Prediction of Protein-Ligand Binding Affinity using Residual to Atomic level Graph Neural Networks

Cong Liu[1]

MSc. Data Science and Machine Learning

Carlo Ciliberto

Robert McHardy

Submission date: 12.09.2022

[1]**Disclaimer:** This report is submitted as part requirement for the MSc. Data Science and Machine Learning at UCL. It is substantially the result of my own work except where explicitly indicated in the text. The report will be distributed to the internal and external examiners, but thereafter may not be copied or distributed except with permission from the author.

Acknowledgements

I would like to first thank my supervisor Professor Dr. Carlo Ciliberto whose expertise is invaluable in formulating the project direction and planning. Your insightful suggestions helped me to better plan my thesis. I am very thankful to my industrial supervisor, Robert McHardy, for his insightful and skilful suggestions during my research project. It is your constructive suggestions that have given me good training both academically and in engineering. I would also like to thank all my colleagues (especially Mr. Yunguan Fu and Mr. Antoine Delaunay) in InstaDeep during my internship for their valuable time as well as for always kind and supportive suggestions on my project. I would like to thank everyone who took the time to proofread this dissertation. Lastly, I want to thank the wonderful Jingxuan Tu and my parents for their mental and financial support during my studies at UCL.

Disclaimer for Code Availability

This dissertation is completed in cooperation with InstaDeep Ltd [1]. Codes are not available until requested following the policy.

[1] https://www.instadeep.com

Abstract

Drug discovery is a vital and promising research area in the development of human beings. Penicillin (FLEMING 1944) was discovered and greatly improved the ability of people to fight germs. Aspirin keeps people safe from most physical pain. These brilliant drugs have been discovered for the benefit of all mankind. Molecule properties and interactions between molecules (e.g. binding affinity and binding prediction) have been identified as key factors in drug discovery. Recent research has shown the great impact brought by deep learning on predicting molecule properties and interactions. The successes of such computational methods mostly rely on good-quality molecule representations. However, most existing deep learning methods are based only on amino acid-level input; thus, biomolecular structural information at the atomic level is not fully utilized, meaning that essential interactions among atoms are also neglected. Given the natural geometric forms of molecules, we propose an atomic-level graph neural network for modelling the geometric structure of molecules as well as the interaction between molecules and thus deriving a better representation of molecules.

The contributions made by this dissertation are listed as 1. Provide clear data processing steps for the PDBbind data set v2016. 2. Investigate the performance of models from transferring amino acid-level graphs to atomic-level graphs by three-stage experiments. Furthermore, we also provide results from other work for comparison of the binding affinity task using the PDBbind data set.

Contents

Chapter 1

Introduction

1.1 Motivation

The interaction between molecules, such as binding affinity, has been shown to be vital in drug design (Beydon et al., 2000). The properties of molecules, such as shapes, often determine the role of a compound in terms of molecular interactions with selected targets (Kortagere et al., 2009). Drug designs have relied on discovered molecular properties, thus, most resources have been spent on chemical experiments to identify accurate quantitative molecular interactions (such as binding affinity) as well as molecular characteristics. However, chemical experiments are laborious and extremely time-consuming (Cohen, 2002). Therefore, computational methods are developed to accelerate drug design and save resources. One of the computational methods that aim to predict the binding affinity between molecules via scoring functions is molecular docking. This method has greatly facilitated the research in terms of fast speed, but the accuracy is relatively low (Venkatachalam et al., 2003, J. Li et al., 2019).

Recent years have witnessed the great change brought by Deep Learning (DL) in fields like Computer Vision (CV), Natural Language Processing (NLP), and Recommender Systems (Chai et al. 2021, Torfi et al. 2020, S. Zhang et al. 2020). Deep learning is also introduced in bioinformatics because of its flexible requirement on data input, as well as great inference speed with current computational resources (high-performance GPUs and TPUs). In particular, Alphafold (Jumper et al. 2021) has been developed based on neural networks to predict protein structure with amino acid sequences and structural features, predicting precise protein structure in much

less time compared to experimental results.

Research on protein-ligand binding affinity prediction is focused on new drug design (Beydon et al. 2000). Previous deep learning-based methods showed their superiority over traditional docking methods such as Venkatachalam et al. 2003 and J. Li et al. 2019 through performance using recurrent neural network (RNN) or convolutional neural network (CNN) to process amino acid sequences of protein and ligand ((Karimi et al. 2018, Lee et al. 2018)). However, in previous research such as Öztürk et al. 2018, scientists separately input amino-acid sequences from protein and ligand in neural networks to derive the high-dimensional representations, while the explicit interactions between ligand and protein are neglected due to the sequential representations for proteins and ligands. Furthermore, given that molecules are in the forms of 3D structures and are composed of atoms, neural networks will lose both the information of interaction between atoms within the interface of the protein-ligand complex and the spatial structural information by only feeding sequences.

Molecules are composed of atoms in 3D space, while Graph Neural Networks (GNNs) are designed for non-euclidean data (Asif et al. 2021), which is suitable for processing the molecule data. Most works focus on using sequential information of amino-acids level data to generalize on unseen drug-target pairs (Öztürk et al. 2018), however, given that molecules are composed of atoms, the granular level of data could be further converted to the 3D atomic level graph to obtain more structural information. To utilize the full information from the data, we propose feeding chemical compounds such as protein and ligand molecules in the form of atomic-level graphs to GNNs, thus deriving better representations of the molecules than sequential inputs.

1.2 Aims and Approach

This dissertation aims to answer the research question of whether converting to atomic-level graphs will help to derive better hidden representations of molecules. To answer this research question, we focus on a specific protein-ligand binding affinity prediction task.

To systematically answer this research question, we propose a three-stage process, moving from amino acid-level graphs to atomic graphs. The first stage is to use the amino acid level graph for

the protein, that is, one node represents one residue. The second stage extracts backbone atoms from each residue in protein to represent nodes. The third stage uses full-atomic-level graphs in the protein to predict binding affinity. Since ligands are mostly short peptides without secondary structures, it is not sensible to divide ligand graphs into three stages, thus, ligand graphs are implemented as full atomic graphs in all three stages. The models are evaluated in the coreset of the PDBbind data set v2016 [1] and trained in the refined set of the PDBbind data set v2016, following S. Li, Zhou, Xu, Huang, et al. 2021. Besides three-stage experiments on atomic level graphs, quantitative analyses have been carried out to examine the node similarity in atomic graphs as well as the noise brought by large-size full atomic graphs. It further shows that it is vital to select a feature selection method for selecting desired features from derived graph representations.

1.3 Contributions

There are two main contributions provided by this dissertation. Firstly, to the best of our knowledge, there is no literature explicitly explain the detailed steps of pre-processing data files and extract graph from files in PDBbind data set, while graph data construction is very important in the processing of using Graph Neural Networks. This dissertation will provide detailed steps to follow to construct graphs from files in the PDBbind v2016 data set. Secondly, the effect of introducing more granular molecular representations is systematically evaluated by defining and evaluating models with increasing granularity. Lastly, we provide an ablation study on PDBbind data set v2016, aiming to provide protein-only and ligand only baselines. These baselines can be referenced in future work to compare the model performance, indicating whether models "truly" learn to predict or just by remembering specific proteins or ligands.

Furthermore, we conduct experiments to answer the following two research questions: 1) Do atomic-level GNNs have a more severe oversmoothing problem? 2) How to design a downstream model framework to fully utilize the obtained atomic-level graph representations?

[1] http://www.PDBbind.org.cn

Chapter 2

Background

This chapter will be divided into three sections. The first section 2.1 is for the concepts of biology that will be encountered in the following chapters. The second section will describe the background information of neural networks 2.2. Finally, the third section 2.3 will introduce graph neural networks and common network structures used in this dissertation.

2.1 Biological concepts

2.1.1 Atoms, Molecules and Covalent Bonds

Everything is made up of matter, and atoms are the fundamental unit of matter (McSween Jr & Huss, 2021). Every solid, liquid, gas, and plasma is composed of neutral or ionized atoms. Every atom is made up of atomic nuclei and electrons outside the nuclei. The number of electrons, as well as the weight of atomic nuclei, are defined by the element. For example, the relative atomic mass of an oxygen atom is 16 and its number of electrons outside the nucleus is 16, while the relative atomic mass of a hydrogen atom is 1 [1]. Figure 2.1 [2] shows the visualization of a generalized single atom.

As can be seen from the figure 2.1, the nucleus is made up of neutrons and protons. Neutrons are particles with no charge. While protons carry positive charged particles, electrons outside the nucleus carry negative charges. Electrons move in orbits around the nucleus.

Atoms can share electron pairs by forming covalent bonds between the atoms at particular

[1] Atomic relative mass read from `https://www.angelo.edu/faculty/kboudrea/periodic/structure_mass.htm`

[2] Extracted from `https://letstalkscience.ca/educational-resources/backgrounders/introduction-atom` on 5th, Sept. 2022

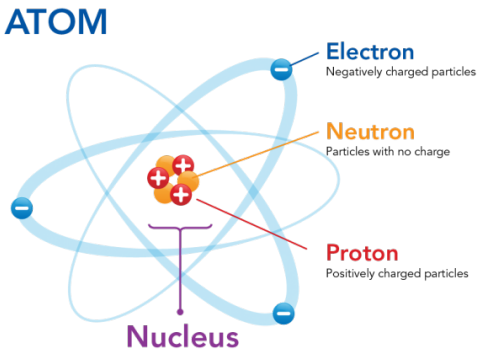

Figure 2.1: Example of a general atom.

angles, which will lead to the formation of molecules (Smith, 2020). The electrons shared between atoms are named "covalent bonds". (It is called "ionic bonds" when the shared electrons are between ions.) There are two types of covalent bonds in general; one is a non-polar covalent bond and the other is a polar covalent bond. For instance, if the molecule is composed of the same atoms, then these atoms have the same ability to attract electrons; thus, the covalent bonds are nonpolar. If the molecule is composed of different atoms that have different ability to attract electrons, then the shared electrons may prefer to be close to a specific type of atoms. Then the covalent bond is a polar covalent bond. Different covalent bonds will lead to different chemical properties, as can be seen in Figure 2.2 [3] showing an example of covalent bonds (ion bonds) within the water residue and Sodium chloride ($NaCl$) compound.

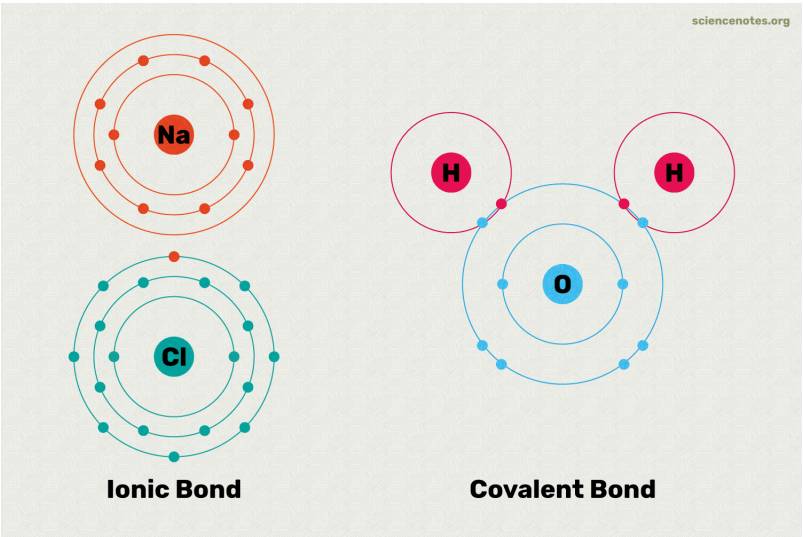

Figure 2.2: Covalent bonds (Ion bonds) in H_2O and $NaCl$

In the figure 2.2, it can be seen that Na shared one electron with Cl to form an ionic bond

[3]Extracted from `https://sciencenotes.org/ionic-vs-covalent-bonds` on 5 September 2022

between Na atom and Cl atom. However, two hydrogen atoms shared two electrons with one oxygen atom to form a covalent bond in one water molecule. In this example, the water molecule is a polar molecule because the covalent bonds in the water molecule are polar covalent bonds.

2.1.2 Protein and Amino Acids

Proteins are biological macromolecules that are essential components of living organisms. Human activities cannot be carried out without proteins. Proteins provide energy for life activities and some proteins with special functions, such as digestive enzymes, catalyse the body's digestion of protein macromolecules. Some proteins in the body can act as receptors for hormone molecules produced by the body's cells and thus respond to them and initiate further reactions (Radzicka & Wolfenden, 1995).

Amino acids are organic compounds that contain amino and carboxylic acid functional groups, along with a side chain (R group) specific to each amino acid. These amino and hydroxyl groups are linked by dehydration condensation to form one or more polypeptide chains. These polypeptide chains are then folded to produce a stable state to form the final protein. Following picture 2.3 shows one example of a polypeptide chain.

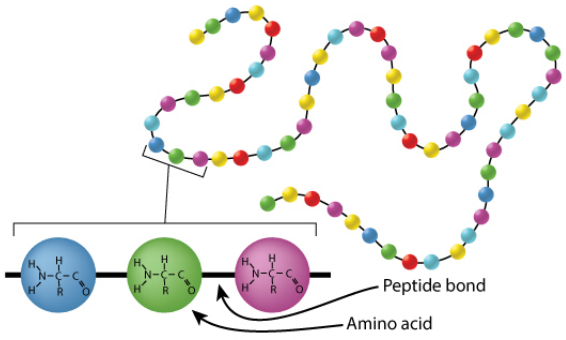

Figure 2.3: One fake example of polypeptide chain

In figure 2.3, amino acids combine with each other by peptide bonds, which are caused by the dehydration between amines and hydroxyls. Note that each amino acid is composed of the same types of atoms, as shown in figure 2.4 [4].

It can be seen that each amino acid is composed of N, O, H, C and side chain (R group). These atoms form the basic structure of amino acids. Special names are attributed to carbons in residues. The first Carbon that connects to the functional group is called carbon alpha C_α, while

[4]Extracted from `https://en.wikipedia.org/wiki/Amino_acid` on 5th, Sept. 2022

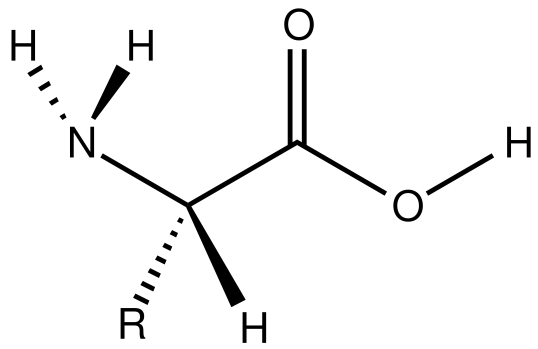

Figure 2.4: General structure of amino acids

carbon beta C_β is used to name the second carbon atom that connects to the functional group. Therefore, in each type of amino acid (except for Glycine, which only has one hydrogen atom in its side chain, that is, there is no C_β in Glycine), we have a fixed backbone structure that contains heavy atoms $N, O, C, C_\alpha, C_\beta$. The example of a three-peptide chain can be viewed in figure 4.3.

2.1.3 Ligand and Receptor

A ligand is a substance that can be bound to a biological molecule to form a complex. The resulting complexes can have biological effects or produce new compounds. A receptor is a biomolecule that can receive extracellular signals and act inside the cell. A receptor can only bind to one or a small number of ligands, and a ligand can only bind to one or a small number of receptors. Their binding allows biological information to be transmitted from the outside of the cell to the inside of the cell, causing the organism to respond. Receptors are generally glycoproteins that bind specifically and affectionately to ligands. The following figures 2.5 and 2.6 show the example ligand and example protein viewed by PyMOL (Schrödinger & DeLano n.d.).

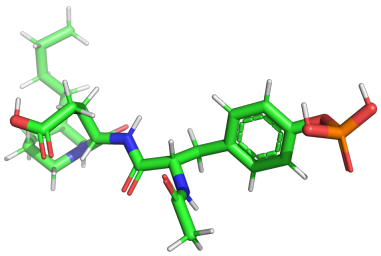

Figure 2.5: An example of ligand (from 1a1e complex) viewed by PyMOL

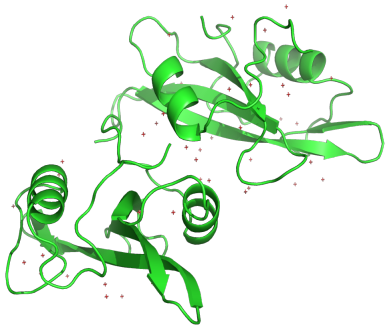

Figure 2.6: An example of receptor (protein from 1a1e complex) viewed by PyMOL

2.1.4 Binding Affinity

Note that the receptor and the ligand will bind to each other in some specific scenarios, the binding affinity is a measure of the strength of binding between the ligand and the receptor molecules (Jarmoskaite et al., 2020). In general, the binding affinity can be expressed as K_d and K_i. K_d stands for dissociation constant and is used to measure the equilibrium between the ligand-protein complex and the dissociated components. K_d can be represented as:

$$K_d = \frac{[P][L]}{[PL]} \tag{2.1}$$

where $[P]$ represents the concentration of free protein molecules, while $[L]$ represents the concentration of free ligand molecules. $[PL]$ represents the concentration of the protein-ligand complex. Therefore, if K_d is small, it means that more protein molecules bind with ligand molecules than free protein molecules dissolved in solution. However, large K_d indicates that protein and ligand molecules are present more as free molecules in solution that are not combined with each other. The inhibition constant K_i also represents a dissociation constant, but more narrowly for the binding of an inhibitor to an enzyme according to *The difference between Ki, KD, IC50, and EC50 values* n.d.. Note that ligand and protein will bind in a specific position to maintain the lowest energy of protein-ligand complex (Lengauer & Rarey 1996). The following figure 2.7 shows the example binding pose of the protein-ligand complex viewed by PyMOL.

Binding affinity is a key reference when selecting designed drugs. Drugs that have high binding affinities with the receptor protein will initially be selected as candidate drugs (C. Wang & Zhang 2017). Binding affinities can be evaluated by experiments, but most of these experiments are laborious and time-consuming (Cohen 2002). While the deep learning-based methods are relying

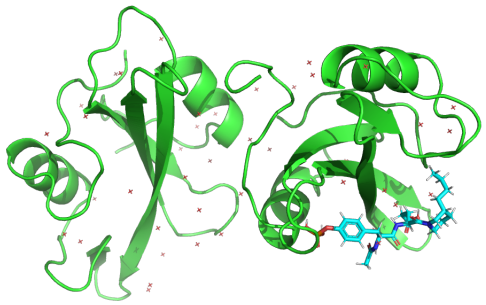

Figure 2.7: An example of protein-ligand complex (1a1e) viewed by PyMOL

highly on the quality of data. Thus, it is vital to find a well-established data set with experimentally evaluated binding affinities for deep learning-based methods.

2.2 Neural Networks

This section will mainly describe the basic concept of neural networks, together with one specific type of neural network (Convolutional Neural Networks) to have better intuition for the introduction to graph neural networks in 2.3. Common optimization methods are also introduced to facilitate the understanding of parameter updating in neural networks.

2.2.1 Overview of Neural Networks

Modern deep neural networks are first originated by an algorithm called "Perceptrons" introduced by Rosenblatt 1958. Perceptrons use artificial neurons to carry information and can be further trained by updating the weights of each neuron. Multilayer stacking of perceptrons successfully solved the XOR problem, which was a problem for the single-layer perceptron. Later on, changes were made to the multi-layer perceptrons to create neural networks suitable for processing image data like convolutional neural networks (LeCun & Bengio, 1998), text data like recurrent neural networks or transformer-based networks (Hochreiter & Schmidhuber, 1997, Vaswani et al., 2017) and even graph data like graph convolutional-based neural networks (Kipf & Welling, 2016, Veličković et al., 2017). Also, with the availability of high-performance computing resources, such as GPUs, a typical optimization method, backward propagation algorithm (Rumelhart et al., 1986) became widely used, facilitating the training of all kinds of neural networks.

2.2.2 Multi-Layer Perceptron

Multi-Layer Perceptron, also known as feedforward neural networks of fully connected neural networks, is a kind of simple structure neural network that has been widely used in areas. Multi-Layer Perceptron consists of a stacking of multiple layers of the perceptron. One layer of perceptron contains several neurons. The overview of 2 hidden-layer perceptrons can be viewed in figure 2.8. Each node in the figure 2.8 represents one neuron in the network, and each edge denotes the weight of the neuron. L th layer hidden neurons are obtained by the weighted sum the of $L-1$ th layer hidden neurons or input neurons if $L = 2$. For instance, the results of i th second layer hidden neurons are derived by i th input in the following equations:

$$h_{11} = g(x_{i1} * w_{11} + x_{i2} * w_{21}) \tag{2.2}$$

$$h_{12} = g(x_{i1} * w_{12} + x_{i2} * w_{22}) \tag{2.3}$$

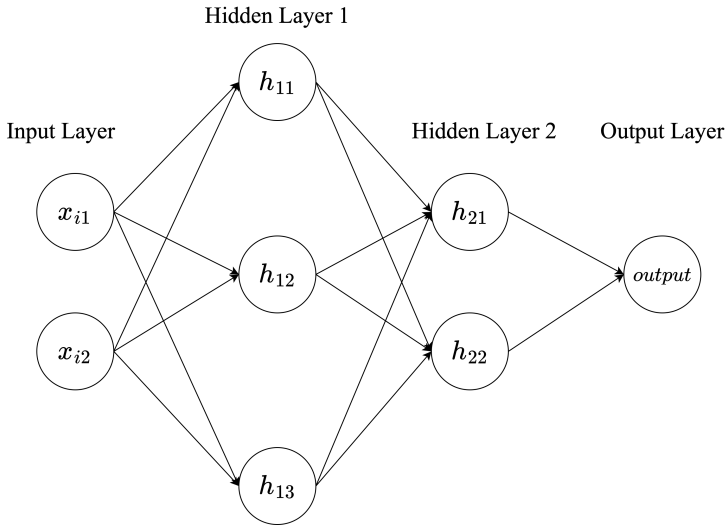

Figure 2.8: An example of 2 layer multi-layer perceptron. Inputs are fed into model, forward propagating through two hidden layers of neurons and obtained as output.

$$h_{13} = g(x_{i1} * w_{13} + x_{i2} * w_{23}) \tag{2.4}$$

where $g(\cdot)$ denotes the activation function. Here the bias term is eliminated as it is represented in x_{i2}. $x_{i:}$ represents the i th instances in the data set. $\mathbf{w} \in \mathbb{R}^{2\times 3}$ represents the weight matrix of the first hidden layer, where w_{ij} represents the weight of i th input neuron to the j th hidden neuron. This process is called "forward propagation". The same process is happening on the next hidden layer with a new weight matrix.

Furthermore, an "activation" function is applied after hidden-layer computation. The role of the activation function is to introduce non-linearity to the neural networks. If there is no activation layer in the neural network, then the whole forward propagation process is equivalent to the one-layer linear transformation. Thus, it is necessary to introduce an activation function in the forward propagation process to fit non-linear data. The most commonly used activation functions are "Sigmoid" and "ReLU (Nair & Hinton 2010)". Sigmoid function is usually utilized to map the continuous value to the range $(0, 1)$, which can be naturally interpreted as probabilities. The sigmoid function is defined as:

$$S(x) = \frac{1}{1 + e^{-x}} \tag{2.5}$$

Given the function of sigmoid, it is suitable for classification tasks; i.e. neural networks with sigmoid function at the final hidden layer will predict the probabilities of classes. In the inference

stage for binary tasks, predictions can be made simply by choosing the class with the largest logit (sigmoid output). For multiclass tasks, softmax layer is used to transfer the continuous value of each output neuron to unified probabilities. The softmax function is defined as:

$$\text{softmax}(x)_i = \frac{e^{x_i}}{\sum_j^n x_j} \tag{2.6}$$

For the other tasks like regression tasks, "ReLU" has been proved useful to introduce non-linearity in hidden layers. The ReLU function is defined as

$$f(x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{2.7}$$

2.2.3 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) is designed to process grid-like data introduced by LeCun & Bengio 1998. Distributions of image data, represented as distribution of pixels are suitable to use CNNs to model. CNNs use filters as feature maps to extract and map features from raw and grid-like data to the hidden dimensional space. The usage of shared feature maps extremely reduces the parameter size (known as sparse connectivity) compared to using multi-layer perceptrons to process image data, which assumes each of the pixels is connected. Further, by applying feature maps to the whole image, the pixels are sharing the same set of weights in the feature map, which enables the extraction of common features for pixels, thus enhancing the models' ability of generalization. The following figure 2.9 depicts the forward process of an example convolutional neural network.

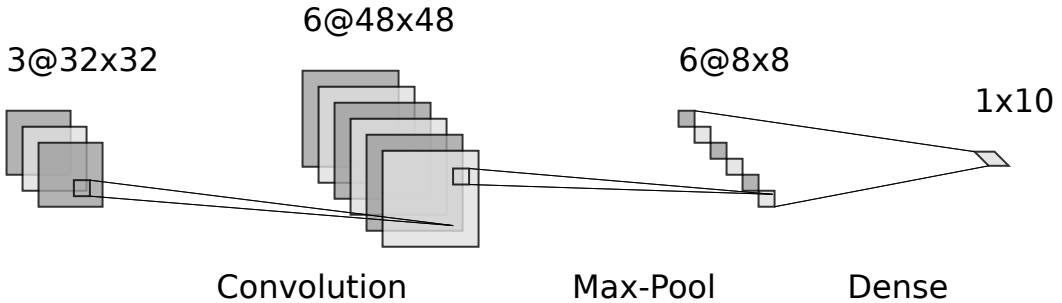

Figure 2.9: An example of CNN model. A three-channel image instance is first fed to a convolutional layer, upsampling to six-channel data with 48×48 pixels, then max pooling layer is used to downsample extracted features and use a dense layer to downsampe to 10-classes probabilities.

As figure 2.9 shows, in the convolutional layer, filters are used for different channels of the

input image instance to up-sample the features in raw images. In each channel, the whole pixels are shared with one set of parameters in the filter. Then, the max pooling layer also uses filters to filter the data with maximum value in specific areas. This down-sampling process can be viewed as feature selection. Finally, a one-layer perceptron is used to convert the dimension of output to 10 to facilitate a 10-class prediction task. The detailed one-step convolution process can be seen in figure 2.10.

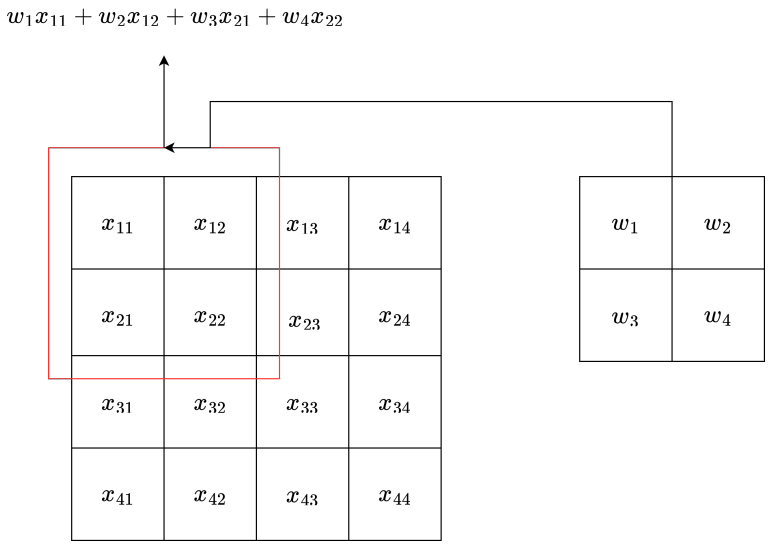

Figure 2.10: An example of one-step convolution with 2×2 kernel on 4×4 image.

2.2.4 Gradient-Based Optimization

Neural networks need to be trained to have a good performance on the test set, i.e. good ability to generalize on true data distribution. The training process includes forward propagation and backward propagation (Rumelhart et al. 1986). As mentioned before, forward propagation refers to the process of flowing input throughout the networks, while backward propagation refers to the parameter updating process. The parameters are updated in the backward direction, i.e. the parameters inside the last hidden layer will be updated first in a updating round, and then the parameters of the previously hidden layer are updated. The update round will end with updates to the parameters of the first hidden layer.

The optimized object function is also known as the loss function, which depicts the difference between the predicted value (distribution) and the true label value (distribution). Thus, by optimizing the loss function, the fitted neural networks are able to fit the given data. The gradient-based optimization method first calculates the gradient of loss w.r.t. the selected parameter, then the selected parameter will be updated as following the opposite direction of the gradient vector of loss, and thus the loss will decrease at the fastest speed with this update. Gradient descent, stochastic gradient descent, and batch gradient descent all use gradient to optimize the objective function. The following paragraphs will briefly introduce each of them.

Gradient Descent Gradient descent uses all data instances from the data set to calculate the gradient and update the selected parameter. Given a train set $T = \{(x_1, y_1), (x_2, y_2), ...(x_n, y_n)\}$, the process of updating one specific parameter w_i is shown as:

$$w_i^{t+1} = w_i^t - \alpha \nabla_{w_i^t} L(T; \boldsymbol{w}) \tag{2.8}$$

where w_i^{t+1} denotes the updated parameter; w_i^t denotes the parameter that needs to be updated; α denotes the learning rate of updating the parameter, i.e. how fast the parameter is changing to make loss function lower value; $\boldsymbol{w}$ represents the whole parameters in the neural networks. It can be seen that $\nabla_{w_i} L(T; \boldsymbol{w})$ represents the gradient, i.e. the direction that will enhance the loss function with the greatest speed, w.r.t. the selected parameter. Scaling this direction with $-\alpha$ will decrease the loss with the greatest speed.

It is natural to calculate the gradient in this way, however, this will usually lead to compu-

tational problems. Given that neural networks usually need to be fed large data sets in order to satisfy the sample complexity, calculating gradients for all data instances is computationally expensive.

Stochastic Gradient Descent To solve the computational problem that existed in gradient descent, stochastic gradient descent only uses one data instance to update the selected parameter in the time step. The process can be viewed as:

$$w_i^{t+1} = w_i^t - \alpha \nabla_{w_i^t} L(T_j; \boldsymbol{w}) \tag{2.9}$$

where T_j represents the j-th data instances in the training set. In this way, parameters will be updated at a very high speed, i.e. one instance will help to update all parameters once. However, there are also potential problems that existed with this method. First, one data instance cannot represent the whole distribution of the data set; therefore, using one instance only to update the parameter will introduce much noise, which will cause fluctuation of loss value in the optimization process. Second, due to the noise brought by each one of the instances, the optimization process is easy to fall into local minima, given that most problems are not convex in the scenario of using neural networks. Therefore, the parameters may not be updated in several epochs.

Mini-Batch Gradient Descent To overcome the mentioned problems, batch gradient descent is proposed to be applied for optimization in neural networks. In each time step, a minibatch of data instances is selected to update the parameters. The process can be viewed as:

$$w_i^{t+1} = w_i^t - \alpha \nabla w_i^t L(T_{[M]}, \boldsymbol{w}) \tag{2.10}$$

where $T_{[M]}$ denotes the M samples selected from training set T. In this scenario, each batched data is used to update the parameters with high speed and is less likely to fall into local minima.

2.2.5 Residual Connection

In the study of convolutional neural networks, it has been found that the stacking of multiple layers of neural networks degrades the performance of the network. Sandler et al. 2018 pointed out that the reason why the performance of a neural network increases with depth is that the network loses some of its information irreversibly at the activation layer, resulting in a gradual

loss of the ability of constant mapping input back to the input. He et al. 2015 proposed a structure called "residual connection", which cleverly allows a neural network to fit the residuals between the target and the input. This allows the neural network to have an intrinsic ability to constantly map input to back to input no matter how deep it is. The structure can be seen in Figure 2.11.

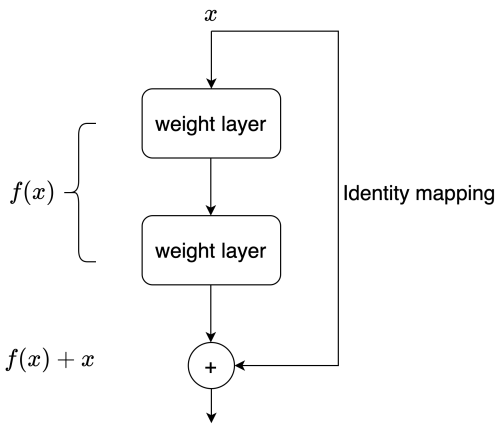

Figure 2.11: Structure of residual connection. 2 weight layers aimed to fit the "residual" between input x and result $f(x) + x$.

This structure has been proved effective not only in CNNs but also in the field of natural language processing (Vaswani et al., 2017) and graph neural networks (Dwivedi & Bresson, 2020).

2.3 Graph Neural Networks

Recent years have witnessed great success of neural networks in euclidean data. It is convenient to use multilayer perceptrons to process continuous 1-D data, 2-D images can be processed with CNNs with great performance. However, in daily lives, a large number of data is in the form of graphs, which is non-euclidean data, such as social networks, protein molecules, ligand molecules and so on. It is desirable for neural networks to be able to explicitly process graph data. Thus, graph neural networks are designed to process graph data elegantly. In general, there are three types of graph neural networks, Recurrent Graph Neural Networks (Wei et al., 2020), Spatial Convolutional Network (Kipf & Welling, 2016, Veličković et al., 2017) and Graph Spectral Convolution Networks (S. Zhang et al., 2019). To introduce graph neural networks, we first introduce the basic concepts of graph data. Then, gentle introductions for graph convolutional networks and graph attention networks are presented.

2.3.1 Graph data

In general, graph data is represented as $G = (V, \mathcal{E})$, where V represents node set $v_i \in V$ and $\mathcal{E}$ represents edge set $e_i \in \mathcal{E}$, with v_i and e_i represent single node and single edge, respectively. Given that matrix manipulation is preferred in neural networks for fast and parallel computation, it is preferable to represent the connectivity of the G using adjacency matrix A. The shape of the adjacency matrix is $\mathbb{R}^{|V|\times|V|}$. The element A_{ij} is 1 when the i th node is connecting to the j th node and 0 otherwise. For instance, for the straight-forward example graph in figure 2.12, The

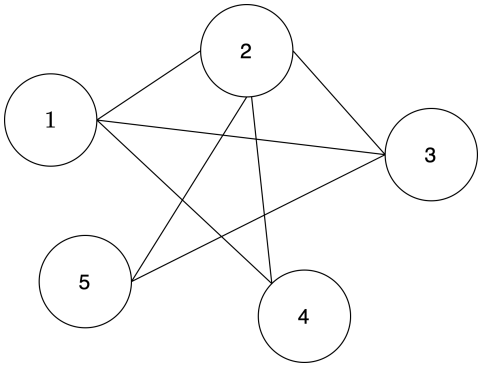

Figure 2.12: An example graph with 5 nodes. Note that each edge is bi-directional.

adjacency matrix can be represented as matrix 2.11:

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix} \tag{2.11}$$

The degree matrix is a diagonal matrix that is used to describe the information about the degree of each node, that is, the number of edges attached to each node. The degree matrix D can be represented as:

$$D_{ii} = \sum_{j=1}^{n} A_{ij} \tag{2.12}$$

In example 2.12, the degree matrix is represented as:

$$\begin{bmatrix} 3 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix} \tag{2.13}$$

2.3.2 Graph Convolutional Networks

Previously, in the introduction of CNNs, it is stated that in each convolutional layer, a shared filter is used to extract features from the image data. In graph convolutional networks, parameters are also shared; however, the parameters are shared with their topological neighbors, instead of the pixels in the fixed sliding window in CNNs.

A simple example can be considered before entering graph convolutional networks for a better understanding of the parameter sharing scheme. First, all nodes V in a graph G are represented as a feature matrix $X \in \mathbb{R}^{N \times D}$, where N represents the number of nodes within the graph G, D represents the hidden number of features for each node v_i. The graph neural network is trying to model a function f that takes the input matrix X and generates the output matrix $Z \in \mathbb{R}^{N \times F}$, where F represents the number of features in the output of each node. Thus, the parameters inside graph neural networks in trying to transform and extract the features of the nodes using

the topological structures of the graph G. Each of the network layers can be represented as

$$H_{l+1} = f(H_l, A, \boldsymbol{w}) \tag{2.14}$$

where H_l denotes the hidden representation of nodes, when $l = 1$, H_l is equivalent to X. w represents the parameters within neural networks.

If the function $f(\cdot)$ is represented as:

$$f(H_l, A, \boldsymbol{w}) = \sigma(AH_l\boldsymbol{w}) \tag{2.15}$$

where A represents the adjacency matrix of graph G, σ represents the activation function. It is trivial to find out that the function f is adding neighboring nodes' representations together and using w as a "feature extractor" to transform or extract the aggregated representation in the other algebraic space.

However, simply adding the neighbouring nodes' representations will change the scales of aggregated representations, which is not desirable for neural networks. For instance, if node 2 in graph G has 100 neighbours, while node 1 only has 1 neighbour, then the scale of aggregated node 1's representation may be totally different to node 2's representations, however, both of them are sharing weights $\boldsymbol{w}$ to transform in this scenario. Furthermore, since A does not contain any of the connections on its diagonal, central node representation is not considered when aggregating node representation. In order to encounter the potential problems mentioned above, Kipf & Welling 2016 proposed graph convolutional networks that change $f(\cdot)$ to:

$$f(H_l, A, \boldsymbol{w}) = \sigma(D^{-\frac{1}{2}}\hat{A}D^{-\frac{1}{2}}H_l\boldsymbol{w}) \tag{2.16}$$

where $\hat{A}$ represents $A+I$ and I is the identity matrix. The proposed graph convolutional networks use the degree matrix D to normalize the aggregated representations of each node and add the identity matrix to the adjacency matrix to manually add self-loops in the graph.

Message Passing Scheme

Following equation 2.16, it can be seen that in each layer, neighboring nodes x_j with $j \in \mathcal{N}_i$ are passing messages to the central node x_i ($\mathcal{N}_i$ denotes the neighbors of node i). In the first layer, the central node x_i will combine the information of the first neighbors (directly connected neighbors).

These first neighbours are called 1-hop neighbours to central node x_i. When it is proceeded to the second layer, all neighbouring nodes x_j will receive the information from their first neighbours, while the central nodes x_i will now have the information from the second neighbour, note that these neighbours are not necessarily directly connected to the central node x_i. These neighbours are called 2-hop neighbours to central node x_i. In this scenario, it is trivial to understand that a deeper model, i.e. more layers need to be applied for central node x_i to receive information from distant neighbours. A toy example is displayed as 2.3.2.

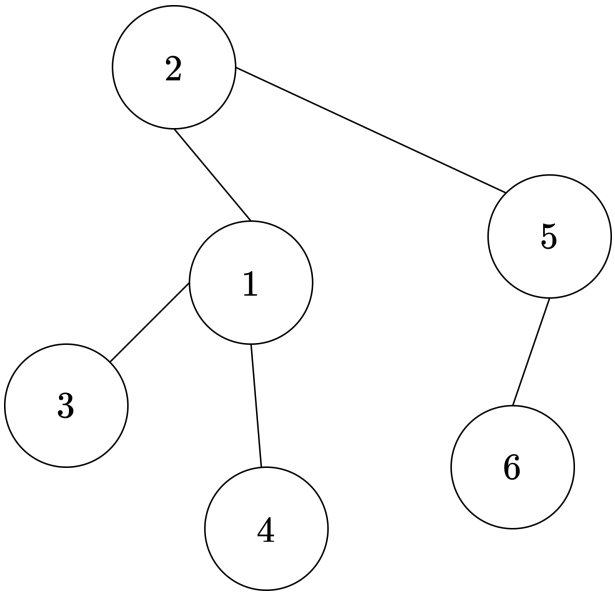

Figure 2.13: In this example, node 1 is defined as the central node for easy explanation. In this first layer of message passing, the information from node 2, node 3 and node 4 can be directly passed to node 1, i.e. they are explicitly connected. The information from node 5 will be passed to node 1 in the second round of message passing through node 2. Therefore node 5 is called 2-hop neighbor to node 1. For node 6, the information will be passed to node 1 in the third layers. In this case, node 6 is called the 3-hop neighbor to node 1.

2.3.3 Graph Attention Networks

In graph convolutional networks, aggregated node representations are obtained by summing all the neighbouring node representations and normalized with the number of neighbouring nodes. However, in this scenario, neighbouring nodes are treated equally, i.e. all neighbours have the same weights. However, in real scenarios, neighbouring nodes are not necessarily equally important to the central nodes. Therefore, Veličković et al. 2017 proposed to add an attention mechanism in the stage of aggregating neighbour representations to obtain central node representations.

The input of graph attention networks is represented as the nodes' input representation matrix

$X \in \mathbb{R}^{N \times D}$, where N represents the number of nodes within the graph G, D represents the hidden number of features for each node v_i. The output of graph attention networks is a transformed nodes' representation matrix $Z \in \mathbb{R}^{N \times F}$, where F is potentially different from D. First, the input matrix X is transformed by a shared weight matrix $W \in \mathbb{R}^{D \times F}$. Then, the self-attention mechanism a following Bahdanau et al. 2014 is applied between the central node representation and neighbouring node representations,

$$e_{ij} = a(h_i, h_j) \tag{2.17}$$

where h_i represents the central node representations, and h_j represents one of the neighbouring node representations. e_{ij} is calculated as the "importance score" of the neighbouring node to the central node. The attention mechanism is represented as:

$$a(h_i, h_j) = LeakyReLU(\boldsymbol{w_a}[h_i||h_j]) \tag{2.18}$$

where $\boldsymbol{w_a}$ represents the weight vector $\boldsymbol{w_a} \in \mathbb{R}^{2F \times 1}$, $||$ represents the concatenation operation, $LeakyReLU$ represents the activation function mentioned in Xu et al. 2015. Furthermore, the final attention scores are obtained by applying softmax to all scores of neighbours.

$$\alpha_{ij} = \frac{\exp^{e_{ij}}}{\sum_{j \in \mathcal{N}_i} \exp^{e_{ij}}} \tag{2.19}$$

Finally, the updated central node representaion can be viewed as:

$$h_i^{l+1} = \sum_{j \in \mathcal{N}_i} \alpha_{ij} h_i^l \tag{2.20}$$

In order to obtain stable central representations, the authors find it helpful to use multi-head attention to obtain central node representations. In this scenario, equation 2.17 and 2.20 are performed in k sub-spaces, where k refers to the number of heads in multi-head attention. Thus, the K embeddings $h_{i,k}^{l+1}$ will be obtained after k-head attentions. The updated central node representations can be viewed as

$$\hat{h_i^{l+1}} = ||_{k=1}^{K} h_{i,k}^{l+1} \tag{2.21}$$

For the final output layer, the central node representations will be averaged to one central node

representation,

$$\hat{h_i^{l+1}} = \sigma(\frac{1}{K}\sum_{k=1}^{K} h_i^{l+1}) \tag{2.22}$$

where σ represents the activation function.

Chapter 3

Related Work

This chapter will present the related work on the use of graph neural networks at the atomic level and the residue level on biological and chemical tasks and the related work on the prediction of the binding affinity task.

3.1 Graph Neural Networks on chemical and biological tasks

Given that molecules are naturally represented as graph data in chemistry and biology, using graph neural networks to tackle molecular modelling, as well as molecular interaction prediction, has been an important research area in recent years. First, Gilmer et al. 2017 proposes to use a message-passing structure to model quantum chemicals that are naturally existed in graph data. The authors use the proposed structure to predict the chemical's properties. They model molecules using all of the atoms represented as nodes and use covalent bonds as edges. Jørgensen et al. 2018 proposes to update the structure with an edge-updating mechanism, the information flowing between atoms is dependent on the receiving atoms, and, in turn, edge representations also depend on the connected atoms. The authors suggested using a k-nearest-neighbour-based method to construct graphs of chemicals, i.e. atoms will connect with the k-nearest neighbouring atoms, regardless of the layout of covalent bonds. Xie & Grossman 2018 proposes to use Crystal Graph Convolutional Neural Networks (CGCNN) to predict the properties of molecules in crystals. Pathak et al. 2021 uses graph neural networks to model the atomic interaction between solvent and solute. The solvent molecules and solute molecules are fed into different graph neural networks, and then MLPs are used to predict the interaction using the output from two separate graph neural networks. J. Wang et al. 2020 uses a graph convolutional network-based method to predict

atomic charges. Choudhary & DeCost 2021 proposes to contain the information of angles in each of the covalent bonds for the prediction of materials' properties. All of the above approaches demonstrate the important impact and outstanding contribution of graph neural networks in molecular modelling and molecular interactions in the fields of chemistry and biology.

3.2 Binding Affinity Prediction

Binding affinity prediction is an active research area in recent years (Thafar et al., 2019). Binding affinity prediction mainly focuses on predicting binding affinities between ligands and receptors. Nguyen et al. 2020 focuses on the prediction of the binding affinity between the drug and the target.The authors only represent drugs as graph data at the atomic level, while they continue to represent the target as 1D sequences at the amino-acids level. Although interaction within the drug-target complex is not provided, the performance of the model still grows by leveraging GNNs with graph input of the drug compared to models that are fed with only sequences, which indicates that atomic-level graph inputs with GNNs provide better performance. Lim et al. 2019a uses both 3D graph data of target and drug to predict drug-target interactions. Furthermore, atoms in the molecules of the target that are far away than 8 Angstroms from atoms in the drug are discarded when constructing graphs. Moreover, inter-molecule information is also considered by connecting atoms from different types of chemicals that are within the range of 5 Angstroms. S. Li, Zhou, Xu, Huang, et al. 2021 further provides the idea of self-supervised interaction learning by using neural networks to reconstruct the interaction matrix, forcing the network to learn the interaction for specific instances, thus leading to better quality data representation. However, the authors mentioned above fail to include the original information from amino acids explicitly. These models are either complicated when modeling the interactions between nodes and edges or fail to consider the interaction between molecules.

Chapter 4

Methods

In this chapter, the data and model framework used for binding affinity prediction are introduced in detail. Data selection, data pre-processing, and complex graph construction are illustrated in order. And the model framework is described in terms of overview and details.

4.1 Data

4.1.1 Data set selection

In this dissertation, PDBbind data set is used as our data to carry out experiments. PDBbind data set is a collection of experimentally measured binding affinity data with complexes obtained from the Protein Data Bank (Berman et al. 2000). PDBbind contains files for protein and ligand molecules separately. The protein and ligand files share the same coordinate system, so that they can be explicitly combined to form protein-ligand complexes. It also contains the binding affinities of the protein-ligand complexes provided in the forms of $-\log(K_d)$ or $-\log(K_i)$. These denote how strongly a ligand binds to a protein as described in 2.1.4. PDB code, data resolution, the release year of protein-ligand complexes and ligand names are also provided. PDBbind data set contains general set, refined set and core set. The general set contains examined protein-ligand complexes with experimentally measured binding affinities. While the refined set contains a subset of the general set, which is selected as a higher-quality data set. Selection is based on the quality of the binding data, crystal structures and the nature of the complexes within the data set ("The PDBbind database: methodologies and updates." 2005). The refined set in this release consists of 5,316 protein-ligand complexes. The core set contains high-quality data that is served as the

primary test set in the popular Comparative Assessment of Scoring Functions (CASF) benchmark (Su et al., 2018). In this dissertation, the refined set is used to split as a training and development set in order to provide neural networks with better quality data. The models are evaluated on the core set in order to maintain the same benchmark as other research work (Son & Kim 2021, S. Li, Zhou, Xu, Huang, et al. 2021, Y. Li et al. 2019, Zheng et al. 2019). PDBbind v2016 is chosen to maintain the same version of the latest CASF data set (core set) and the same version of other research (S. Li, Zhou, Xu, Huang, et al. 2021, Zheng et al. 2019, Son & Kim 2021 Y. Li et al. 2019).

4.1.2 Data Preprocessing

Given that the PDBbind data set provides separated files for protein and ligand, it is necessary to combine protein and ligand into one complex for further graph extraction.

Note that proteins are provided in the form of two files and ligands are also provided in the form of two files. One protein file contains all atoms from protein molecules, while the other only contains atoms from protein molecules that are within the 10 Angstrom range from each ligand atom in the experimentally defined protein-ligand binding pose. Note that these protein files are all in the format of "pdb", so they can be easily read by biopython (Cock et al. 2009). In order to be consistent with the way of processing protein files, openbabel (O'Boyle et al. 2011) command lines are used to convert ligand files to pdb files that can be read by biopython. Ligand files contain the same content, i.e. all atoms of the ligand, in different file formats. Following Lim et al. 2019b and S. Li, Zhou, Xu, Huang, et al. 2021, only heavy atoms are extracted from protein and ligand to process. Further, since proteins and ligands are preserved in the solvent, water residues are contained in PDB files provided by the PDBbind data set. In this dissertation, water residues are removed from protein-ligand pairs to reduce the potential noise source. Finally, biopython is used to combine protein pdb files and ligand pdb files into complex pdb files. The cleaned data set contains 4057 pairs of protein-ligand complexes in the refined data set. Note that not all atoms are extracted from protein to form protein graphs. Huge graphs in GNNs are not preferred for two reasons. 1. Given that molecules are composed of atoms, large molecules such as proteins contain thousands of atoms. The largest protein known in humans is Titin, which has 38138 residues according to Krüger & Linke 2011. Even the smallest known protein (TRP-cage) has 154 atoms that form 20 amino acids. If each atom is selected to form the node in graph data, then each atom

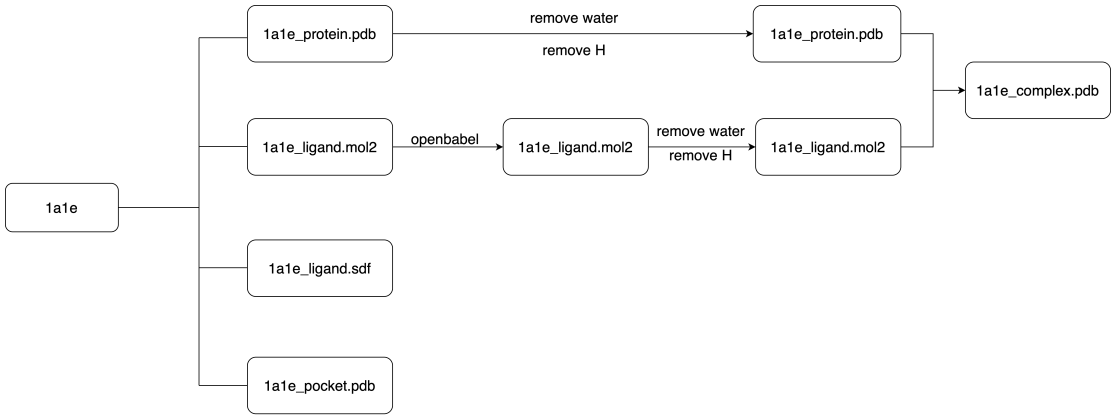

Figure 4.1: Every folder contains four files, two files for protein and two files for ligand. For protein, one file is in the format of pdb, which contains full protein information, i.e. all atoms from the protein is contained in this file. The other files is named "pocket", which means that this file only contains atoms from protein that are within 10 Angstrom range from each of the ligand atoms. Ligand files are provided in the formats of "sdf" and "mol2". There are no explicit different in the content of these two files. ("1a1e" represents the name of complex.)

will be regarded as a node, and an edge will be connected if there is a covalent between two atoms. Under this circumstance, consider a graph convolutional network with L layers with D dimensions trained on such graph G, the memory complexity of storing activations is $O(LND)$, where N is the number of nodes within graph G. This would cause memory problems for GNNs as indicated in G. Li et al. 2021. Thus, most of the research uses only the binding site of the protein atoms to model the protein-ligand complexes. 2. As the size of the graph increases, more layers should be used for the GNNs to obtain knowledge from the k-hop nodes according to 2.3.2. Although GNNs with many layers will introduce the "over smoothing" problem as mentioned in Cai & Wang 2020, which will cause degradation of network performance. Thus, in order to encounter these potential problems, we follow the previous works (S. Li, Zhou, Xu, Huang, et al. 2021, Nguyen et al. 2020) by only selecting atoms from proteins that are within a pre-defined filter threshold "th_f" from any of the atoms that are from the ligand, i.e. if $d(atom_p, atom_l) < th_f$, then the protein atom is preserved, where $atom_p$ represents protein atom, $atom_l$ represents ligand atom and $d(\cdot)$ represents distance function. In this dissertation, the Euclidean distance is used as a distance function.

4.1.3 Graph Extraction

Protein-ligand complexes in the form of PDB files need to be converted to graph data after data pre-processing. In general a graph G can be represented as $G = (V, \mathcal{E})$, where V represents node set $v_i \in V$ and $\mathcal{E}$ represents edge set $e_i \in \mathcal{E}$, with v_i and e_i represent single node and single edge, respectively. Given the natural data form of molecules, atoms or backbone atoms from amino acids

can be set to be nodes in graphs and covalent bonds between atoms to edge. In general, graph extraction is completed in three steps, protein graph construction, ligand graph construction, and interactive edge construction. These steps are illustrated in the following content.

Protein graphs construction

In order to view clearly how model performance will change by moving amino acid-level graphs to full atomic-level graphs. Protein graphs are constructed differently in three stages. In stage 1, only C_α is extracted from each of the amino acids in the protein to represent the nodes in the protein. As stated in Mukherjee et al. 2005, backbone atoms from protein, i.e., carbon alpha, carbon beta, nitrogen, oxygen and carbon atoms, can play a significant role in protein folding and interaction with other proteins, and molecules, therefore, in stage two, backbone atoms are extracted from each of the amino acids to represent nodes in graphs. Most importantly, while amino acids can have varying numbers of atoms, all amino acids will have these 5 atoms in common (except for glycine). In stage three, full atomic graphs are used to represent proteins. Note that ligands are represented as full atomic graphs in all three stages. The procedures to build graphs for proteins in three stages are illustrated in the followings:

Stage 1 Given that only C_α is extracted from each of the amino acids in the protein, the natural covalent bonds between the amino acids are then neglected. Thus, edges between carbon alphas of neighbouring amino acids are added manually to facilitate the message-passing procedure. The constructed graph example can be viewed in Figure 4.2.

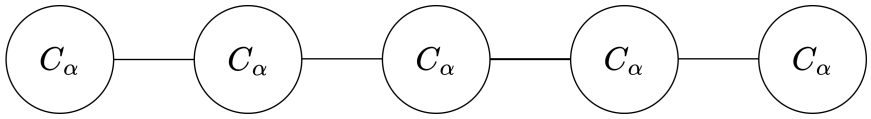

Figure 4.2: Edges are constructed manually between neighboring carbon alpha atoms.

Stage 2 In stage 2, the backbone atoms are extracted from each of the amino acids to represent the nodes in the protein. Therefore, natural covalent bonds can be preserved and represent edges in protein graphs. The edge connection in the backbone structure of the amino acid is fixed, as shown in figure 4.3.

Stage 3 In stage 3, protein graphs are represented as full atomic graphs, therefore, it is natural to represent nodes with protein atoms and edges as covalent bonds in protein structure.

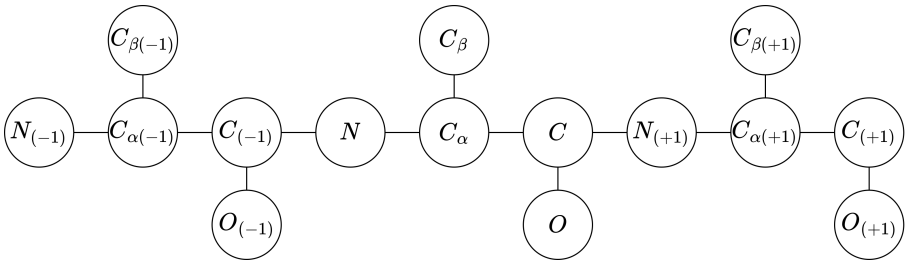

Figure 4.3: Covalent bonds inside backbone structure automatically form edges in protein graphs.

Ligand graphs construction

For ligand graphs, atoms and covalent bonds from ligand are used to represent nodes and edges in the ligand graphs, respectively.

Interactive edge construction

In order to model interactions between protein and ligand in-depth, edges are added between protein atoms and ligand atoms if protein atoms are within the pre-defined threshold "th_v". The intuition is that the closer the two atoms are, the more possible they will have interactions with each other. Further, adding edges between protein graphs and ligand graphs will facilitate the message passing in protein-ligand complex, thus will provide networks with more interactive information. However, not all of the atoms of the protein and ligand are connected together. Since GNNs is scaling with the number of edges in the graphs, if all atoms are connected to each other, the GNNs will result in being very difficult to train.

4.1.4 Feature Extraction

This dissertation uses atoms' biological features mentioned in Xiong et al. 2020. According to Allen et al. 1987, we manually decide the bond lengths of C-C, C-O, C-N, N-O, O-O and N-N, since C, N and O are the most frequent elements present in the protein. These bonds are treated as special bonds, since proteins are formed mostly with these three types of atoms; thus, the bonds between these atoms will influence the interaction with ligand molecules. Furthermore, three general classes are set according to the bond length. If the bond length of the incoming edge matches our decided bond lengths, then we assign the corresponding special bond type to the edge. If none of the bond lengths matches, we assign the edge with a bond class according

to the bond length range. The intuition behind these three general classes is that the distance between atoms will decide the interactions of two atoms that will be triggered according to Bé guin et al. 2013.

4.1.5 Input Graph and Features

Input complex graphs are obtained as discussed in 4.1.3. Atomic features are the aromaticity of the selected atom, number of hydrogen around the selected atom, number of bonds, i.e. degree of the selected atom, number of radical electrons around the atom and atomic number of the selected atom. Furthermore, categorical features of atoms and residues are used to obtain node embeddings. All atomic features are obtained using RDkit [1]. Therefore, seven atomic biological features are used to describe an atom. These categorical features are encoded using embedding layers for nodes and edges, respectively.

For atomic categorical features, 25 classes of atoms are extracted in the PDBbind data set in total. To classify the origin of the atom (protein, ligand), a shift is added to the atom class, so the total number of atomic classes is $25 \times 2 = 50$.

For residual categories, 20 fixed residue classes are added manually following Lopez & Mohiuddin 2021 and 1 additional type representing "unknown". The same shift is applied to residual classes, that is, so the total number of atomic classes is $(20 + 1) \times 2 = 42$.

Edges are divided into 13 categories. For edge categories, as discussed in 4.1.4, C-C, C-O, N-N, N-O, O-O, C-N six special bond types and three general edge classes are defined in total according to the atom classes and bond lengths, i.e., TYPE1, TYPE2, and TYPE3. Furthermore, for virtual connection, an additional edge type VIRTUAL is defined. Sometimes, the hyper-parameter th_v is set too small for any of the ligand atoms to connect any of the protein atoms, the 1 nearest protein atom to each of the ligand atoms is manually connected. Thus, a new edge type MANUAL is defined. For stage 1 experiments, protein graphs are extracted using carbon alpha from residues to represent graph nodes. So, there are no natural covalent bonds within the graph, we manually connect neighbour residue to form a chain in protein graphs. Therefore, we have the edge type RESIDUE. Last, we also have edge type UNKNOWN if none of the situations suits the specific

[1] `http://www.rdkit.org`

edge distance. In total, 13 edge types are defined for graphs in the data set.

4.2 Model Framework

Inspired by research on binding affinity prediction (S. Li, Zhou, Xu, Huang, et al. 2021, Son & Kim 2021, Lim et al. 2019b, Nguyen et al. 2020), graph neural networks are applied to predict the binding affinity of the protein-ligand complex. In this dissertation, a transformer-based graph is selected as the model to process complex graphs. Model structure is adopted from Shi et al. 2020. Unlike Graph Attention Networks (GATs) that only compute the attention score between central nodes and neighbouring nodes, transformer based graph neural networks (Dwivedi & Bresson 2020) can consider the interaction globally by using positional encoding. In the research of binding affinity prediction, even though ligand atoms are connected to the atoms from protein that are within the range th_v, the other atoms from protein can also be interactive with ligand atoms. Furthermore, the graph transformer also considers the edge attributes. Thus, transformer based graph neural networks are the optimal solution in this scenario.

4.2.1 Positional Encoding

In transformer model (Vaswani et al. 2017) designed for NLP tasks, sinusoidal positional encodings are applied to the input vectors to identify the position of each token in the sentences. In GNNs, positional encodings are also introduced to inject nodes' relative positional information into the model to derive better performance. In our model, the learned positional encoding is used to encode each of the node embeddings to use the information of the absolute node orders. The molecule embedding (M) is represented as

$$M = M + pos\_encoding[0 : M.length] \tag{4.1}$$

where $pos\_encoding$ represents the learned positional encoding parameters, $M.length$ represents the number of nodes in the selected molecule.

4.2.2 Multi-head Attention Block

A transformer-based graph neural network is composed of several attention blocks, which represent the core of the model. The flow of input and output can be viewed in figure 4.4. This figure represents the attention block in the l th layer. Note that positional encoding is only applied at the first layer, which means $l = 1$ here. Q^l, K^l and V^l are derived from the central node embedding x_i^l and neighbor embeddings x_j^l from subspaces just as Vaswani et al. 2017 did, where $Q^l, K^l, V^l \in \mathbb{R}^{d_k \times d}$,

, d represents the dimension of the node embedding, d_k represents the dimension of the subspace embedding. Additionally, edge embeddings have interacted in the computation of attention score. Attention scores are obtained by rescaling the results of matrix multiplication between Q^l with K^l and interaction with E^l. Weighted sum is applied to V^l using derived attention scores and apply a one-layer perceptron to the weighted sum of V^l to obtain our final update central node embedding x_i^{l+1}. The same transformation is applied to edge embeddings. The central node embedding and edge embedding update processes can be viewed in 4.2 and 4.3

$$x_i^{l+1} = O_x^l \parallel_k (\sum_{j \in \mathcal{N}_i} w_{ij}^l V^l x_j^l) e_{ij}^{l+1} = O_e^l \parallel_k (w_{ij}^l) \tag{4.2}$$

$$w_{ij}^l = softmax_j(\frac{Q^l h_i^l \cdot K^l h_j^l}{\sqrt{d_k}}) \cdot E^l e_{ij}^l \tag{4.3}$$

where $O_x^l, O_e^l \in \mathbb{R}^{d \times d}$.

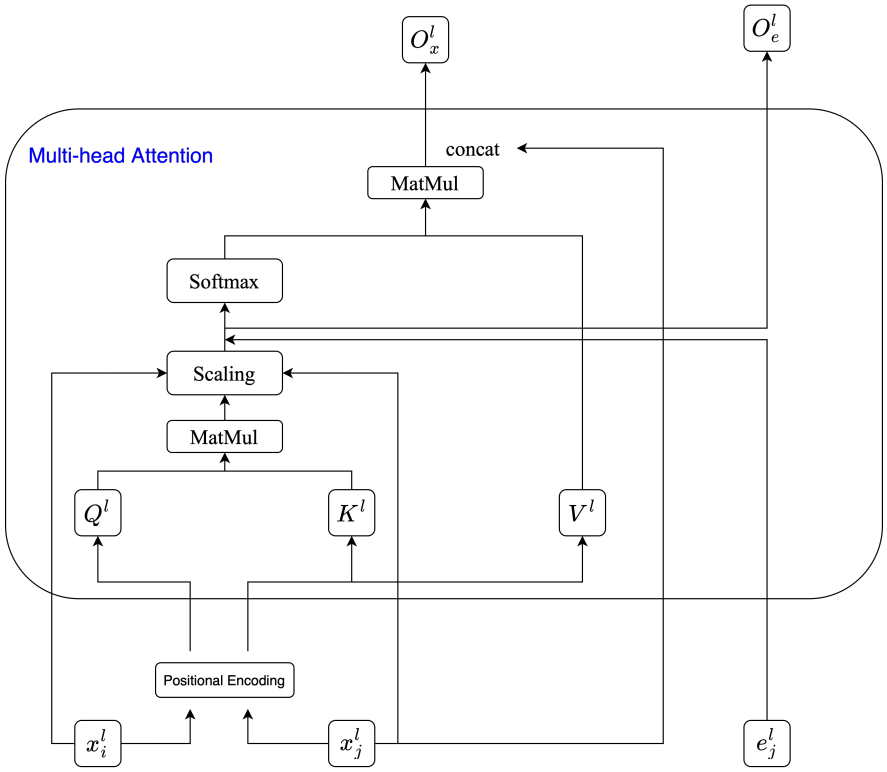

Figure 4.4: Attention block (at the first layer). x_i^l represents i-th central node embedding in layer l, x_j^l represents j-th neighbor of central node embedding x_i^l in layer l. After adding positional encoding, two embeddings are expanded to N heads to derive Q^l, K^l V^l in sub-space, and then attention scores are computed using Q^l and K^l, which will be multiplied with V_l and x_j^l. Next, weighted outputs from N sub-spaces will be concatenated together and transformed by O_x^l matrix. The same transformation is applied to edge embeddings.

4.2.3 Residual Connection and Feed Forward Network

After obtaining the normalized output, central node embeddings are first added by the input central embeddings (before multi-head attention) to form a residual connection, then fed into two feed-forward layers to obtain transformed embeddings. The process can be seen in 4.4.

$$FFN(x) = max(0, xW_1 + b)W_2 + b \tag{4.4}$$

These two-layer FFN will enlarge the dimension to 4 times, then convert back to embedding dimensions in order to extract features in higher-dimensional spaces.

4.2.4 Prediction Head

After several layers of multi-head attention blocks combined with feedforward networks, updated graph node embeddings are obtained. First, a readout layer is applied to "read" the content within the graph node embeddings to obtain a single vector, i.e. graph representation for each graph. Then, the graph representations will be fed into Multi-Layer Perceptron to convert the embedding size to 1, i.e. output the regression value of binding affinity. The whole structure of our model can be seen in Figure 4.5.

4.2.5 Parameter Setting

In our model, the absolute positional encoding layer is used to encode the orders of nodes in graphs. For stage 1's graph construction, th_f that is used for filtering protein atoms from the whole protein graph is set to $8\mathring{A}$, th_v that is used for connecting ligand atoms with protein atoms is set to $16\mathring{A}$. For stage 2 and stage 3, th_f is set to $6\mathring{A}$ and $8\mathring{A}$ for different experiments, th_v is set to $16\mathring{A}$. Since in stage 2 and stage 3, protein graphs have more atoms compared to stage 1, it is necessary to reduce th_f to reduce the graph size in order to avoid the potential problems described in 4.1.2. Dimensions of embeddings for node class features, node biological features, and edge class features are all set to 128. The number of heads in the multi-head attention layer is set to 4. The number of multi-head attention layers is set to 2. Feed-forward layers are used to raise the dimension to 4 times then drop back to the original dimensions. The readout function in the prediction head is set to max pooling. The number of the linear layer in MLPs in the prediction head is set to 2. All activation function in models are set to ReLU (Nair & Hinton, 2010). The batch size is set to 32 in all experiments. A learning rate equal to 1×10^{-5} is used to train our

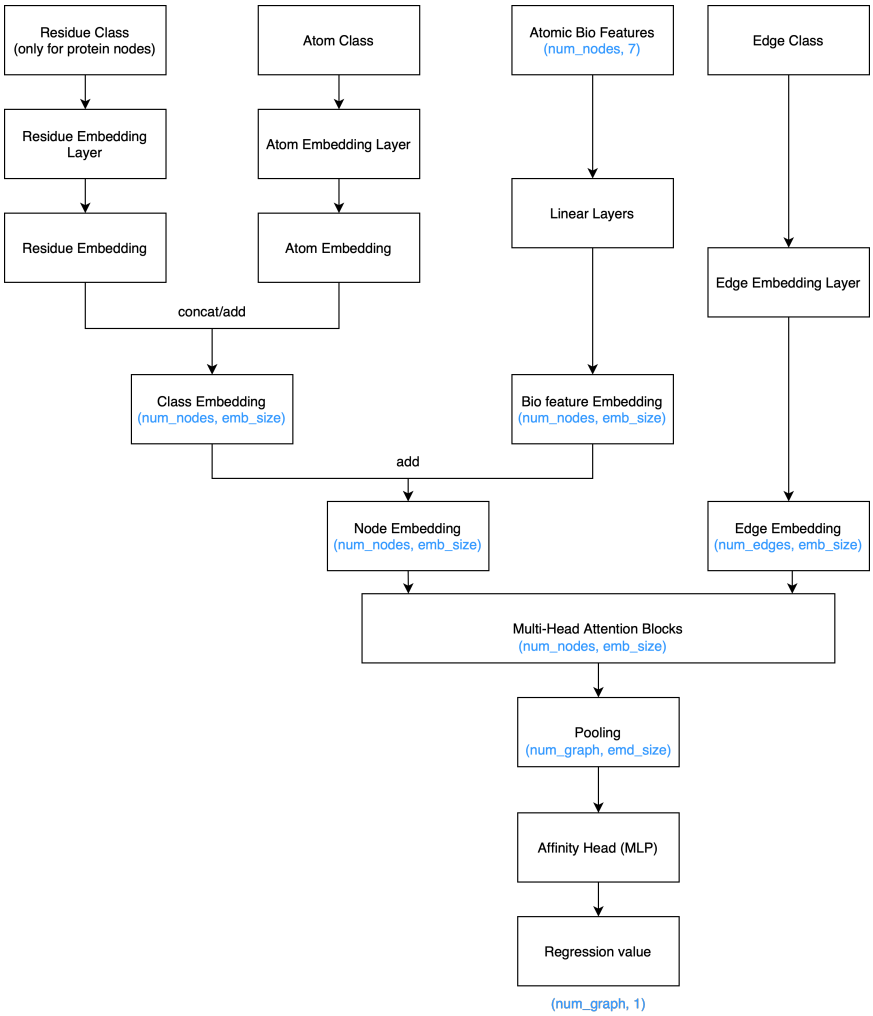

Figure 4.5: Overview of model structure. Class features and atoms' biological features are utilized to obtain node embeddings. Edge classes are used to obtain edge embeddings. Node embeddings and edge embeddings are fed into multi-head attention blocks. Readout (pooling) layer extracts graph representation. The graph representation is fed into MLPs to obtain the final predicted value.

models for 1500 epochs. The dropout rate is set to 0.2 in the training stage. $L1$ loss between targets and predictions forms our object function. Adam optimizer (Kingma & Ba, 2015) is used to optimize parameters in neural network models.

4.3 Evaluation Metric

In this dissertation, two metrics, root mean squared loss (RMSE) and mean absolute loss (MAE), are used to evaluate the performance of our models. RMSE is defined as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2} \tag{4.5}$$

MAE is defined as:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y_i}| \tag{4.6}$$

where n represents the total number of instances in the evaluated data set, $\hat{y_i}$ represents the predicted binding affinity and y_i represents the truth binding affinity of $i-th$ instance.

4.4 Baseline Model

In order to compare the structural information contained in amino acid-level graphs and full atomic-level graph. Our baseline model is chosen as the model that uses protein-side amino acid-level graphs as input. Thus, the baseline model only utilizes the amino acid-level structural information. The structural details of the model are completely the same as the models for stage 2 and stage 3 as mentioned in 4.1.3.

Chapter 5

Experiment

In this chapter, experiments for answering the following three research questions are described in details.

- **RQ1:** How does performance change when converting from residual level graphs to atomic level graphs?
- **RQ2:** Does atomic level GNNs have the more severe over-smoothing problem?
- **RQ3:** How to design a downstream model framework to fully utilize the atomic-level graph representations obtained?

5.1 Experiment Setting

5.1.1 Data set

All experiments are performed on the PDBbind data set as described in 4.1.1. The core set contains 285 complexes, and the rest of the data set contains 3772 complexes. The train-validation split ratio is 9:1. Thus, the models are trained on 3395 instances, validated on 377 instances and tested on 285 instances. Note that the test set is consistent with S. Li, Zhou, Xu, Huang, et al. 2021, Son & Kim 2021, Y. Li et al. 2019 and Zheng et al. 2019.

In addition, there is one research question for the PDBbind data set referred to by Yang et al. 2020, which is also investigated in this dissertation:

- **RQ4:** Does our model really learn the complexes in depth from PDBbind data set?

5.2 Experiment details

In this section, experiment details for each research question will be described.

5.2.1 Research Question 1: How does performance change when converting from residual level graphs to atomic level graphs?

To answer this research question, graph data is generated as mentioned in 4.1.3 for three stages, respectively. Then, the best hyper-parameter settings are searched in three stages in the parameter spaces in order to achieve the best performance in three scenarios.

5.2.2 Research Question 2: Does atomic level GNNs have the more severe oversmoothing problem?

To investigate the oversmoothing problem, the global cosine similarities of the three stages are calculated. The cosine similarity is calculated between two pairs of residue embedding in each stage. Residue embeddings are derived by the weighted sum of node embeddings in each time step in stages 2 and 3. For instance, in Stage 3, after propagating through the transformer, node embeddings for each batch that are from the same type of residue are added together with weights and derive them as the specific type of residual embedding. Global cosine similarity can be seen in equation 5.1.

$$score = \frac{1}{N}\sum_{i=1}^{N} score_i \tag{5.1}$$

where $score_i$ represents the averaged cosine similarity of residue embedding x_i to the other residue embedding x_j, N represents the total number of residue embeddings.

$$score_i = \frac{1}{P}\sum_{j=1}^{P} \frac{||x_i \cdot x_j||}{||x_i||||x_j||} \tag{5.2}$$

where P represents the number of pairs of residue embeddings. In this scenario, 42 residue embeddings are derived in total, and cosine similarities are obtained by each pair of residue embeddings. Global cosine similarity is derived by averaging the sum of all the similarities of residue embedding pairs.

5.2.3 Research Question 3: How to design down-stream model framework to fully utilize the obtained atomic-level graph representations?

This question is answered by adding context gating (Miech et al. 2017) layer in prediction head discussed in 4.2.4. Context gating layer is a one-layer MLP that outputs the re-weighted graph embeddings. This layer is represented as:

$$Y = X \circ \sigma(WX + b) \tag{5.3}$$

where σ represents the sigmoid function S.

$$S(x) = \frac{1}{1 + e^{-x}} \tag{5.4}$$

5.2.4 Research Question 4: Does our model really learn the complexes indepth from PDBbind data set?

To answer this research question on the PDBbind data set, two baseline models have been implemented. The first baseline model only uses protein information, i.e. only extracts protein graphs as input to the model. The second baseline model uses ligand information as input to the model.

Chapter 6

Results and Discussion

In this section, the results of the experiment will be elaborated in order of the research questions mentioned in 5.2.

6.1 Results

Note that all experiments have been implemented three times with different random seeds for stable results and conclusion.

6.1.1 Research Question 1: How does performance change when converting from residual level graphs to atomic level graphs?

By following the experiment setting discussed in 5.1, experiments are conducted for stage 1, stage 2 and stage 3. The results are shown in Table 6.1:

Stage	MAE (Mean ± Std)	RMSE (Mean ± Std)
1	1.036 ± 0.020	1.360 ± 0.013
2	1.081 ± 0.014	1.397 ± 0.017
3	1.106 ± 0.012	1.410 ± 0.005

Table 6.1: Results for three stages

It can be seen from the results that stage 1 outperforms stage 2, and stage 3 has the worst performance. It contradicts the previous hypothesis that an atomic level graph will bring more structural as well as biological information. We suspect that stage 2 and stage 3 introduce more

noise with more nodes than the information they provided. Thus the aggregation models are proposed to be implemented to reduce the noise, i.e. aggregate atoms back to residue when or after GNN's message passing. Embeddings of nodes that are from the same residues are added together with weights to form one residue node in the aggregation model. The visualization of aggregation model is shown in 6.1. These two models are named as the pure aggregation model

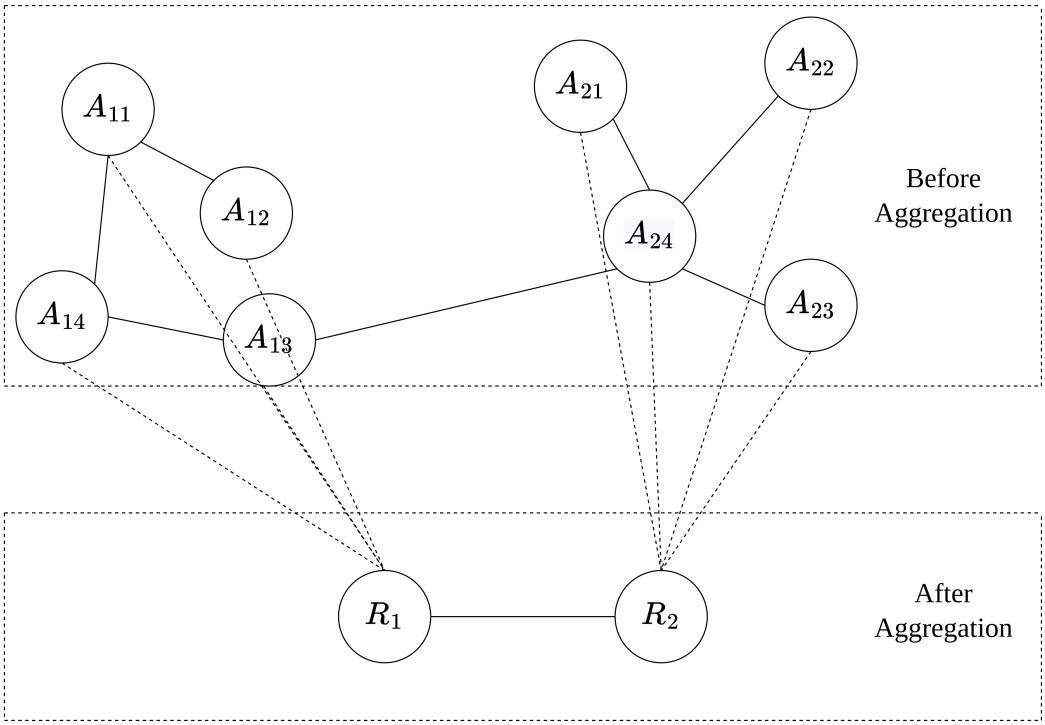

Figure 6.1: Atoms that are from the same residue in the origin graphs are added together via embeddings to form a new residue node. Edges within the same residue are eliminated. The edges between residues are preserved and assign with the new edge type RESIDUE as mentioned in 4.1.5.

and intermediate aggregation model. The visualization of these two aggregation models can be viewed in figure 6.2 and figure 6.3.

In this way, the number of nodes will decrease from atomic level to amino acid level. The intuition behind these two models is that the number of nodes will be reduced after transferring graph size to residue level. Thus, the noise brought by large-size graphs in stage 2 and stage 3 will be reduced as a result. The other method is to reduce the number of nodes in stage 2 and stage 3 in the graph extraction process, i.e. use small th_f to filter most of the protein atoms, thus obtaining small graphs in stage 2 and stage 3.

The following tables 6.2, 6.3 show the results of the aggregation models.

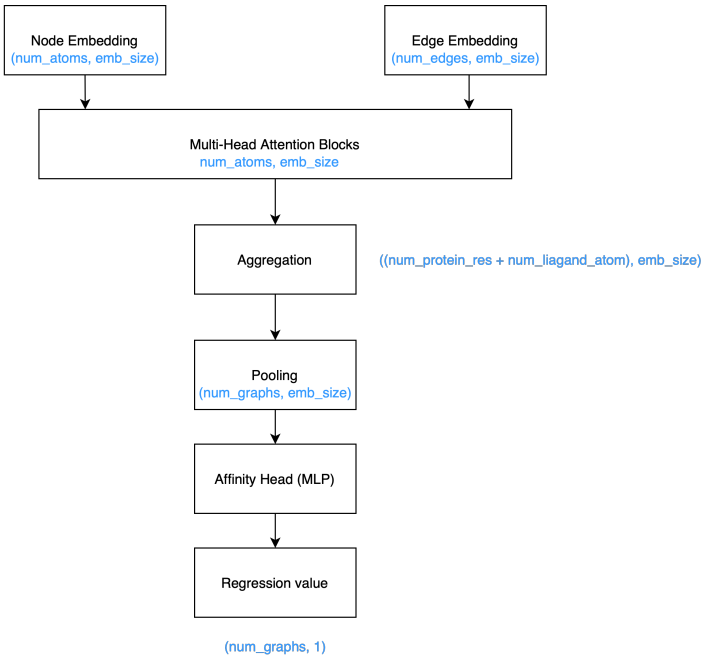

Figure 6.2: Pure aggregation model uses aggregation block before entering multi-head attention blocks.

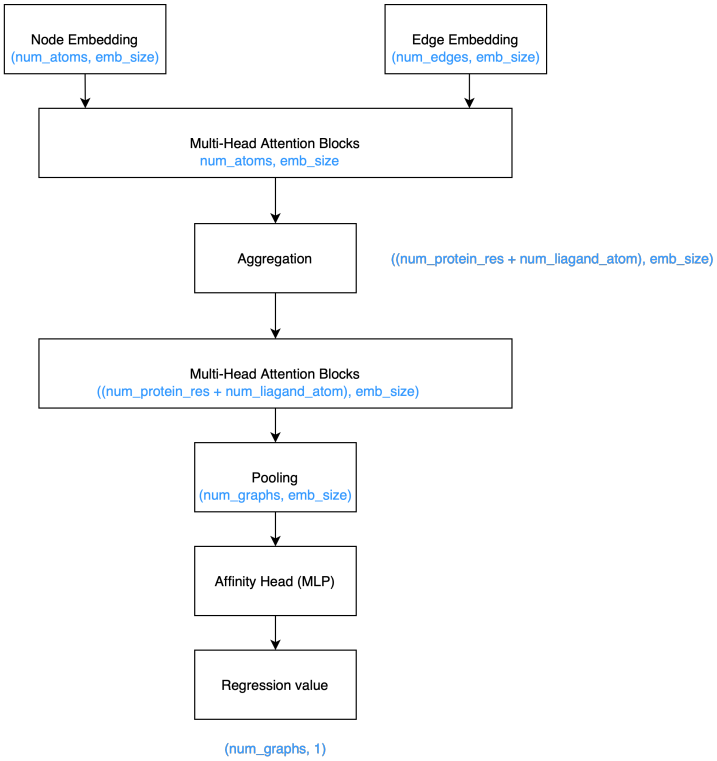

Figure 6.3: Intermediate aggregation model aggregates the nodes in the process of message passing in multi-head attention blocks.

Stage	MAE (Mean ± Std)	RMSE (Mean ± Std)
2	1.110 ± 0.034	1.396 ± 0.033
3	1.128 ± 0.005	1.429 ± 0.001

Table 6.2: Results for pure aggregation model

Stage	MAE (Mean ± Std)	RMSE (Mean ± Std)
2	1.119 ± 0.002	1.439 ± 0.007
3	1.097 ± 0.005	1.386 ± 0.018

Table 6.3: Results for intermediate aggregation model

It can be seen from the results that pure aggregation and intermediate aggregation models do not explicitly enhance the performance of models by aggregating nodes back to the amino acid level. The suspected reason is that the model still reads all the information (containing noise) from large graphs, and thus simply aggregating nodes back to the residue level does not have the effect of extracting effective information.

In addition, it is proposed to reduce the number of nodes in stage 2 and stage 3 to reduce the noise in the graph. The following table 6.4 shows the results of using $th_f = 6$ at stages 2 and 3.

Stage	MAE (Mean ± Std)	RMSE (Mean ± Std)
2	1.153 ± 0.027	1.478 ± 0.035
3	1.192 ± 0.015	1.496 ± 0.013

Table 6.4: Results for lower filtering threshold.

Tables 6.5, 6.6, show the number of nodes in each stage with different th_f.

Stage	Max Number of Nodes	Average Number of Nodes
1	124	55
2	384	165
3	555	253

Table 6.5: Number of nodes with $th_f = 8$

Stage	Max Number of Nodes	Average Number of Nodes
2	229	85
3	332	139

Table 6.6: Number of nodes with $th_f = 6$

Results show that, by reducing the number of nodes of stage 2 and stage 3, models perform even worse than previous. Thus, it can be confirmed from the results that models indeed use

information from large-size atomic-level input. As can be seen from the results, stage 1 still outperforms stage 2 and stage 3. Thus, it is necessary to suspect that stage 2 and stage 3 have the more severe over-smoothing problem than in stage 1, i.e., node embeddings are similar to each other in stage 2 and stage 3.

6.1.2 Research Question 2: Does atomic level GNNs have the more severe over-smoothing problem?

In order to quantitatively identify the over-smoothing problems in stages 1, 2, and 3, global cosine similarities of each residue embeddings are computed in each stage.

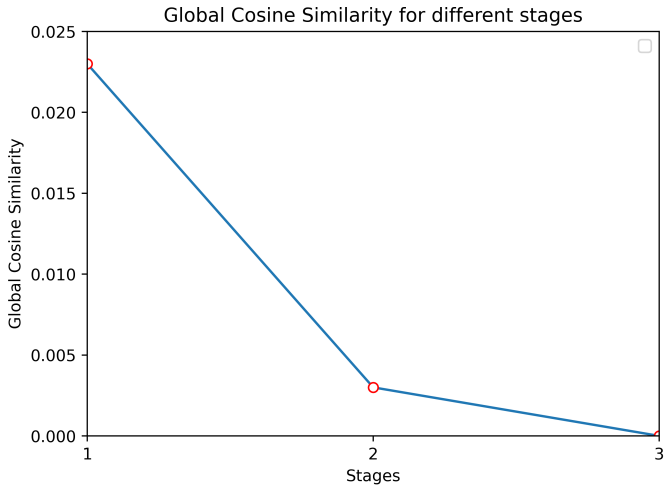

Figure 6.4: Cosine similarities for different stages. There is a descending trend in global cosine similarities of the three stages. In stage 3, cosine similarity decrease to 0.000, which means that almost all residue embeddings are orthogonal to each other in the embedding space.

It can be seen from figure 6.4 that, global cosine similarities are showing descending trend from stage 1 to stage 3, which means that stage 3 has the most high-quality residual embeddings. Note that the global cosine similarity of stage 3 is almost zero; in this case, residue embeddings in stage 3 are almost orthogonal to each other in the embedding space, which is desirable for GNNs. The intuition behind this is that, if more corelated embeddings are derived, GNNs are more likely to have an over-smoothing problem after k-hop message passing, i.e. the nodes from different classes are more similar to each other after message passing. Overall, the results show that stage three has the highest quality of embeddings in terms of node similarity at the residue level. Furthermore, during the experiments, it can be observed that changing the pooling method in the readout layer from “mean aggregation” to “max aggregation” will help to enhance the

models' performance. Therefore, it is necessary to suspect that our prediction head is not able to extract useful information from stage 2 and stage 3 graph embeddings compared to stage 1. Max aggregation is selecting features, i.e. selecting the feature with maximum magnitude and feeding it to the next layer. Thus, we propose that a more careful design needs to be applied in the prediction head in order to grasp high-quality information by applying feature selection of graph representations.

6.1.3 Research Question 3: How to design down-stream model framework to fully utilize obtained atomic-level graph representations?

In order to investigate a structure that can observe the useful information from graph representations, we propose to use a context gating layer as described in 5.2.3 after the multihead attention blocks as a "feature re-weighter" to re-weight features in graph representations. Thus, the model is able to select useful features on its own before feeding input into the pooling layer and prediction head. The following table 6.7 shows the results of adding a context gating layer with $th_f = 8$.

Stage	MAE (Mean ± Std)	RMSE (Mean ± Std)
1	1.069 ± 0.032	1.388 ± 0.020
2	1.079 ± 0.030	1.392 ± 0.015
3	1.062 ± 0.001	1.372 ± 0.010

Table 6.7: Results for adding context gating layer with $th_f = 8$

It can be seen that when $th_f = 8$, stage 3 and stage 2 have a higher performance, while stage 1 has a lower performance. It is viewed that context gating layer indeed has effects on feature selection. The effect of using context gating layer depends on the input graphs. However, the degradation in performance seen in stage 1 indicates that this method is not the optimal solution for feature selection.

6.1.4 Research Question 4: Does our model really learn the complexes indepth from PDBbind data set?

Additional experiments are designed to use protein-only or ligand-only graphs as input to the model to compare the model's performance with complex graphs. In this way, it is clear to understand whether the model is remembering specific protein or ligand structures to predict

or whether the model is "truly" learning from the complex graphs. As processed in previous experiments, protein-only graphs have three-stage experiments.

Stage	MAE (Mean ± Std)	RMSE (Mean ± Std)
1	1.190 ± 0.041	1.501 ± 0.011
2	1.353 ± 0.032	1.677 ± 0.035
3	1.414 ± 0.001	1.792 ± 0.044

Table 6.8: Results for pure protein graphs.

MAE (Mean ± Std)	RMSE (Mean ± Std)
1.474 ± 0.022	1.803 ± 0.008

Table 6.9: Results for pure ligand graphs.

It can be seen from tables 6.8, 6.9 that models perform badly when only using protein-only or ligand-only graphs compared to complex graphs. Further, protein structures seem to contribute more by comparing model performance between protein-only and ligand-only models.

6.2 Comparison of Results from other literatures

In order to compare to the other works that use the PDBbind v2016 data set for the binding affinity task, the results of S. Li, Zhou, Xu, Huang, et al. 2021, Son & Kim 2021, Y. Li et al. 2019 and Zheng et al. 2019 are provided for comparison with our current best model, that is, stage 1 model.

Models	MAE (Mean ± Std)	RMSE (Mean ± Std)
OnionNet	**0.984 ± 0.000**	**1.278 ± 0.000**
SIGN	1.027 ± 0.025	1.316 ± 0.037
GraphBar	1.144 ± 0.033	1.371 ± 0.048
DeepAtoms	1.039 ± 0.016	1.318 ± 0.212
Our Model	**1.036 ± 0.020**	**1.360 ± 0.013**

Table 6.10: Comparison of results with the other models.

The results table 6.10 shows that our model performs relatively well compared to the other state-of-the-art models. It also shows that the residual level model is able to compete with the

full atomic level model in this scenario. Thus, it is valid to choose to use residual level graph with limited computational or time resources.

Chapter 7

Conclusion

In this dissertation, we investigate how the selected model, i.e. transformer based graph neural network reacts when transforming input from amino acids-level graphs to atomic level graphs. It is found out that performance is not enhanced from amino acids-level graphs to atomic level graphs as expected in most cases. We further used the cosine distance between the residue embeddings to evaluate the embedding similarity in three stages. Results show that atomic level graphs have the most high-quality residue embeddings in terms of embeddings' similarity. To extract useful information from the atomic level graph, we further propose to use a context gating layer to automatically re-weight graph representations from GNN models. Finally, the experiment results show that GNN models with atomic-level graphs are able to outperform amino acid-level graphs in specific parameter settings. However, the model with amino acid level graphs can have good performance with the advantage of fewer nodes in graphs compared to the model with atomic level graphs in most cases, which leads to faster training and inference and lower memory demand.

Chapter 8

Future work

Future work for this dissertation includes investigating the three-stage process in different data sets to improve the stability of the conclusion derived. Further four directions can be applied in general for future work.

In terms of model structure, the following subdirections are suggested for future work: 1) More concrete pooling methods can be chosen, i.e. hierarchical graph pooling (Z. Zhang et al. 2019) can be applied in the prediction head. In this way, more related information will be preserved in the pooling process while the recognized "noise" will be discarded. 2) More stable feature selection methods are needed to replace the context gating layer. Context-gating layers only select the preferred features from graph representations with unstable performance.

In terms of node similarity measurement, we believe that global cosine distance is not the best metric to measure the node similarities inside graph-based models. Further metrics need to be selected for measuring node similarity in order to understand to quality of generated graph representations for stages 1, 2 and 3.

In terms of the binding affinity prediction task, we have seen that the threshold th_f is an important hyper-parameter, thus, a more biologically related method needs to be proposed for filtering nodes from protein and potential from ligand for better-quality prediction of binding affinities. The focus should also be on how to model the interaction between protein and ligand molecules.

In terms of the hidden features in the data set, i.e. coordinate information hidden in 3D molecules, we only use this spatial information in defining our type of edges implicitly. However, recent research (Satorras et al. 2021, S. Li, Zhou, Xu, Dou, & Xiong 2021) has shown that models with explicit spatial information will have significant improvement on downstream tasks. Future work can be focused on explicitly leveraging the spatial information in the models.

References

Allen, F. H., Kennard, O., Watson, D. G., Brammer, L., Orpen, A. G., & Taylor, R. (1987). Tables of bond lengths determined by x-ray and neutron diffraction. part 1. bond lengths in organic compounds. *J. Chem. Soc., Perkin Trans. 2*, S1-S19. Retrieved from `http://dx.doi.org/10.1039/P298700000S1` doi: 10.1039/P298700000S1

Asif, N. A., Sarker, Y., Chakrabortty, R. K., Ryan, M. J., Ahamed, M. H., Saha, D. K., ... Tasneem, Z. (2021). Graph neural network: A comprehensive review on non-euclidean space. *IEEE Access*, *9*, 60588-60606. doi: 10.1109/ACCESS.2021.3071274

Bahdanau, D., Cho, K., & Bengio, Y. (2014). *Neural machine translation by jointly learning to align and translate.* arXiv. Retrieved from `https://arxiv.org/abs/1409.0473` doi: 10.48550/ARXIV.1409.0473

Bé guin, L., Vernier, A., Chicireanu, R., Lahaye, T., & Browaeys, A. (2013, jun). Direct measurement of the van der waals interaction between two rydberg atoms. *Physical Review Letters*, *110*(26). Retrieved from `https://doi.org/10.1103%2Fphysrevlett.110.263201` doi: 10.1103/physrevlett.110.263201

Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., ... Bourne, P. E. (2000, 01). The Protein Data Bank. *Nucleic Acids Research*, *28*(1), 235-242. Retrieved from `https://doi.org/10.1093/nar/28.1.235` doi: 10.1093/nar/28.1.235

Beydon, M.-H., Fournier, A., Drugeault, L., & Becquart, J. (2000). Microbiological high throughput screening: An opportunity for the lead discovery process. *SLAS Discovery*, *5*(1), 13-21. Retrieved from `https://www.sciencedirect.com/science/article/pii/S2472555222085586` doi: https://doi.org/10.1177/108705710000500105

Cai, C., & Wang, Y. (2020). *A note on over-smoothing for graph neural networks.* arXiv. Retrieved from https://arxiv.org/abs/2006.13318 doi: 10.48550/ARXIV.2006.13318

Chai, J., Zeng, H., Li, A., & Ngai, E. W. (2021). Deep learning in computer vision: A critical review of emerging techniques and application scenarios. *Machine Learning with Applications*, *6*, 100134. Retrieved from https://www.sciencedirect.com/science/article/pii/S2666827021000670 doi: https://doi.org/10.1016/j.mlwa.2021.100134

Choudhary, K., & DeCost, B. (2021, nov). Atomistic line graph neural network for improved materials property predictions. *npj Computational Materials*, *7*(1). Retrieved from https://doi.org/10.1038%2Fs41524-021-00650-1 doi: 10.1038/s41524-021-00650-1

Cock, P. J., Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J., Dalke, A., ... others (2009). Biopython: freely available python tools for computational molecular biology and bioinformatics. *Bioinformatics*, *25*(11), 1422–1423.

Cohen, P. (2002, April 1). Protein kinases - the major drug targets of the twenty-first century? *Nature Reviews Drug Discovery*, *1*(4), 309–315. doi: 10.1038/nrd773

The difference between ki, kd, ic50, and ec50 values. (n.d.). Retrieved from https://www.sciencesnail.com/science/the-difference-between-ki-kd-ic50-and-ec50-values

Dwivedi, V. P., & Bresson, X. (2020). *A generalization of transformer networks to graphs.* arXiv. Retrieved from https://arxiv.org/abs/2012.09699 doi: 10.48550/ARXIV.2012.09699

FLEMING, A. (1944, 01). THE DISCOVERY OF PENICILLIN. *British Medical Bulletin*, *2*(1), 4-5. Retrieved from https://doi.org/10.1093/oxfordjournals.bmb.a071032 doi: 10.1093/oxfordjournals.bmb.a071032

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., & Dahl, G. E. (2017, 06–11 Aug). Neural message passing for quantum chemistry. In D. Precup & Y. W. Teh (Eds.), *Proceedings of the 34th international conference on machine learning* (Vol. 70, pp. 1263–1272). PMLR. Retrieved from https://proceedings.mlr.press/v70/gilmer17a.html

He, K., Zhang, X., Ren, S., & Sun, J. (2015). *Deep residual learning for image recognition.* arXiv. Retrieved from https://arxiv.org/abs/1512.03385 doi: 10.48550/ARXIV.1512.03385

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Jarmoskaite, I., AlSadhan, I., Vaidyanathan, P. P., & Herschlag, D. (2020, aug). How to measure and evaluate binding affinities. *eLife*, *9*, e57264. Retrieved from `https://doi.org/10.7554/eLife.57264` doi: 10.7554/eLife.57264

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., ... Hassabis, D. (2021, 08). Highly accurate protein structure prediction with alphafold. *Nature*, *596*, 1-11. doi: 10.1038/s41586-021-03819-2

Jørgensen, P. B., Jacobsen, K. W., & Schmidt, M. N. (2018). *Neural message passing with edge updates for predicting properties of molecules and materials.* arXiv. Retrieved from `https://arxiv.org/abs/1806.03146` doi: 10.48550/ARXIV.1806.03146

Karimi, M., Wu, D., Wang, Z., & Shen, Y. (2018, 06). *Deepaffinity: Interpretable deep learning of compound-protein affinity through unified recurrent and convolutional neural networks.*

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. *CoRR*, *abs/1412.6980*.

Kipf, T. N., & Welling, M. (2016). *Semi-supervised classification with graph convolutional networks.* arXiv. Retrieved from `https://arxiv.org/abs/1609.02907` doi: 10.48550/ARXIV.1609.02907

Kortagere, S., Krasowski, M., & Ekins, S. (2009, 02). The importance of discerning shape in molecular pharmacology. *Trends in pharmacological sciences*, *30*, 138-47. doi: 10.1016/j.tips.2008.12.001

Krüger, M., & Linke, W. (2011, 03). The giant protein titin: A regulatory node that integrates myocyte signaling pathways. *The Journal of biological chemistry*, *286*, 9905-12. doi: 10.1074/jbc.R110.173260

LeCun, Y., & Bengio, Y. (1998). Convolutional networks for images, speech, and time series. In *The handbook of brain theory and neural networks* (p. 255–258). Cambridge, MA, USA: MIT Press.

Lee, I., Keum, J., & Nam, H. (2018, 11). *Deepconv-dti: Prediction of drug-target interactions via deep learning with convolution on protein sequences.*

Lengauer, T., & Rarey, M. (1996). Computational methods for biomolecular docking. *Current Opinion in Structural Biology*, *6*(3), 402-406. Retrieved from `https://`

www.sciencedirect.com/science/article/pii/S0959440X96800613 doi: https://doi.org/10.1016/S0959-440X(96)80061-3

Li, G., Müller, M., Ghanem, B., & Koltun, V. (2021). Training graph neural networks with 1000 layers. *CoRR*, *abs/2106.07476*. Retrieved from https://arxiv.org/abs/2106.07476

Li, J., Fu, A., & Zhang, L. (2019, 03). An overview of scoring functions used for protein–ligand interactions in molecular docking. *Interdisciplinary Sciences: Computational Life Sciences*, *11*. doi: 10.1007/s12539-019-00327-w

Li, S., Zhou, J., Xu, T., Dou, D., & Xiong, H. (2021). *Geomgcl: Geometric graph contrastive learning for molecular property prediction.* arXiv. Retrieved from https://arxiv.org/abs/2109.11730 doi: 10.48550/ARXIV.2109.11730

Li, S., Zhou, J., Xu, T., Huang, L., Wang, F., Xiong, H., ... Xiong, H. (2021). *Structure-aware interactive graph neural networks for the prediction of protein-ligand binding affinity.* arXiv. Retrieved from https://arxiv.org/abs/2107.10670 doi: 10.48550/ARXIV.2107.10670

Li, Y., Rezaei, M., Li, C., & Li, X. (2019, 11). Deepatom: A framework for protein-ligand binding affinity prediction. In (p. 303-310). doi: 10.1109/BIBM47256.2019.8982964

Lim, J., Ryu, S., Park, K., Choe, Y., Ham, J., & Kim, W. (2019b, 08). Predicting drug-target interaction using a novel graph neural network with 3d structure-embedded graph representation. *Journal of Chemical Information and Modeling*, *59*. doi: 10.1021/acs.jcim.9b00387

Lim, J., Ryu, S., Park, K., Choe, Y. J., Ham, J., & Kim, W. Y. (2019a). *Predicting drug-target interaction using 3d structure-embedded graph representations from graph neural networks.* arXiv. Retrieved from https://arxiv.org/abs/1904.08144 doi: 10.48550/ARXIV.1904.08144

Lopez, M. J., & Mohiuddin, S. S. (2021). *Biochemistry, essential amino acids.* StatPearls Publishing, Treasure Island (FL). Retrieved from http://europepmc.org/books/NBK557845

McSween Jr, H., & Huss, G. (2021). *Cosmochemistry.* Cambridge University Press.

Miech, A., Laptev, I., & Sivic, J. (2017). *Learnable pooling with context gating for video classification.* arXiv. Retrieved from https://arxiv.org/abs/1706.06905 doi: 10.48550/ARXIV.1706.06905

Mukherjee, A., Bhimalapuram, P., & Bagchi, B. (2005, 08). Orientation-dependent potential of mean force for protein folding. *The Journal of chemical physics*, *123*, 014901. doi: 10.1063/1.1940058

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on international conference on machine learning* (p. 807–814). Madison, WI, USA: Omnipress.

Nguyen, T., le, H., Quinn, T., Nguyen, T., le, T., & Venkatesh, S. (2020, 10). Graphdta: Predicting drug–target binding affinity with graph neural networks. *Bioinformatics*, *37*. doi: 10.1093/bioinformatics/btaa921

O'Boyle, N., Banck, M., James, C., Morley, C., Vandermeersch, T., & Hutchison, G. (2011, 10). Open babel: An open chemical toolbox. *Journal of cheminformatics*, *3*, 33. doi: 10.1186/1758-2946-3-33

Pathak, Y., Mehta, S., & Priyakumar, U. D. (2021). Learning atomic interactions through solvation free energy prediction using graph neural networks. *Journal of Chemical Information and Computer Sciences*, *61*(2), 689-698. Retrieved from `https://doi.org/10.1021/acs.jcim.0c01413` doi: 10.1021/acs.jcim.0c01413

The pdbbind database: methodologies and updates. (2005). *Journal of Medicinal Chemistry*, *48*(12), 4111-4119. doi: 10.1021/JM048957Q

Radzicka, A., & Wolfenden, R. (1995, 02). A proficient enzyme. *Science (New York, N.Y.)*, *267*, 90-3. doi: 10.1126/science.7809611

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, *65 6*, 386-408.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*, 533-536.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. Retrieved from `https://arxiv.org/abs/1801.04381` doi: 10.48550/ARXIV.1801.04381

Satorras, V. G., Hoogeboom, E., & Welling, M. (2021). *E(n) equivariant graph neural networks.* arXiv. Retrieved from https://arxiv.org/abs/2102.09844 doi: 10.48550/ARXIV.2102.09844

Schrödinger, L., & DeLano, W. (n.d.). *Pymol.* Retrieved from http://www.pymol.org/pymol

Shi, Y., Huang, Z., Feng, S., Zhong, H., Wang, W., & Sun, Y. (2020). *Masked label prediction: Unified message passing model for semi-supervised classification.* arXiv. Retrieved from https://arxiv.org/abs/2009.03509 doi: 10.48550/ARXIV.2009.03509

Smith, M. B. (2020). *March's advanced organic chemistry: reactions, mechanisms, and structure.* John Wiley & Sons.

Son, J., & Kim, D. (2021, 04). Development of a graph convolutional neural network model for efficient prediction of protein-ligand binding affinities. *PLOS ONE*, *16*, e0249404. doi: 10.1371/journal.pone.0249404

Su, M., Yang, Q., Du, Y., Guoqin, F., Liu, Z., Li, Y., & Wang, R. (2018, 11). Comparative assessment of scoring functions: The casf-2016 update. *Journal of Chemical Information and Modeling*, *59*. doi: 10.1021/acs.jcim.8b00545

Thafar, M., Raies, A., Albaradei, S., Essack, M., & Bajic, V. (2019, 11). Comparison study of computational prediction tools for drug-target binding affinities. *Frontiers in Chemistry*, *7*, 782. doi: 10.3389/fchem.2019.00782

Torfi, A., Shirvani, R. A., Keneshloo, Y., Tavaf, N., & Fox, E. A. (2020). Natural language processing advancements by deep learning: A survey. *CoRR*, *abs/2003.01200*. Retrieved from https://arxiv.org/abs/2003.01200

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). *Attention is all you need.* arXiv. Retrieved from https://arxiv.org/abs/1706.03762 doi: 10.48550/ARXIV.1706.03762

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2017). *Graph attention networks.* arXiv. Retrieved from https://arxiv.org/abs/1710.10903 doi: 10.48550/ARXIV.1710.10903

Venkatachalam, C., Jiang, X., Oldfield, T., & Waldman, M. (2003, 02). Ligandfit: A novel method for the shape-directed rapid docking of ligands to protein active sites. *Journal of molecular graphics modelling*, *21*, 289-307. doi: 10.1016/S1093-3263(02)00164-X

Wang, C., & Zhang, Y. (2017, January 30). Improving scoring-docking-screening powers of protein–ligand scoring functions using random forest. *Journal of Computational Chemistry*, *38*(3), 169–177. (Funding Information: We thank NYU-ITS and NYUAD for providing computational resources. Publisher Copyright: © 2016 Wiley Periodicals, Inc.) doi: 10.1002/jcc.24667

Wang, J., Cao, D., Tang, C., Xu, L., He, Q., Yang, B., ... Hou, T. (2020, 08). DeepAtomicCharge: a new graph convolutional network-based architecture for accurate prediction of atomic charges. *Briefings in Bioinformatics*, *22*(3). Retrieved from `https://doi.org/10.1093/bib/bbaa183` (bbaa183) doi: 10.1093/bib/bbaa183

Wei, X., Huang, H., Ma, L., Yang, Z., & Xu, L. (2020, 10). Recurrent graph neural networks for text classification. In (p. 91-97). doi: 10.1109/ICSESS49938.2020.9237709

Xie, T., & Grossman, J. C. (2018, apr). Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. *Physical Review Letters*, *120*(14). Retrieved from `https://doi.org/10.1103%2Fphysrevlett.120.145301` doi: 10.1103/physrevlett.120.145301

Xiong, Z., Wang, D., Liu, X., Zhong, F., Wan, X., Li, X., ... Zheng, M. (2020, August). Pushing the boundaries of molecular representation for drug discovery with the graph attention mechanism. *Journal of medicinal chemistry*, *63*(16), 8749—8760. Retrieved from `https://doi.org/10.1021/acs.jmedchem.9b00959` doi: 10.1021/acs.jmedchem.9b00959

Xu, B., Wang, N., Chen, T., & Li, M. (2015). *Empirical evaluation of rectified activations in convolutional network.* arXiv. Retrieved from `https://arxiv.org/abs/1505.00853` doi: 10.48550/ARXIV.1505.00853

Yang, J., Shen, C., & Huang, N. (2020, 02). Predicting or pretending: Artificial intelligence for protein-ligand interactions lack of sufficiently large and unbiased datasets. *Frontiers in Pharmacology*, *11*, 69. doi: 10.3389/fphar.2020.00069

Zhang, S., Tong, H., Xu, J., & Maciejewski, R. (2019, 11). Graph convolutional networks: a comprehensive review. *Computational Social Networks*, *6*. doi: 10.1186/s40649-019-0069-y

Zhang, S., Yao, L., Sun, A., & Tay, Y. (2020, jan). Deep learning based recommender system. *ACM Computing Surveys*, *52*(1), 1–38. doi: 10.1145/3285029

Zhang, Z., Bu, J., Ester, M., Zhang, J., Yao, C., Yu, Z., & Wang, C. (2019). *Hierarchical graph pooling with structure learning.* arXiv. Retrieved from `https://arxiv.org/abs/1911.05954` doi: 10.48550/ARXIV.1911.05954

Zheng, L., Fan, J., & Mu, Y. (2019, 09). Onionnet: a multiple-layer intermolecular-contact-based convolutional neural network for protein–ligand binding affinity prediction. *ACS Omega*, *4*. doi: 10.1021/acsomega.9b01997

Öztürk, H., Özgür, A., & Ozkirimli, E. (2018, 09). DeepDTA: deep drug–target binding affinity prediction. *Bioinformatics*, *34*(17), i821-i829. Retrieved from `https://doi.org/10.1093/bioinformatics/bty593` doi: 10.1093/bioinformatics/bty593