compressed protein embedding matrix and the chemical compound embedding are provided as input to the attentive pooling layer, which calculates the relevance scores between protein and chemical compound features. The attention scores are then multiplied with the concatenated feature vector of the protein and chemical compound embeddings and passed through a multilayer perceptron (MLP) to make the final CPI prediction. The model is trained using a binary cross-entropy loss function and optimized using the Adam optimizer.

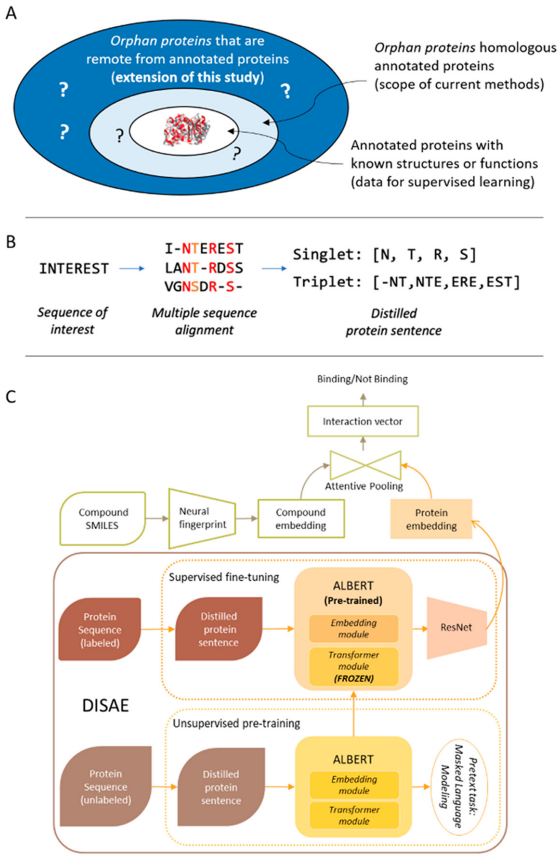

FIGURE 3.6: DISAE Architecture (note that we use the pre-trained DISAE (only))

3.3.3 Graph Neural Network (GNN)

We define a complex GNN whose message passing is based on Graph Isomorphism Network (GIN) called GIN Convolution or GINConv. The key thought behind GINConv is to characterize a generic message-passing scheme to upgrade the node representations in any graph neural network (GNN) design. GINConv accomplishes this by aggregating the node features of a node's neighbors and utilizing this aggregated feature as the input to a multilayer perceptron (MLP) with batch normalization and ReLU activation. The yield of the MLP is added to the node's original features to obtain the updated node representation. The aggregation operation in GINConv is characterized as a sum of the node features of a node's neighbors multiplied by a learnable weight. This weight is initialized to 1 and learned during training, allowing GINConv to learn a nonlinear aggregation function to capture complex node dependencies. One of the fundamental benefits of GINConv is its capacity to perform permutation-invariant aggregation, meaning that the order in which the nodes are handled does not influence the output of the layer. This property makes GINConv well-suited for handling graph isomorphism, where two graphs have the same structure but different node labels, like chemical compound graphs.

The GNN module includes an embedding layer for each allowable node feature (atom type, degree,

formal charge, hybridization, aromaticity, and chirality tag). The embeddings are initialized with Xavier uniform initialization. The GNN module also includes a list of GINConv modules and a list of batch normalization layers. The forward function of the GNN module takes as input the node features, edge indices, and edge features. The node embeddings are computed by passing the input through the GINConv layers, followed by batch normalization and ReLU activation. The pooling operation is then applied to the resulting node embeddings, depending on the value of JK. Finally, the output of the pooling operation is passed through a linear layer with dropout and sigmoid activation to obtain the chemical embeddings.

3.3.4 ResNet

For our study, we present a customized ResNet model with a ResNetEncoder as the main component responsible for encoding the input into a feature vector. The ResNetEncoder comprises multiple ResNet layers, each containing either ResNetBasicBlocks or ResNetResidualBlocks, to learn hierarchical representations of the input at various levels of abstraction. Skip connections are utilized to preserve gradients during training and allow for learning deeper and more complex features. The ResNetEncoder also incorporates a gate module consisting of a convolutional layer with a stride of 2 to downsample the input, followed by batch normalization, activation function, and max-pooling layer to reduce the dimensions further. We explain that a skip connection, also known as a residual connection, is a direct connection between two layers that allows for bypassing one or more intermediate layers. This mechanism addresses the vanishing gradient problem in deep neural networks, enabling gradients to flow back to earlier layers for better optimization.

In our model, once we have obtained the representations from DISAE, we pass these representations to the resnet to obtain the new features.

3.3.5 Attentive Pooling

Now the resnet obtained features for the protein embedding and the chemical embedding from the GNN are passed to an Attentive Pooling network. The Attentive Pooling network aims to combine information from chemical and protein embeddings to capture the relevant interactions between them. The model captures the interdependencies and interactions between chemicals and proteins by incorporating both embeddings through the attention mechanism. Specifically, for the chemical embedding, the final chemical embeddings are computed by considering the protein embedding based on their relevance to the chemical embedding. Similarly, for the protein embedding, they are computed based on their relevance to the chemical embedding. This allows for a more comprehensive understanding of their relationship and improves the accuracy of predicting chemical-protein binding affinity.

3.3.6 Hyperparameters

The hyperparameters involved in training include:

1. **Classes**: The number of classes the model is trained on varies from 2, 3, 5, 7, and 10.
2. **Loss Balancing Methods**: The different techniques used to obtain the loss balancing weights.
3. **Dropout**:The different techniques used to obtain the loss balancing weights. To prevent the deep learning models from overfitting, some neurons from the deep architecture are dropped from the model as part of regularization, called dropout. It ranges from 0.1 to 1, where 0.1 means dropping only 10% of the neurons from the whole architecture will be dropped.
4. **Learning Rate & Learning Rate Scheduler**: As opposed to training of traditional machine learning models, which involved keeping a constant learning rate throughout the training process, the training of deep neural networks now consists in having a learning rate scheduler that changes the learning rate at every time step when the gradients, loss are calculated based on different mathematical functions. This is advantageous as faster convergence to the optimal solution is possible, gives better generalizability, and avoids being stuck in the same local minima. This concept was first introduced by Kirkpatrick et al. in [19] and later proposed by Bottou in [4] and in [5] as part of works to improve neural network performance. In our case, we deploy the cosine annealing with warm restarts. Loshchilov and Hutter [23] proposed cosine annealing learning with warm restarts as part of their work - "SGDR: Stochastic Gradient Descent with Warm Restarts." It involves periodically changing the learning rate, gradually decreasing the learning rate, and then slowly increasing it back to the original value after some time steps. Fig. 3.7 depicts the working of the scheduler. The authors conclude that the cosine annealing with warm restarts learning rate scheduler leads to better generalization, robust learning, and state-of-the-art results for the model.
5. **Batch Size**: Batch size refers to the subset of the batch used for training the model in a single time step. Usually, higher batches imply better generalization power for the model. We use the batch size of 256, as a larger batch size causes memory problems.
6. **Frozen**: While training our deep learning model, since we utilize the pretrained DISAE to obtain representations of proteins since it has already been trained, another way of regularizing is to freeze certain parts of the DISAE model, that is, to not update their weights while training. In our case, we perform the linear probing fine-tuning strategy by Kumar et al. [20], and hence we only freeze the head.

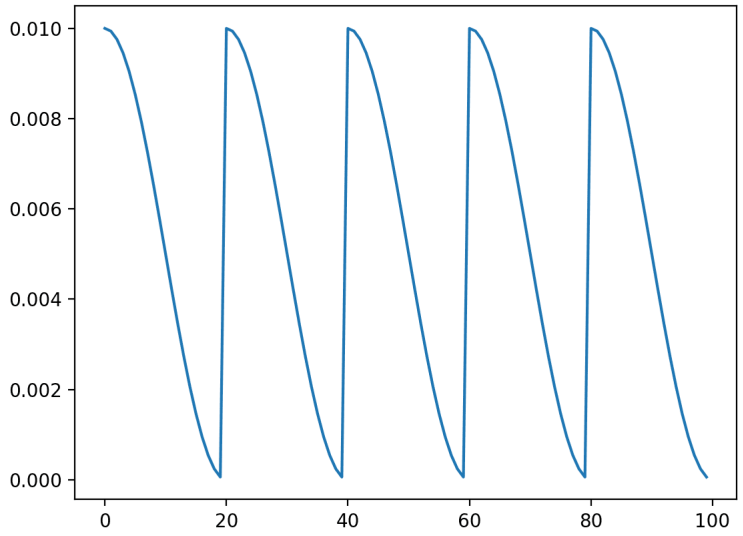

FIGURE 3.7: Cosine Annealing with Warm Restarts (from https://machinelearningmastery.com/)

7. **Regressor**: There are various options for a regressor model, in our case we explore only two:

   (a) **Random Forest (RF) Regressor** [3]

   (b) **Extreme Gradient Boosting (XGBoost) Regressor** [9]

3.3.7 Evaluation Metrics

For the evaluation of the classification model is performed using the Accuracy, Precision Recall (PR AUC) curve, and area under the receiver operating characteristic curve (ROC AUC). Whereas the regression model is evaluated on the mean squared error, mean absolute error and R2 score.

3.4 Results & Discussions

The various models are trained on the DLS server of Hunter College, City University of New York (CUNY), because of it's GPU capabilities. The various models run are shown in the Table 3.2, with the different hyperparameters as described in the Hyperparameters section.

As we can see from table 3.3, the models fail to achieve generalizability when the batch size is 32. Therefore it seemed appropriate to increase the batch size from 32 to the highest possible (256) because of memory constraints. The performance improvement is around 16.4%. When

Model	Batch Size	Classes	Type	Dropout	LR	Regressor	Frozen	Epochs
1	32	2	STL	0.7	0.05	XGB	None	40000
2	32	2	STL	0.4	0.05	XGB	None	40000
3	32	2	STL	0.4	0.001	XGB	None	40000
4	256	2	STL	0.4	0.001	XGB	None	40000
5	32	2	STL	0.4	0.005	XGB	None	40000
6	32	2	STL	0.4	0.05	XGB	None	40000
7	256	2	STL	0.4	0.005	XGB	None	40000
8	256	2	STL	0.4	0.0005	XGB	None	80000
9	256	2	STL	0.4	0.001	XGB	None	40000
10	256	2	STL	0.4	0.0005	XGB	None	40000
11	256	2	STL	0.4	0.0005	XGB	Head	40000
12	256	3	STL	0.4	0.0005	XGB	None	40000
13	256	5	STL	0.4	0.0005	XGB	None	40000
14	256	7	STL	0.4	0.0005	XGB	None	40000
15	256	10	STL	0.4	0.0005	XGB	None	40000
16	256	7	STL	0.4	0.0005	XGB	Head	80000
17	256	7	STL	0.4	0.0005	RF	None	80000
18	256	7	STL	0.4	0.0005	RF	Head	80000
19	256	7	MTL (baseline)	0.4	0.001	RF	None	40000
20	256	7	MTL (RLW)	0.4	0.001	RF	None	40000
21	256	7	MTL (UW)	0.4	0.001	RF	None	40000

Table 3.2: Models Trained on various Hyperparameters

changing the learning rate, it is observed that the models with lower learning rates achieve better results than those with higher learning rates. Higher learning rate models fail to converge to the optimal solution. As we increase the number of epochs or time steps, we naturally obtain better performance since the model continues training for longer. The dropout rate also significantly impacts the training as a higher dropout rate, like 0.7, fails to match the model's performance with a lower dropout rate, like 0.4. This is because when we drop 70% of the model neurons, the model cannot learn anything. The features generated for the regression tasks are better when freezing the layers. Changing the number of classes from binary (2) to multi (3, 5, 7, 10) improves the generalizing power of the models for the regression task. As the MAE, MSE, and R2 scores are improved, the classification model has to model more divisions as we increase the number of classes. Regression mathematically is the case of classification where the number of divisions is infinite. The random forest regressor performs better than the xgboost regressor for our task as it is less prone to overfitting. In the STL, the loss from the regression doesn't impact the loss from the classification, and the classification model is independently trained and evaluated. In contrast, in the MTL, the loss from the regression is taken into account. In the case of the MTL, the best task-balancing method is the uncertainty weights. This is because these weights are learned during the training process itself.

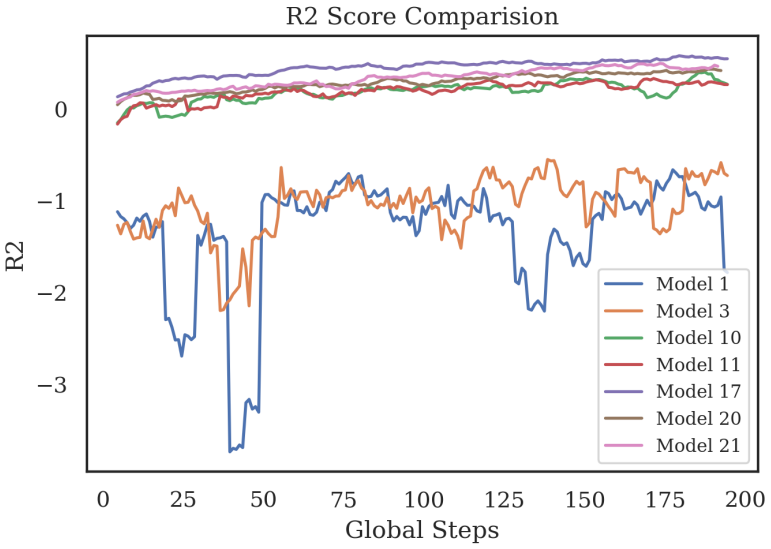

FIGURE 3.8: R2 Score

Model	Accuracy	ROC AUC	PR AUC	MAE	MSE	R2
1	0.595±0.045	0.618±0.074	0.595±0.044	3.301±0.464	19.797±10.855	-1.371±2.049
2	0.595±0.043	0.608±0.092	0.596±0.043	3.259±0.479	19.125±10.837	-1.122±1.132
3	0.632±0.058	0.61±0.153	0.632±0.056	3.062±0.491	17.528±10.754	-1.051±1.059
4	0.756±0.046	0.761±0.044	0.756±0.046	2.36±0.218	10.825±3.186	0.116±0.198
5	0.601±0.041	0.609±0.07	0.603±0.04	3.303±0.495	19.948±10.417	-1.186±1.724
6	0.595±0.039	0.614±0.074	0.594±0.039	3.241±0.513	19.164±12.21	-1.281±1.693
7	0.607±0.061	0.565±0.2	0.609±0.059	2.759±0.187	13.699±3.189	-0.142±0.167
8	-	-	-	3.087±0.16	16.707±3.384	-0.391±0.249
9	0.735±0.044	0.74±0.046	0.735±0.044	2.452±0.213	11.491±3.132	0.068±0.193
10	0.777±0.047	0.784±0.041	0.778±0.046	2.258±0.211	10.254±3.032	0.176±0.188
11	0.778±0.045	0.783±0.041	0.778±0.044	2.271±0.211	9.994±2.985	0.187±0.19
12	0.642±0.052	0.632±0.02	0.731±0.039	2.147±0.217	9.373±2.75	0.248±0.172
13	0.483±0.054	0.423±0.04	0.676±0.034	2.103±0.261	9.352±3.305	0.254±0.209
14	0.394±0.048	0.344±0.05	0.646±0.028	2.064±0.264	8.832±2.977	0.299±0.184
15	0.315±0.044	0.295±0.01	0.619±0.024	2.086±0.259	9.023±2.861	0.276±0.19
16	0.4±0.047	0.39±0.07	0.649±0.027	2.07±0.253	9.343±3.528	0.263±0.239
17	0.424±0.047	0.479±0.071	0.663±0.027	**1.738±0.223**	**6.438±2.425**	**0.496±0.116**
18	0.425±0.046	0.478±0.07	0.664±0.027	1.746±0.221	6.551±2.399	0.493±0.13
19	0.34±0.047	0.614±0.027	0.34±0.047	2.094±0.243	8.498±2.455	0.324±0.13
20	0.316±0.045	0.6±0.026	0.316±0.045	2.161±0.22	8.807±2.213	0.286±0.134
21	0.343±0.05	0.616±0.029	0.343±0.05	2.062±0.249	8.257±2.259	0.332±0.139

TABLE 3.3: Results

As we can see the STL model 17 performs the best because it has been fine-tuned multiple times on the LPFT strategy, therefore more fine-tuning is required for the MTL tasks.

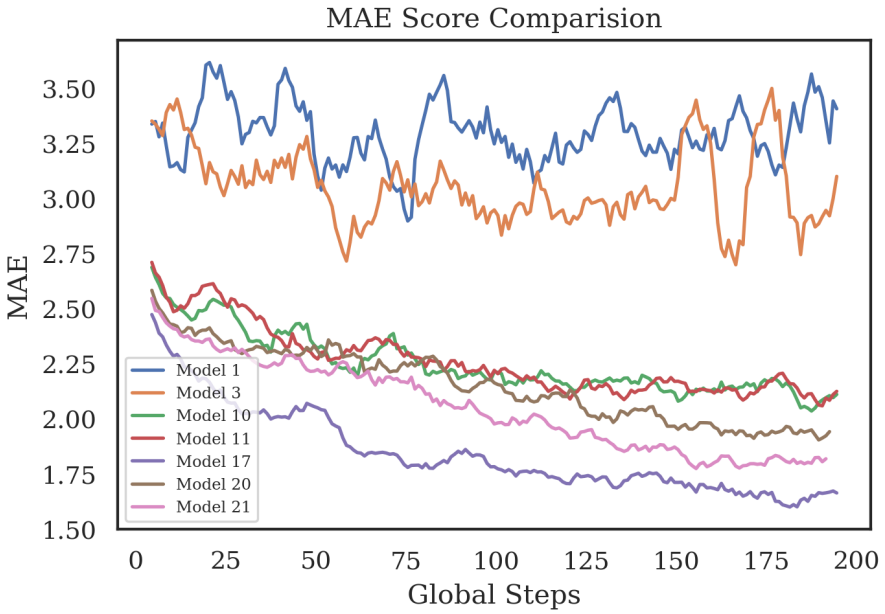

FIGURE 3.9: MSE

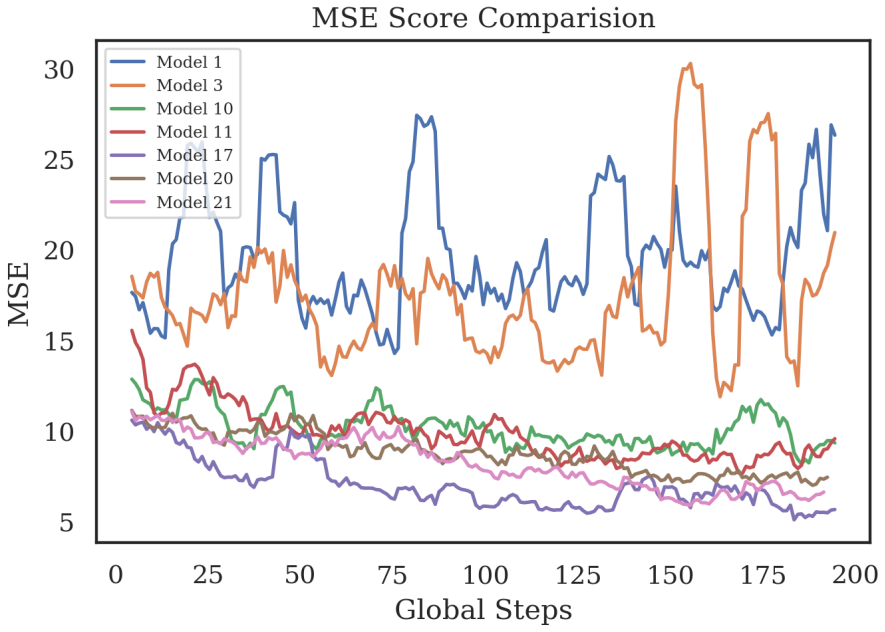

FIGURE 3.10: MAE

Chapter 4

Deep Regression-Based Prediction of Chemical Protein Interaction

4.1 Introduction

In this Chapter, we propose and investigate a deep regression-based model for prediction of chemical protein interaction. It uses the same data as the deep classification model did in Chapter 3. But instead of using traditional machine learning algorithms such as Random Forests and Support Vector Machines. It uses deep neural networks, and instead of using the DISAE model it uses the ESM2 (Evolutionary Scale Modeling) model to obtain the protein sequence embedding. The chapter explores the following:

1. Evolutionary Scale Modeling (ESM2)
2. Deep Regression-Based Model Architecture
3. Results and Discussions
4. Conclusion

4.2 Evolutionary Scale Modeling

The ESM-2 model is an excellent example of how the use of deep learning can assist in predicting the protein structure on an evolutionary scale. The authors illustrate that using large language models (LLMs) to theorize the protein structure from the primary protein sequence directly facilitates a significant speed-up in predicting high-resolution structures. The language model is trained on MGnify90, which contains over 617 million proteins. As it is scaled up from 8 million

to 15 billion parameters (the largest number of parameters a protein language model is known to have had), it can predict the three-dimensional structure of a protein at the resolution of individual atoms. The model predicts the structures of over 617 million metagenomic proteins, including more than 225 million high-confidence predictions. The authors also found that the models developed attention patterns that resembled the protein's residue-residue contact map of amino acids. This indicated the rise of low-resolution protein structures in the models. Further, the authors examined whether atomic-resolution information was present in the models by projecting spatial coordinates for each of the atoms from the internal representations of the language model using an equivariant transformer. The evaluation of this projection on experimentally determined protein structures showed that the projection's accuracy increased as the model's parameter scale increased, indicating the emergence of atomic-level details about the protein structure in the models. The Fig. 4.1 depicts the ESM2 model for the prediction of the protein structure.

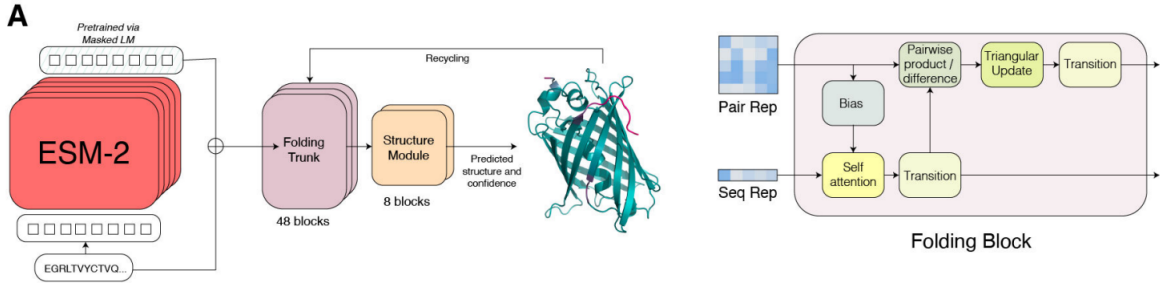

FIGURE 4.1: ESM-2 Architecture

4.3 Deep Regression Architecture

We experimented with a varying amount of deep learning models and algorithms like Long Short-Term Memory (LSTMs), Bi-directional LSTMs (Bi-LSTMs), Multi Head Attention Mechanisms, Attentive Pooling, Transformers, and deep convolutional neural networks (DCNNs) such as ResNet. The Fig. 4.2 shows the best model architecture for the results obtained. We see that the ResNet based model performs the best as we will see in the Results and Discussions section. As we can clearly see the regression model has two different distinct changes to it -

1. The DISAE model has been replaced with the ESM-2 model to obtain the protein embeddings.

2. We longer make use of feature extraction to obtain the features for Machine Learning based predictions, instead we allow the deep learning model to extract the relevant features from it's embeddings optimized on the loss function.

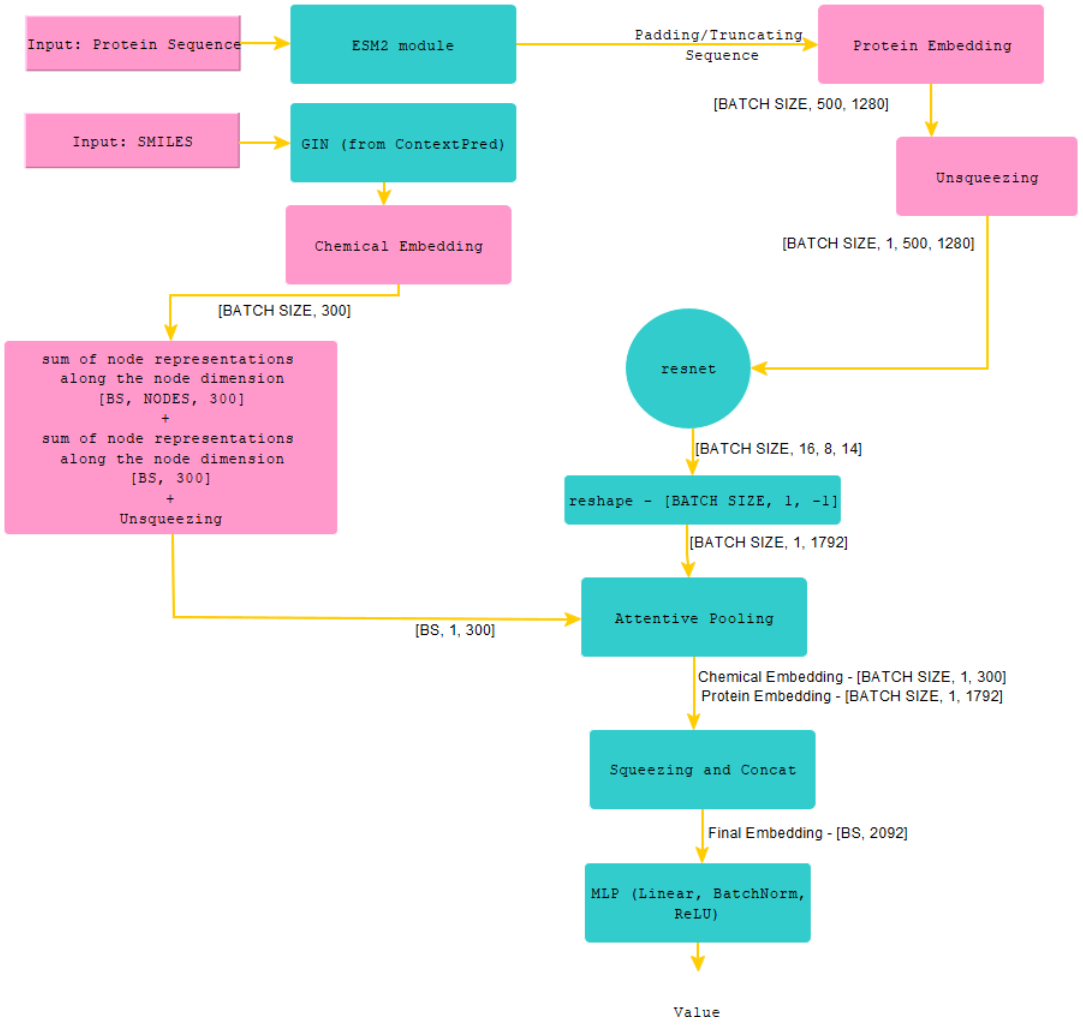

FIGURE 4.2: Deep Regression-based Architecture

4.3.1 LSTMs & Bi-LSTMs

LSTMs, or Long Short-Term Memory networks, are recurrent neural network (RNN) architectures designed to capture long-term dependencies in sequential data. LSTMs are widely used in various fields, including natural language processing, time series analysis, and image captioning.

The need for LSTMs arises from the constraints of traditional RNNs in capturing long-range dependencies. RNNs suffer from the "vanishing gradient" issue. The impact of earlier inputs declines rapidly as the network processes the sequence, causing difficulty learning and retaining data and information from distant past states. LSTMs overcome this issue by introducing a memory cell and different gating mechanisms facilitating them to remember or forget information over longer sequences. While LSTMs offer several advantages but have some limitations, particularly their unidirectional nature. Bidirectional LSTMs (Bi-LSTMs) were introduced to overcome this limitation of learning dependencies in a unidirectional nature. The bidirectional nature of Bi-LSTMs allows them to extract rich and diverse features from the protein sequence.

In our case, we process the protein embedding obtained from the ESM-2 model through the LSTM and Bi-LSTMs to capture long range dependencies in the network. We replace the resnet model as shown in Fig. 4.2 with the LSTM and Bi-LSTM model for processing the protein embeddings.

4.3.2 Attention Mechanisms & Transformers

Among the different Attention Mechanisms present, we deploy:

1. Multi Head Attention: In case of multi head attention, the input sequences are processed through several iterations of attentive pooling, this works well for when their are long sequences.
2. Attentive Pooling: Attentive Pooling as mentioned in the Chapter 3 subsection 3.3.5, captures the dependency between the protein and the chemical.

Transformers are the architectures that utilize these attention mechanisms to better understand the relations present within a sequence. Transformers stand out in learning the long-range dependencies among the sequence by allowing different parts to directly attend to other parts within the sequence, regardless of their distance.

4.4 Results and Discussions

The models were trained on the 'DLS' server of the Hunter College, City University of New York on the 'NVIDIA Tesla V100 PCIe' GPUs of 32 GBs of memory. The models were evaluated on the following metrics:

1. Pearson Correlation (r): The Pearson correlation coefficient quantifies the linear association between two variables. It is a numerical value ranging from -1 to 1, wherein -1 signifies a complete negative linear correlation, 0 indicates no correlation, and +1 represents an absolute positive correlation. In the equation 4.1, the x_i & y_i refer to the values of the two variables.
$$r = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2 \sum_{i=1}^{n}(y_i - \overline{y})^2}} \tag{4.1}$$
2. Spearman Correlation (ρ): Spearman correlation is a statistical measure that assesses the monotonic relationship between two variables. Monotonic relationships are those in which the variables tend to change together consistently, regardless of whether the relationship is positive or negative. The Spearman correlation coefficient ranges from -1 to 1, where -1

indicates a perfect negative monotonic correlation, 0 suggests no monotonic correlation, and +1 represents a perfect positive monotonic correlation. In 4.2, n is the total number of samples and d is the pairwise distance between the ranks of the variables.

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)} \tag{4.2}$$

3. MAE: Mean Absolute Error refers to mean of the difference between the actual value vs the predicted value. MAE is a versatile measure, easily interpretable for evaluation of the models.

$$MAE = \frac{\sum_i^N |x_i - y_i|}{N} \tag{4.3}$$

4. MSE: Mean Squared Error refers to the square of the difference between the actual value vs the predicted value.

$$MSE = \frac{\sum_i^N (x_i - y_i)^2}{N} \tag{4.4}$$

5. RMSE: Root Mean Square is simply the root of the MSE value.

$$RMSE = \sqrt{(\frac{\sum_i^N (x_i - y_i)^2}{N})} \tag{4.5}$$

The table 4.1 decipt the various models trained as well as a comparision with the classification-based model from Chapter 3 that utilizes MTL. The Model 6 uses the Multi Head Attention on the protein sequence embedding itself. And the Model 7 refers to the earlier classification-based MTL-deep learning model.

Model	Protein Processing Algorithm	Attention Mechanism
1	ESM2 + ResNet	Attentive Pooling
2	ESM2 + ResNet	Multi Head Attentive Pooling
3	ESM2 + BiLSTM	Attentive Pooling
4	ESM2 + LSTM	Attentive Pooling
5	ESM2 + Transformer	Attentive Pooling
6	ESM2 + Multi Head Attention	Attentive Pooling
7	DISAE + ResNet	Attentive Pooling

TABLE 4.1: Various models

The Table 4.2 shows the final results obtained after training the model for around 50 epochs with the AdamW optimizer, MSE loss being optimized, the Cosine Annealing with Warm Restarts learning rate scheduler and the learning rate of 5×10^{-4}. As we can clearly see the ESM-2 based models outperform the DISAE-based model, even after the best DISAE-based model has been pretrained and fine tuned numerous times. This is due to the fact that the ESM-2 model has been trained on a large dataset and also it uses the protein sequence directly instead

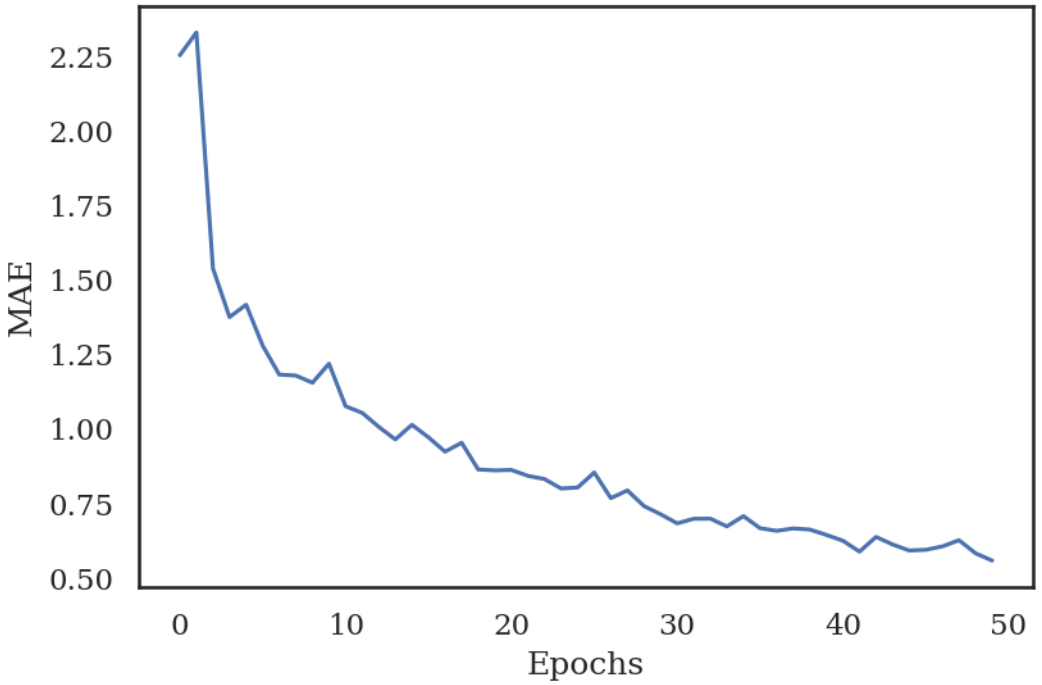

FIGURE 4.3: Best Model MAE plot

of distilling the sequence into triplets, which has shown loss of information. Moreover, the ESM-2 model is a protein language model with a large number of parameters ranging from a couple millions upto 15 billion, therefore it is able to learn the complexities of interactions within most of the proteins. We also note that the ResNet model used for extraction of features from the protein embeddings performs better than any LSTM, BiLSTM, or Transformer based models, because of mainly the better ability of ResNet models to features for simpler processing. ResNet employs convolutional layers explicitly crafted to capture localized features and patterns in protein sequences. By using convolutional filters, the network examines the input protein sequence, effectively recognizing significant local motifs and patterns that play a crucial role in determining the overall structure and function of the protein binding pockets. This feature of convolutional layers enables ResNet to extract fundamental features like secondary structure elements, conserved regions, and localized interactions among amino acids.

Model	MAE	MSE	Pearson Correlation	Spearman Correlation
1	**0.562±0.049**	**0.781±0.083**	**0.813±0.063**	**0.802±0.049**
2	0.783±0.052	1.175±0.301	0.746±0.091	0.705±0.089
3	0.808±0.095	1.175±0.301	0.716±0.095	0.682±0.097
4	1.094±0.065	2.05±0.205	0.698±0.076	0.653±0.079
5	1.101±0.06	2.097±0.197	0.664±0.074	0.622±0.074
6	1.344±0.1	3.13±0.37	0.66±0.126	0.62±0.11
7	1.738±0.223	6.438±2.425	-	-

TABLE 4.2: Results from the various models

The underperformance of the Multi-Head Attentive Pooling and Transformer-based models can be attributed to the introduction of increased complexity. Deep learning models have a fragile equilibrium between model complexity and generalization. When a model becomes unduly complex, it risks overfitting the training data, hindering its ability to learn effectively. In the case of Multi Head Attentive Pooling and Transformer-based models, the introduction of multiple attention heads constraining the model's capacity to learn long-range dependencies can increase the overall complexity of the model. Although these architectures have worked wonders in the textual domain of natural language processing, their application to protein sequences can sometimes lead to overfitting. The complexity of the model enables it to capture and memorize intricate details and noise in the training set, resulting in reduced performance on new protein embeddings.

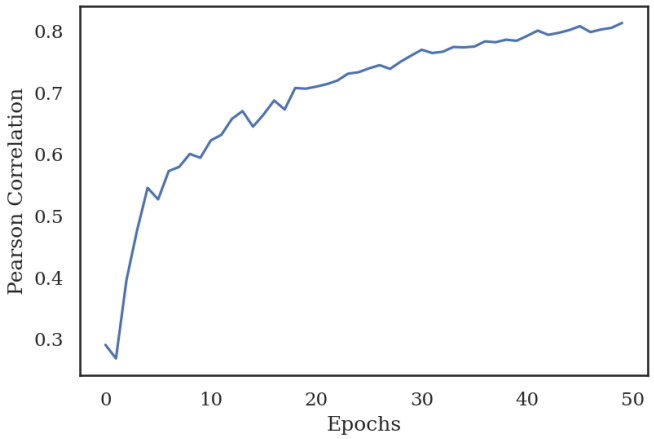

FIGURE 4.4: Pearson Correlation

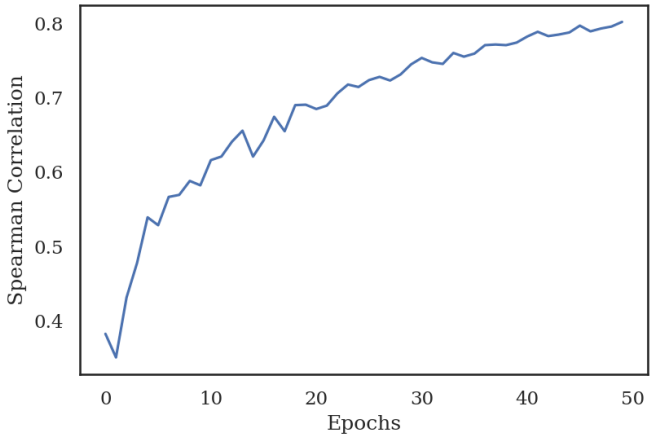

FIGURE 4.5: Spearman Correlation

4.5 Conclusion

This study has successfully demonstrated the following key findings:

1. The development of a deep learning-based regression model that exhibits high accuracy in predicting the binding affinity between ligands/chemicals/potential drugs and proteins. This computational model can be effectively integrated into traditional drug discovery pipelines, facilitating faster and more accurate predictions.

2. The ESM-2 model has demonstrated its efficacy in predicting the binding affinity of proteins and chemicals. Furthermore, its applicability can be extended to other tasks such as protein-protein interaction prediction and residue contact map interaction prediction, eliminating the complete protein structure required by pre-existing deep learning algorithms and relying solely on the protein sequence for interaction computation.

3. The ResNet model has demonstrated proficiency in extracting relevant features from protein data. Designed explicitly for protein analysis, the ResNet model can be employed with ESM-2 embeddings to facilitate the downstream tasks mentioned above.

Bibliography

[1] Mariette Awad et al. "Support vector regression". In: *Efficient learning machines: Theories, concepts, and applications for engineers and system designers* (2015), pp. 67–80.

[2] Jeremy M Berg, John L Tymoczko, and Lubert Stryer. *Biochemistry (Loose-Leaf)*. Macmillan, 2007.

[3] Gérard Biau and Erwan Scornet. "A random forest guided tour". In: *Test* 25 (2016), pp. 197–227.

[4] Léon Bottou. "Stochastic gradient descent tricks". In: *Neural Networks: Tricks of the Trade: Second Edition* (2012), pp. 421–436.

[5] Léon Bottou, Frank E Curtis, and Jorge Nocedal. "Optimization methods for large-scale machine learning". In: *SIAM review* 60.2 (2018), pp. 223–311.

[6] Tian Cai et al. "Msa-regularized protein sequence transformer toward predicting genome-wide chemical-protein interactions: Application to gpcrome deorphanization". In: *Journal of chemical information and modeling* 61.4 (2021), pp. 1570–1582.

[7] Nigel Chaffey. *Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K. and Walter, P. Molecular biology of the cell. 4th edn.* 2003.

[8] Deliang Chen et al. "Regulation of protein-ligand binding affinity by hydrogen bond pairing". In: *Science advances* 2.3 (2016), e1501240.

[9] Tianqi Chen et al. "Xgboost: extreme gradient boosting". In: *R package version 0.4-2* 1.4 (2015), pp. 1–4.

[10] Zhao Chen et al. "Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks". In: *International conference on machine learning*. PMLR. 2018, pp. 794–803.

[11] UniProt Consortium. "UniProt: a worldwide hub of protein knowledge". In: *Nucleic acids research* 47.D1 (2019), pp. D506–D515.

[12] Sandro Cosconati et al. "Virtual screening with AutoDock: theory and practice". In: *Expert opinion on drug discovery* 5.6 (2010), pp. 597–607.

[13] Mindy I Davis et al. "Comprehensive analysis of kinase inhibitor selectivity". In: *Nature biotechnology* 29.11 (2011), pp. 1046–1051.

[14] Long Duong et al. "Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser". In: *Proceedings of the 53rd annual meeting of the Association for Computational Linguistics and the 7th international joint conference on natural language processing (volume 2: short papers)*. 2015, pp. 845–850.

[15] Ran Friedman. "Computational studies of protein–drug binding affinity changes upon mutations in the drug target". In: *Wiley Interdisciplinary Reviews: Computational Molecular Science* 12.1 (2022), e1563.

[16] Hans-Dieter Holtje et al. *Molecular modeling*. Vol. 5. Wiley-VCH Weinheim, Germany, 2003.

[17] Kenneth A Johnson and Roger S Goody. "The original Michaelis constant: translation of the 1913 Michaelis–Menten paper". In: *Biochemistry* 50.39 (2011), pp. 8264–8269.

[18] Alex Kendall, Yarin Gal, and Roberto Cipolla. "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7482–7491.

[19] Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. "Optimization by simulated annealing". In: *science* 220.4598 (1983), pp. 671–680.

[20] Ananya Kumar et al. "Fine-tuning can distort pretrained features and underperform out-of-distribution". In: *arXiv preprint arXiv:2202.10054* (2022).

[21] Zeming Lin et al. "Evolutionary-scale prediction of atomic-level protein structure with a language model". In: *Science* 379.6637 (2023), pp. 1123–1130.

[22] Shikun Liu, Edward Johns, and Andrew J Davison. "End-to-end multi-task learning with attention". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 1871–1880.

[23] Ilya Loshchilov and Frank Hutter. "Sgdr: Stochastic gradient descent with warm restarts". In: *arXiv preprint arXiv:1608.03983* (2016).

[24] Bryan McCann et al. "The natural language decathlon: Multitask learning as question answering". In: *arXiv preprint arXiv:1806.08730* (2018).

[25] David Mendez et al. "ChEMBL: towards direct deposition of bioassay data". In: *Nucleic acids research* 47.D1 (2019), pp. D930–D940.

[26] Jaina Mistry et al. "Pfam: The protein families database in 2021". In: *Nucleic acids research* 49.D1 (2021), pp. D412–D419.

[27] Sebastian Ruder. "An overview of multi-task learning in deep neural networks". In: *arXiv preprint arXiv:1706.05098* (2017).

[28] Oleg Trott and Arthur J Olson. "AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading". In: *Journal of computational chemistry* 31.2 (2010), pp. 455–461.

[29] W Patrick Walters, Matthew T Stahl, and Mark A Murcko. "Virtual screening—an overview". In: *Drug discovery today* 3.4 (1998), pp. 160–178.

[30] Tianhe Yu et al. "Gradient surgery for multi-task learning". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 5824–5836.

[31] Yu Zhang and Qiang Yang. "A survey on multi-task learning". In: *IEEE Transactions on Knowledge and Data Engineering* 34.12 (2021), pp. 5586–5609.

[32] Jie Zhou et al. "Graph neural networks: A review of methods and applications". In: *AI open* 1 (2020), pp. 57–81.