THESIS

ASSESSMENT OF PROTEIN-PROTEIN INTERFACES USING GRAPH NEURAL NETWORKS

Submitted by

Yashwanth Reddy Virupaksha

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Summer 2021

Master's Committee:

Advisor: Asa Ben-Hur

Charles W Anderson
Henry Hugh Adams

ABSTRACT

ASSESSMENT OF PROTEIN-PROTEIN INTERFACES USING GRAPH NEURAL NETWORKS

Proteins are fundamental building blocks of cellular function. They systematically interact with other proteins to make life happen. Understanding these protein-protein interactions is important for obtaining a detailed understanding of protein function and to enable the process of drug and vaccine design. Experimental methods for studying protein interfaces including X-ray crystallography, NMR, and Cryo-electron microscopy, are expensive, time consuming, and sometimes unsuccessful due to the unstable nature of many protein-protein interactions. Computational docking experiments are a cheap and fast alternative. Docking algorithms produce a large number of potential solutions that are then ranked by quality. However, current scoring methods are not good enough for finding a docking solution that is close to the native structure. That has led to the development of machine learning methods for this task. These methods typically involve extensive engineering of features to describe the protein complex, and are not very successful at identifying good quality solutions among the top ranks. In this thesis, we propose a scoring technique that uses graph neural networks that function at the atomic level to learn the interfaces of docked proteins without the need for feature engineering. We evaluate our model and show that it performs better than commonly used docking methods and deep learning methods that use 3D CNNs.

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Asa Ben-Hur for his support, encouragement, and guidance throughout my time at CSU. Thanks to my parents for giving me this life and supporting my education. I would also like to thank my colleagues Mridula Bontha, Don Neumann, Ali Ebrahimpour, Fahad Ullah, and Soumyadip Roy, for helping me with my research and supporting me. Finally, I would like to thank everyone who were part of my life.

TABLE OF CONTENTS

ABSTRACT ii
ACKNOWLEDGEMENTS iii
LIST OF TABLES vi
LIST OF FIGURES vii

Chapter 1 Introduction 1
1.1 Proteins 2
1.1.1 Protein Structure 4
1.1.2 Protein complexes 5
1.1.3 Interfaces of protein complexes 8
1.2 Determining the structure of protein complexes 10
1.2.1 Experimental methods 10
1.2.2 Protein-Protein Docking 15
1.2.3 Scoring docked protein-protein models 16
1.2.4 Evaluation of docked protein-protein models 17

Chapter 2 Traditional approaches in scoring docked protein-protein models 20
2.1 Physics-based approaches 20
2.2 Interface-based approaches 21
2.3 Knowledge-based statistical potentials 24
2.4 Machine learning based approaches 25

Chapter 3 Artificial Neural Networks 27
3.1 Convolutional Neural Networks 32
3.2 Graph Neural Network 35

Chapter 4 Application of Neural Networks in scoring docked protein-protein models . . 38

Chapter 5 Proposed Neural Network techniques 42
5.1 Docked protein-protein models as a Graph 42
5.2 Proposed Graph Convolutional Networks 46
5.3 Proposed Graph Attention Networks 47
5.4 Graph Pooling 48

Chapter 6 Data and training 50
6.1 Data 50
6.1.1 Protein-Protein interactions databases 50
6.1.2 Decoy generation using docking softwares 51
6.1.3 Train, Validation, and Test sets 52
6.2 Training 53
6.2.1 Multi-level sampler 53
6.2.2 Model architecture 54

6.2.3 Model training . . . 55
6.2.4 Metrics . . . 56
6.2.5 Model Selection . . . 57

Chapter 7 Results . . . 59
7.1 Comparision of proposed Graph Neural Networks . . . 59
7.2 Comparison against other methods . . . 63
7.3 Misclassifications . . . 67
7.4 Conclusion . . . 68

Chapter 8 Future work . . . 70

Bibliography . . . 72

Appendix A Supplementary Data . . . 85
A.1 Results on validation set . . . 85
A.2 Loss curve . . . 85

LIST OF TABLES

1.1 Quality categories and corresponding parameter limits proposed by CAPRI and DOCKQ 19

5.1 Atom types used in constructing node features. 45

6.1 Summary of number of decoys per docking software 51
6.2 Distribution of the complexes and decoys in train, validation, and test sets 52
6.3 Hyper-parameters and their search space used in model selection. The best combination is listed in the last column. 58

7.1 List of experiments 60

LIST OF FIGURES

1.1 The 21 different amino acids categorized by the electrical properties of the side chain. Each of the amino acids full name, three letter and single letter abbreviations are given. The chemical structures are oriented with the carboxyl group, α-carbon, and amine groups on top and the side chain extending downwards. This work is "Molecular structures of the 21 proteinogenic amino acids" by Dan Cojocari, Princess Margaret Cancer Centre, University of Toronto. It is under a Creative Commons Attribution-Share Alike 3.0 Unported license. File: https://commons.wikimedia.org/wiki/File:Molecular_structures_of_the_21_proteinogenic_amino_acids.svg, License: https://creativecommons.org/licenses/by-sa/3.0/deed.en 3

1.2 Two amino acids react with each other resulting in a peptide bond. The left hand side of the chemical reaction shows two amino acids: one having the carboxyl group and the other having the amine group. The right hand side of the chemical reaction shows two products: peptide and water molecule. The chemical elements belonging to the resultant water molecule are colored blue whereas those belonging to the peptide bond are colored red. The R in the structural formula of both the amino acids and the peptide represents the side chain. This work is "Basic amino acid condensation" by Rik van der Lingen. File: https://commons.wikimedia.org/wiki/File:AminoacidCondensation.svg. License: Not required. 4

1.3 Structural representation of the protein 1AXC at Primary, secondary, tertiary, and Quaternary levels. Primary structure starting with the N-terminus (left end), followed by the amino acids represented using three letters (left to right), and ending with the C-terminus (right end). Secondary structure represented with α-helix in red color and anti-parallel β-sheets in blue color. Tertiary structure represented in the form of a 3D structure with α-helix in red and β-sheet in blue color. The quaternary structure is a protein complex made up of three polypeptide chains with each chain (monomer) representing the same protein 1AXC. This work is "Summary of protein structure (primary, secondary, tertiary, and quaternary) using the example of PCNA. (PDB: 1AXC)" by Thomas Shafee. It is under a Creative Commons Attribution 4.0 International license. FIle: https://upload.wikimedia.org/wikipedia/commons/5/5f/Protein_structure_%28full%29.png. License: https://creativecommons.org/licenses/by/4.0/deed.en . . . 6

1.4 3D cartoon of Staphylococcus aureus UDG / UGI complex (3WDG) [107] determined using X-Ray diffraction technique. The two colors red and blue show the presence of two different chains in the protein complex. The interaction of the two chains can be observed by the close proximity of the red and blue colored proteins. Image created using Chimera [82] 8

1.5 3D cartoon of bovine alpha-chymotrypsin-eglin c complex (1ACB) [33] showing the interface of the protein complex in yellow color, part of the receptor in red color and part of the ligand in blue color 9

1.6 Procedure to generate the three dimensional structure of the protein using X-Ray Crystallography technique. It is under a Creative Commons Attribution-Share Alike 3.0 Unported license. File: https://upload.wikimedia.org/wikipedia/commons/7/73/X_ray_diffraction.png, License: https://creativecommons.org/licenses/by-sa/3.0/deed.en . . . 12

3.1 Structure of the Biological neuron showing the dendrites, soma, and axon. Attached to the axon is a myelin sheath that acts as an insulator and increases the electrical signal's speed of transmission. Mathematically, the biological neuron can be represented by a function f that transforms the inputs $x_1, \ldots, x_n$ to y. It is under a Creative Commons Attribution-Share Alike 3.0 Unported license. File: https://commons.wi kimedia.org/wiki/File:Neuron3.png#/media/File:Neuron3.svg, License: https://creativecommons.org/licenses/by-sa/3.0/deed.en 28
3.2 A feed forward neural network (FFNN) with L layers. The diagram shows three fundamental layers: input layer in orange color, hidden layers in blue color, and output layer in green color. x is the input. For each hidden layer l, $w^{(l)}$ and $b^{(l)}$ are the learnable parameters, $z^{(l)}$ and $a^{(l)}$ are the intermediate values. The intermediate outputs for unit i and hidden layer l can be identified using $z_i^{(l)}$ $a_i^{(l)}$. The output is $\hat{y}$. 29
3.3 Application of strided convolution where a filter of the size of the receptive field 3×3 is applied on the input to get an output unit. All output units are computed by sliding the receptive fields and performing the convolution operation. 33
3.4 Comparision of the convolution operation between the CNN on the left and the GCN on the right. The CNN aggregates all the hidden representation surrounding a central pixel C in a receptive field shown in green to output the final representation of the central pixel C show in red color. Similarly, the GCN aggregates the hidden representations of the neighboring nodes around a central node to output the central nodes final representation shown in red color. 35

5.1 Graph extracted from the interface of the protein complex. The protein complex consists of two chains A and B, shown in blue and orange. The atoms belonging to residues 1, 2, and 3 are circled in green, red, and yellow. The atoms represent the nodes of the graph, and the edges between the nodes show the distance. All interface atoms are labeled as I, and the neighboring atoms are labeled as N. 43
5.2 Interface extracted from a docked protein-protein model (model.000.00.pdb) generated using ClusPRO docking software from the unbound proteins of the complex 1ACB. The atoms from the entire docked protein-protein model are shown in red color. The extracted interface is shown using yellow and green colors with yellow color representing the interfacing atoms and the green color representing the neighboring atoms of the interfacing atoms. 44

6.1 Multi-level sampling process. 54
6.2 Model architecture showing the propogation of inputs through multiple graph convolutional layers, a graph pooling layer, a dense layer and finally a single neuron that produces the predicted score. 55

7.1 Comparing top-n ranking performance of the experiments listed in Table 7.1 on the test set. Reported are the number of complexes having (a) high quality and above decoys and (b) natives, in top-p ranks. 61
7.2 Comparing top-n ranking performance of the best model at different quality thresholds on the test set. Reported are the number of complexes having docking decoys above the quality threshold in top-p ranks. 62
7.3 Comparing top-n ranking performance of the best model against (a) ZDock, (b) FroDock, and (c) Haddock docking + scoring methods at different quality thresholds on the test set. Reported are the number of complexes having docking decoys above the quality threshold in top-1, 5, 10, 20, and 50 ranks. 64
7.4 Comparing top-n ranking performance of the best model vs Dove[109] at different quality thresholds on the test set. Reported are the number of complexes having docking decoys above acceptable and high quality threshold in top-p ranks. 65
7.5 Comparing top-n ranking performance of the best model vs 3DCNN at different quality thresholds on the test set. Reported are the number of complexes having docking decoys above acceptable and high quality threshold in top-p ranks. 66
7.6 Misclassified docking solution for the protein complex (a) 1OUS generated using FroDock docking software (b) 1S1Q generated using Haddock docking software (c) 2FHZ generated using ClusPro docking software, and their corresponding native complex 68

A.1 Comparing top-n ranking performance of the experiments listed in Table 7.1 on validation set. Reported are the number of complexes having (a) high quality and above decoys and (b) natives, in top-p ranks. 86
A.2 Loss curve of the best model during training. 87

Chapter 1

Introduction

Proteins are essential molecules that perform a vast spectrum of cellular processes. They interact with other proteins or small molecules most of the time, hence their full potential isn't unlocked when they act alone. Understanding protein interactions empowers the current knowledge of disease etiology and the rational design of therapeutic molecules. Protein interactions occur within a localized region of the protein called the interface, which has unique properties [72].

Identifying the interfaces between proteins involves finding the 3D protein structures using experimental approaches such as x-ray crystallography or nuclear magnetic resonance techniques. These experimental techniques are time-consuming, expensive, and sometimes unsuccessful [13, 47, 108]. Computational methods like protein docking algorithms are designed as an alternative to the experimental approaches and are faster and inexpensive. The two major tasks of the protein docking algorithms are to provide docking solutions and assess these solutions' quality, both of which are challenging problems.

This thesis presents a novel assessment method to determine the docking solutions that are similar to the native structure by examining the interface of a docking solution. This assessment method uses a new realm in deep learning called Graph Neural Networks, motivated by the success of neural network approaches in Euclidean spaces [111]. The docking solution's interface is represented as a graph and fed to a Graph Neural Network. The network predicts an assessment score of the closeness of the docking solution's interface to the native structure. This method outperforms the existing state-of-the-art approaches based on deep learning [109, 17].

The rest of the thesis is organized as follows. The remainder of the introduction overviews proteins, protein complexes, experimental methods, docking methods, and scoring techniques. Chapter 2 reviews the traditional approaches to scoring docked protein-protein models. Chapter 3 gives an overview of artificial neural networks in which graph neural network is introduced. Chapter 4 discusses existing neural network-based approaches for scoring docked protein-protein

models. Chapter 5 explains our proposed neural network techniques. In Chapter 6, we discuss the construction of our dataset and training procedure. Chapter 7 discusses the experimental results and compares our best model against other scoring techniques. Chapter 8 gives ideas for future research. Appendix A discusses additional results.

1.1 Proteins

All organisms are made up of the crucial building blocks called proteins. They are responsible for most cellular activity and help maintain the structure, function, and regulation of the tissues and organs in an organism. Proteins are formed through a process called gene expression, which consists of two steps. The first step involves converting the DNA sequence to mRNA; this process is called transcription. In the second step, mRNA is translated into a chain of amino acids that make up a protein. Amino acids are organic compounds made up of amine $(-NH_2)$, carboxyl groups (-COOH), and a side chain (R group). The geometry of amino acids is tetrahedral with a α carbon atom connecting the amine, carboxyl groups, and the side chain. The difference in the side chain gives rise to 21 different amino acids in eukaryotic DNA with different structural and electrochemical properties. Figure 1.1 shows the different amino acids.

Amino acids are linked by a covalent chemical bond called a peptide bond. Peptide bond formation occurs when the carbon atom in the carboxyl group of one molecule bonds covalently with the nitrogen atom in the amine group of another molecule, releasing a water molecule. All the amino acids participating in such bonds are called residues. The atoms participating in peptide bonds are referred to as the backbone of the peptide chain. The peptide ordering is canonical, starting with the residue's N-terminus with a free amine group to the C-terminus of the residue with a free carboxyl group as shown in Figure 1.2. This ordering is in congruence with the order of polypeptides created during gene expression.

The side chain of a residue plays a vital role in interacting with other atoms and molecules. For instance, the hydrophilic polar chains attract water molecules, and the hydrophobic non-polar

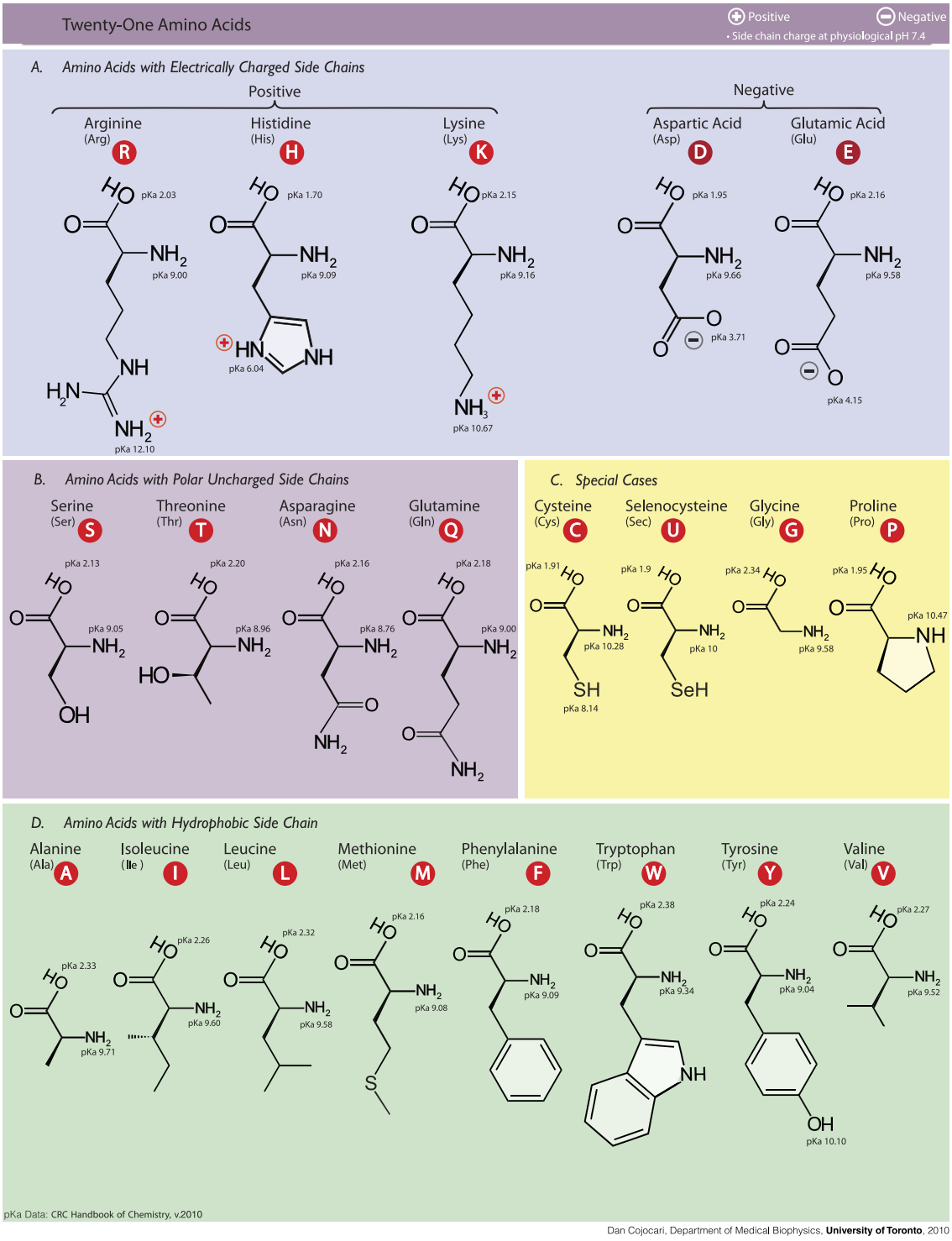


Figure 1.1: The 21 different amino acids categorized by the electrical properties of the side chain. Each of the amino acids full name, three letter and single letter abbreviations are given. The chemical structures are oriented with the carboxyl group, α-carbon, and amine groups on top and the side chain extending downwards. This work is "Molecular structures of the 21 proteinogenic amino acids" by Dan Cojocari, Princess Margaret Cancer Centre, University of Toronto. It is under a Creative Commons Attribution-Share Alike 3.0 Unported license. File: https://commons.wikimedia.org/wiki/File:Molecular_structures_of_the_21_proteinogenic_amino_acids.svg, License: https://creativecommons.o rg/licenses/by-sa/3.0/deed.en

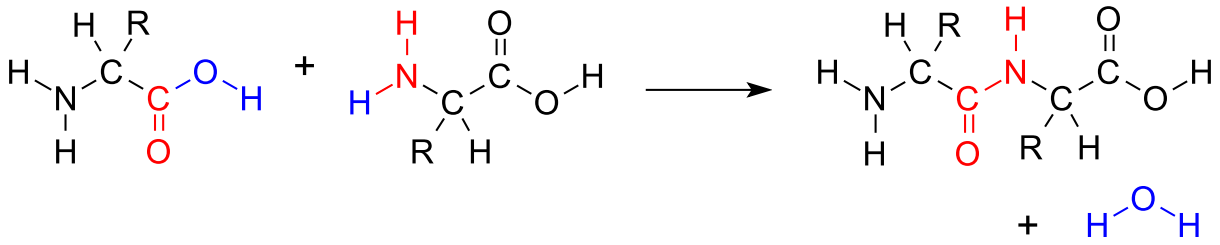


Figure 1.2: Two amino acids react with each other resulting in a peptide bond. The left hand side of the chemical reaction shows two amino acids: one having the carboxyl group and the other having the amine group. The right hand side of the chemical reaction shows two products: peptide and water molecule. The chemical elements belonging to the resultant water molecule are colored blue whereas those belonging to the peptide bond are colored red. The R in the structural formula of both the amino acids and the peptide represents the side chain. This work is "Basic amino acid condensation" by Rik van der Lingen. File: https://commons.wikimedia.org/wiki/File:AminoacidCondensation.svg. License: Not required.

chains avoid polar molecules such as water. Therefore, research has shown that these interactions are responsible for folding proteins into 3D structures [15].

1.1.1 Protein Structure

Protein structure can be explained at four distinct levels with increasing order of complexity. They are named as Primary structure, Secondary structure, Tertiary structure, and Quaternary structure. The protein's primary structure refers to the sequence of amino acids in the polypeptide chain that are linked together in a unique order to form a protein. The sequence of the primary structure begins at the N-terminal and goes through the C-terminal of the polypeptide chain, with each character in the sequence representing the unique letter assigned to different amino acid residues.

One of the peptide bonds' properties is that the nitrogen and carbon atom involved in the peptide bond, and the adjacent α-carbon atoms lie on the same plane called the amide plane. Each amide plane can rotate about its bond to the α-carbon atom. The amide planes can also push away from each other because of the side chain's steric hindrance in the amino acid residue. All these chemical properties provide flexibility in the polypeptide backbone, which can be studied with increasing abstraction levels.

Secondary structure describes the recurring local 3D structures caused by the adjacent amino acid residues in the polypeptide chain. These local structures can be organized into three cate-

gories: α-helix, β-sheet, and loops. The local 3D structure is called α-helix when the polypeptide chain is arranged in a right-handed helix, and the side chains are pointing away from the helical axis. The helical structure is stabilized by the hydrogen bonds acting between one residue's carbonyl oxygen to a second residue's amino group. A local 3D structure is called β-sheet when the non-adjacent parts of the polypeptide chain align so that the residue in one section interacts with other residues in another section through a hydrogen bond. The strands in the β-sheets can be parallel, anti-parallel, or mixed arrangements, with the most common being anti-parallel. The difference between parallel and anti-parallel β sheets is illustrated in Figure 1.3. A loop is a local 3D structure that does not have any patterns. The essential function of a loop is to connect two secondary structures, and the connection is possible due to the lack of hydrogen bonds in it. Figure 1.4 shows a cartoon depiction of part of a protein, with different secondary structural elements highlighted in different colors.

The tertiary structure of the protein combines the α-helices, β-sheets, and the loops to represent the more complex structure of the protein in its final and stable 3D shape. The secondary structures' arrangement is determined from the sequence of amino acids present in the primary structure. In this arrangement process, some residues hide inside the protein core while others reside on the protein surface. One of the protein surfaces' properties is that they have a higher percentage of hydrophilic residues than the protein cores. Similarly, protein cores have a higher percentage of hydrophobic residues compared to the protein surfaces. It is also observed that the protein cores show higher evolutionary conservation than the protein surfaces [112]. Many proteins interact with one or more proteins to form a quarternay structure [15]. The quarternary structure explains how multiple polypeptide chains, also known as subunits, combine, resulting in a protein complex.

1.1.2 Protein complexes

Most biochemical events in a living organism involve physical contact between two or more proteins molecules with high specificity. These biochemical events include hydrogen bonding, electrostatic forces, and hydrophobic effect. The interaction between multiple protein molecules

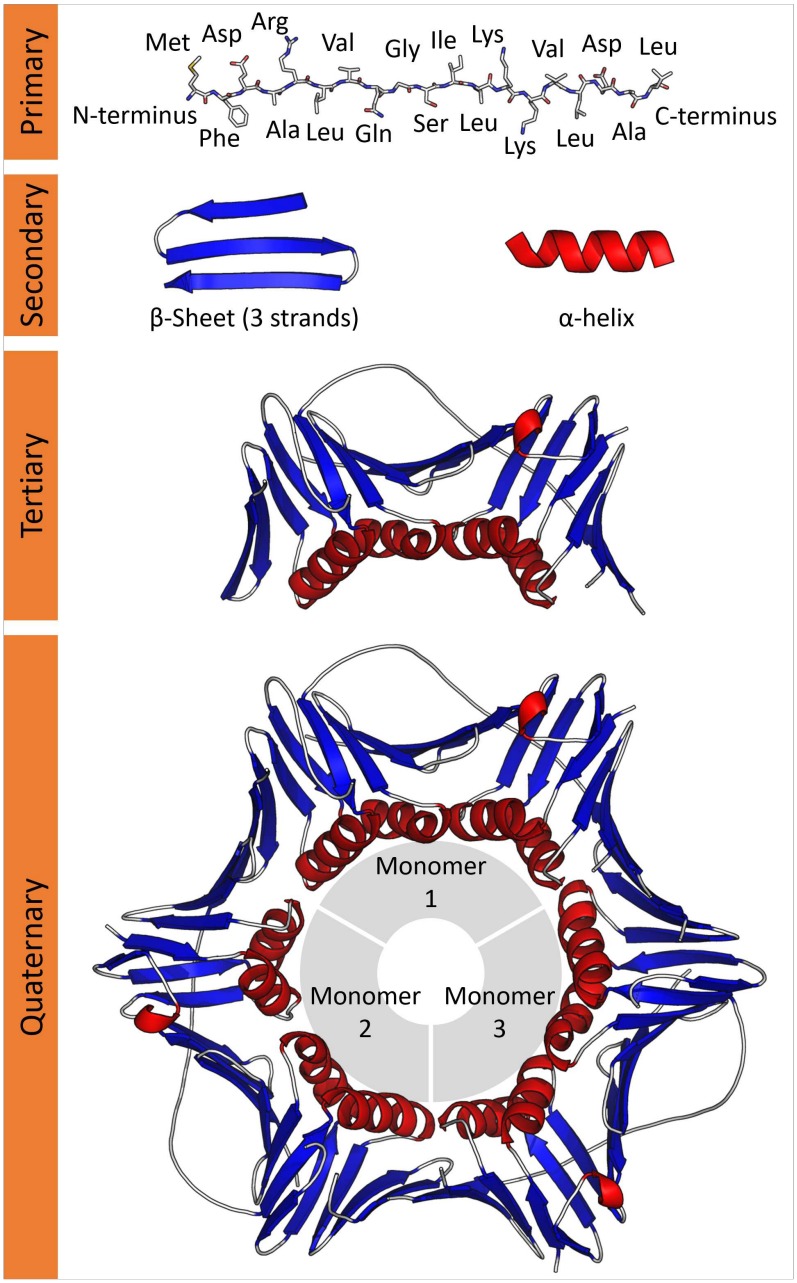


Figure 1.3: Structural representation of the protein 1AXC at Primary, secondary, tertiary, and Quaternary levels. Primary structure starting with the N-terminus (left end), followed by the amino acids represented using three letters (left to right), and ending with the C-terminus (right end). Secondary structure represented with α-helix in red color and anti-parallel β-sheets in blue color. Tertiary structure represented in the form of a 3D structure with α-helix in red and β-sheet in blue color. The quaternary structure is a protein complex made up of three polypeptide chains with each chain (monomer) representing the same protein 1AXC. This work is "Summary of protein structure (primary, secondary, tertiary, and quaternary) using the example of PCNA. (PDB: 1AXC)" by Thomas Shafee. It is under a Creative Commons Attribution 4.0 International license. FIle: https://upload.wikimedia.org/wikipedia/commons/5/5f/Protein_structure_ %28full%29.png. License: https://creativecommons.org/licenses/by/4.0/deed.en

unlocks biological functions at the cellular and system level. One of the crucial steps to understanding the complicated relationship between molecules is to map proteins' physical interactions. This complete map of protein interactions in a living organism is called its interactome [22]. Although the interactome provides a higher-level picture of the PPIs, it fails to give adequate information to study the protein structure for understanding the interaction mechanism, diseases, and developing therapeutics. An example of a protein complex is shown in Figure 1.4. Based on the protein-protein interaction in the protein complex, the PPIs can be divided into multiple types:

- **Homo-oligomers vs. Hetero-oligomers.** Homo-oligomers are complexes containing macromolecules where all the molecules are of the same protein subunit. The non-covalent interactions between the protein subunits guide the formation of the quaternary protein structure. Hetero-oligomers are made up of distinct subunits of proteins.

- **Stable interactions vs. transient interactions.** In stable interactions the interaction between the subunits is long-lived. This type of interaction usually happens in homo-oligomers and some hetero-oligomers. In transient interactions, the interaction between proteins is weak and short-lived thus helping separate the proteins to modulate the functional activity at a specific time point. Compared to stable interactions, transient interactions have less structural and chemical complementarity. In transient complexes the interfaces are small because of less conservation of evolutionary information [51]. Also, the transient interfaces are made up of weak forces for bond formation such as hydrogen bonds and Van der Waals forces. This thesis fouses on protein complexes that are transient because knowing the structure of the transient complexes is a very important and difficult task.

- **Obligate vs. Non-obligate.** Complexes formed by proteins where at least one of the proteins is stable are called non-obligate complexes. Whereas, complexes in which one or more proteins depend on rest of the proteins in the protein complex to remain stable are called obligate complexes [3]. This thesis focuses on complexes that are non-obligate. Obligate

complexes are found to be stable in nature whereas non-obligate are either stable or transient [2].

- **The role of water.** The interaction between proteins is significantly affected by water molecules [49, 8]. Examining the interaction between different complexes shows that water molecules are responsible for the hydrogen bonds between complexes. In some cases, water molecules mediate the interactions between two proteins.

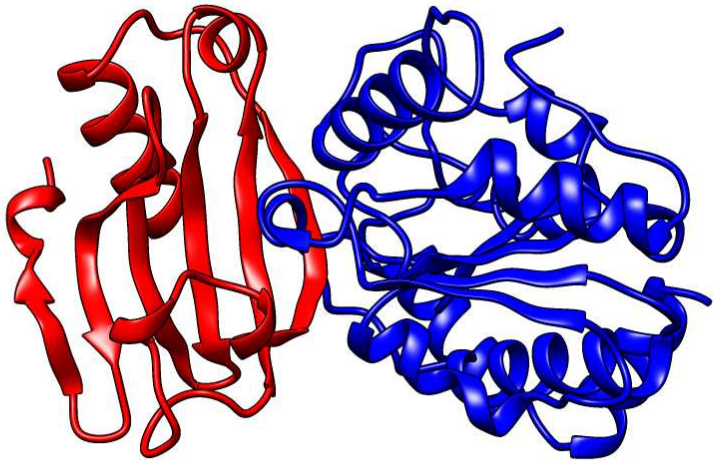


Figure 1.4: 3D cartoon of Staphylococcus aureus UDG / UGI complex (3WDG) [107] determined using X-Ray diffraction technique. The two colors red and blue show the presence of two different chains in the protein complex. The interaction of the two chains can be observed by the close proximity of the red and blue colored proteins. Image created using Chimera [82]

1.1.3 Interfaces of protein complexes

The interface is the region of the protein complex where the proteins bind each other as shown in Figure 1.5. The structure of the interface of a protein complex includes any atom or residue of a protein that is at a distance below the threshold to another atom or residue of a different protein in a protein complex [1, 112, 51, 79]. Proteins bind with each other at the interface to form a

protein complex because of the shape complementarity between proteins [68] and physiochemical properties of atoms or residues in the interface. The key physiochemical properties affecting the protein complex are the electrostatic-complementarity and hydrophobic complementarity between the atoms or residues of different proteins in the interface [112].

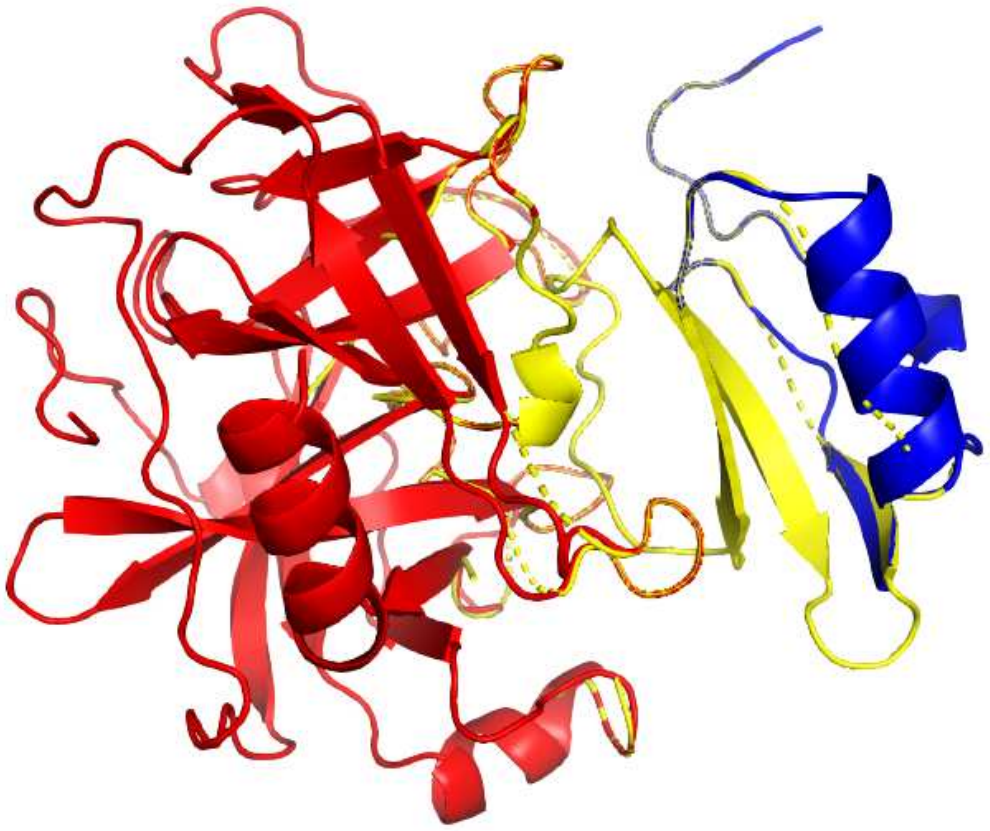


Figure 1.5: 3D cartoon of bovine alpha-chymotrypsin-eglin c complex (1ACB) [33] showing the interface of the protein complex in yellow color, part of the receptor in red color and part of the ligand in blue color

Depending on the type of protein-protein interaction, some protein complexes form with limited structural changes from the unbound structures whereas other protein complexes are formed with significant structural changes to the unbound protein structures [104]. This is done to attain shape and energy complimentarity and is aided by the characteristics of interface such as high presence of hydrophobic residues that help bury the atoms and residues of the interface. Because

all the properties combined at the interface help in stabilizing the protein complex the interface is the single most important region for the resultant protein complex structure.

1.2 Determining the structure of protein complexes

Most of the experimental methods used for studying protein structures are also used in determining the protein-protein complex structure. In studying protein-protein complex structures, the emphasis is placed on understanding how atoms in different proteins function to make the interaction happen. Understanding the interaction between proteins helps us learn how protein complexes perform diverse biological functions in various organisms. This could help us explain the cause of diseases and eventually design therapeutic molecules to cure them. There are two primary ways to study proteins: One uses experimental methods, and the other uses computational methods. Experimental methods are historically the first set of techniques used to understand protein complexes and are well-established methods. The disadvantage with experimental techniques is that the protein complex needs to stabilized for conducting the experiment which is not a feasible task in many cases thus limiting the ability to capture every protein complex structure. Whereas computational techniques, which are the other set of tools to understand proteins, are much more recent developments, and there is still much work in progress [63, 64].

1.2.1 Experimental methods

Experimental methods primarily use various probing techniques to find the structure of the protein complexes. As of 2021, there are about 155,778 protein structures available in the Protein Data Bank [12]. Most of these structures were determined using experimental methods like x-ray Crystallography (138,684), NMR techniques (11701), and electron microscopy (5124). Each of the experimental methods has its unique set of advantages, which can be used to determine the structure of proteins of various degrees of complexity. Some of the most popular experimental methods are described below.

- **X-Ray Crystallography** is the first technique used to determine the protein structure. Initial experiments using this technique were with small molecules, and with its success, scientists used it for macromolecular complexes. The history of determining the protein structures using X-ray Crystallography started with the X-Ray diffraction experiment carried out by Max von Laue in 1912. Later, X-Rays' importance in studying crystal structures was discovered by Lawrence Bragg and his father William Bragg in an experiment where they observed interpretative patterns of spots on photographic plates when the photographic plates were placed close to the crystals exposed with X-Rays. The procedure to spatially re-construct the crystal structure starts by passing x-rays through the crystal to generate the diffraction pattern. The diffraction pattern is the scattered set of rays identified on the photographic plate that is processed further using computer software to generate the electron density map. The electron density map gives the confidence of observing different atom types in a 3D space to construct the atomic model. The last three steps of generating the diffraction pattern, electron density map, and atomic model are repeated many times to increase the confidence of the final atomic model. The entire process described above is neatly illustrated in the Figure 1.4. Although many protein complex structures in PDB were determined using X-Ray crystallography this method only works for protein complexes that can be crystallized. Crystallization of protein complexes is dependent on the stability of the protein complexes thus making it very difficult for transient complexes. Therefore, X-Ray Crystallography may not be the tool that can determine every protein complex structure.

- **Nuclear Magnetic Resonance (NMR)** for protein structure determination is a relatively new development compared to the X-Ray Crystallography technique. The NMR phenomenon was first observed by Felix Bloch and Edward Purcell in 1945. This NMR spectra phenomenon occurs when a particle absorbs a photon's energy and transitions the nuclear spin from the ground state to the excited state. The above observations, combined with radiofrequency irradiation, helped in the initial developments of NMR spectroscopy. Later, the major setbacks in determining the protein structure using the NMR technique were mitigated by

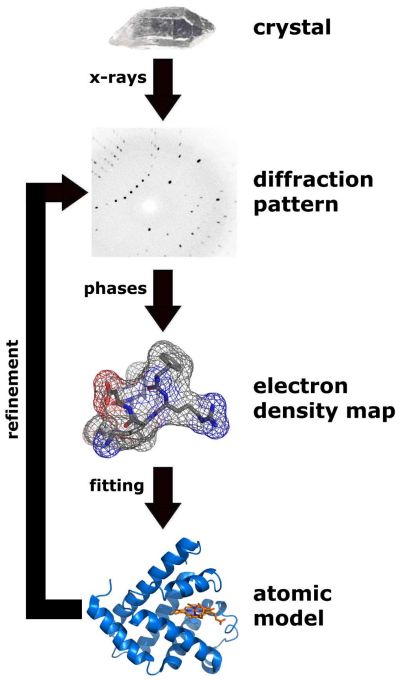


Figure 1.6: Procedure to generate the three dimensional structure of the protein using X-Ray Crystallography technique. It is under a Creative Commons Attribution-Share Alike 3.0 Unported license. File: https://upload.wikimedia.org/wikipedia/commons/7/73/X_ray_ diffraction.png, License: https://creativecommons.org/licenses/by-sa/3.0/deed.en

some key contributions. One such contribution was the observation that each nucleus has a different frequency of nuclear transition than others. The nucleus belonging to different chemical groups have subtle differences in the frequency of transition from one another. Another significant contribution was the transient techniques proposed by Richard Ernst to convert transient signals to a standard spectrum. Before this contribution, the NMR spectrum obtained after bombarding the protein structure with radiation was not understandable to construct the protein structure. The transient techniques which use Fourier transformations helped construct the standard spectrum from the NMR spectrum. They eventually played a crucial role in multi-dimensional NMR spectroscopy.

The protein structure is determined using the NMR technique using the following set of steps:

- **Sample preparation** The protein under investigation is isolated from other structures and placed in an aqueous solution.
- **Data collection** The sample is placed under a powerful magnet that sends radio frequency signals through the samples. In this process, some of the signals are absorbed by the nucleus, and the absorption data is collected.
- **Resonance assignment** From the nuclear magnetic resonance data, the resonant frequency of a nucleus relative to the standard in the magnetic field is determined to identify the atom.
- **Restraint generation** The structure calculation is made using various restraints that have been experimentally determined. The most widely used restraints are distance and angle. Distance restraints are determined using the NOESY experiment, which gives the spatial distance between two nuclei under examination. Angle constraints are determined using the Karplus equation from coupling constants.

The final structure is calculated using computer programs such as XPLOR-NIH, CYANA, or GeNMR, which use the above constraints as input.

- **Cryogenic Electron Microscopy (Cryo-EM)** Visualization is a powerful tool in understanding biological processes, and electron microscopy played an essential role in understanding the cell and its organelles. This technique was extended to determine the 3D structures by Aaron Klug in his experiment, where he studied bacteriophage T4's viral organization. Aaron's work captures 2D electron density maps at different angles of observation to construct the 3D structure. Later, this technique became more robust with developments in technology and understanding of biologies. One such work is Henderson and Unwin's study of proteins at high atom resolution, better instrumentation techniques, specimen preparation, data processing, and increased computational power of electron microscopy. This technique is mostly used where X-ray crystallography and NMR techniques cannot be used.

  One of the significant advances in Electron Microscopy is using Transmission electron microscopes (TEM). TEM shoots an electron beam onto a molecule under study, making the electrons from the beam interact with the molecule and project an image on the detector. The drawback of TEM is that not all biomolecules can be determined because they get destroyed due to high-vacuum conditions and intense electron beams. To overcome these drawbacks, a revolutionary technique called Cryo-EM was proposed in 2017. Cryo-EM uses frozen samples, electron beams that are gentle and sophisticated image processing.

  Cryo-EM technique became successful with the contribution of the three of the authors, Richard Henderson, Joachim Frank, and James Dubochet. Richard Henderson used TEM to generate the first set of protein structures. He added purple light emitting bacteriorhodopsin proteins to the sample and passed an electron beam with low-power at multiple angles to generate a 3D image of the protein. Joachim Frank's work uses cryo-EM images to construct the 3D structure of the protein using computer algorithms. These computer algorithms involve processing and analyzing images randomly oriented to a uniform orientation to get a high-resolution 2D image. All the 2D projections are put together for the 3D image.

  James Dubochet developed a method to freeze water in water-based TEM samples to form a disordered glass instead of crystalline ice. This is done to prevent the strong diffraction of the

electron beam in crystalline ice that can produce incorrect information about the molecule under study. The sample containing water is cryo-frozen by hurling the sample into nitrogen-cooled ethane liquid. This causes the water on the sample to freeze so quickly that crystalline lattice formation is impossible and results in the disordered glass. All three contributions combined resulted in the Cryo-EM technique, which won the Noble prize for all the three authors in 2017.

1.2.2 Protein-Protein Docking

Although 11,600 protein complexes in PDB [12] were determined using experimental methods, they only account for a minuscule percentage of the biological systems' interactions. From the techniques described above, we observed that the experimental methods' significant disadvantage is that they take much time, are expensive, and most importantly, they are not always successful. To address this and quickly solve the protein complex structure, computer algorithms commonly referred to as Protein-Protein Docking in the bioinformatics literature are used.

Protein-Protein Docking, also called Macromolecular docking, is a computational modeling technique to determine the three-dimensional structure of a protein complex given the structures of the unbounded proteins that make up the complex. This technique aims to produce solutions that are similar to the native protein complex by exploring the possible ways in which the proteins can bind each other. This goal is addressed by the protein-protein docking community using two steps: Sampling docking models followed by scoring them. Sampling aims to predict 3D complex structures, a.k.a docking models that are realistic using techniques that explore the proteins' structural dynamics and behavior. The sampling step is followed by scoring, which uses another set of techniques to assess the docked models' quality. These steps, sampling and scoring, ideally give us the docked model closest to the native structure.

Depending on the proteins' conformations before and after binding, protein docking can be classified into rigid docking and flexible docking. Rigid docking is considered when the difference in the proteins' conformation is negligible and involves six degrees (three translational and three rotational) of freedom to bind the proteins. Suppose the difference in conformation is taken into

account. In that case, flexible docking is used, wherein a conformational search is performed to find the bound state of the protein complex from the unbound state of proteins. Protein docking can also be roughly classified as template-based methods (TBM) or ab-initio methods. Template-based methods construct protein complexes of undetermined targets using structural information from other protein complexes that are experimentally determined [95]. Whereas, ab initio methods may not use any external additional information [80]. Although, all of the above docking methods provide samples/decoys that can be near-native, the next important and equally tricky objective is to find the best sample. In the following sections we discuss how the samples are evaluated for ground truth and which brings us to the problem that we are solving in this thesis.

1.2.3 Scoring docked protein-protein models

Scoring is the most important and difficult step in the protein docking process because it directly influences the overall docking performance. As discussed earlier, the sampling step in the docking process produces a large set of decoys that have few near-native models. To identify these near-native models among the large decoy set, scoring techniques are used. The scoring technique assigns a score that helps identify the near-native decoys and compare the quality of a sample w.r.t other samples. Thus the goal of the scoring step is to precisely discriminate the decoys based on the quality and arrange them in an order to help identify the near-native samples. Another important goal of the scoring function is to be computationally efficient. Scoring techniques use different strategies to assign scores:

- **Physics based approaches** mainly use molecular dynamics (MD) to identify near-native from the incorrect docked models. MD use force fields to estimate the strength of the interaction between the atoms of protein molecules in the complex. The strength can be quantified by summing up the forces due to van der Waals and electrostatic.

- **Interface based approaches** use the interface of the docked model to score. As explained in Section 1.1.3, the atoms at the interface play a decisive role in how the proteins interact.

Thus information that encapsulates the properties of the interface are used such as the shape complimentarity and evolutionary information of the interface.

- **Knowledge-based statistical potential approaches** use Knowledge-based potentials that are derived from known structures of protein-protein complexes. The entire method works on the idea that certain types of molecular interactions are favorable if the protein-protein complexes exhibit a certain kind of behavior more frequently.

- **Machine Learning based approaches** Scoring using Machine Learning involves optimizing a set of non-linear equations to map the relevant features extracted from the docked protein-protein model to the scores that predict the quality. The difference between Knowledge-based statistical potential and Machine learning approaches is that in Knowledge-based statistical potential the mapping between the derived potentials and the near-native is visibly evident whereas in machine learning based methods the mapping between the features and near-natives is needs to be derived.

The prior work in scoring docked protein-protein models is discussed in Chapter 2 and Chapter 4 where different scoring techniques are discussed briefly along with their limitations.

1.2.4 Evaluation of docked protein-protein models

Docking models are evaluated by comparing the structure of the model to its native counterpart. The standards used in evaluation are provided by Critical Assessment of Predicted Interactions (CAPRI) [48, 50]. CAPRI uses the following three measures to assess model quality: $\mathrm{F_{nat}}$, LRMS, and iRMS. These three measures use the interface of the protein complex for calculating the quality score. The protein complex is made up of two proteins that are called the ligand and receptor. The ligand is usually the protein molecule that has shorter chain in the protein complex and the other protein molecule that has larger chain is called the receptor. In order to calculate all the measures CAPRI defines the interface as all pairs of Cα atoms that are at a distance of at most 5Å, where each Cα atom in the pair belongs to a different protein molecule. $\mathrm{F_{nat}}$ is defined as the fraction

of heavy atom pairs in the target interface that are in the interface of the predicted model. LRMS is the ligand root mean square, computed by initially super-imposing the receptor molecule of the predicted model and the native, and then computing the root mean square of the predicted model and the native structure's ligands backbone Cα atoms. To compute the interface Root Mean Square (iRMS), first the interface contact threshold on the native structure is relaxed from 5Å to 10Å. Then the quality measure is computed by super-imposing the interface residues of both the docked model and the native structure, and computing the root mean square deviation between the interface backbone Cα atoms of the predicted model and the native structure. The three measures are then classified into four quality classes using different cut-offs as proposed by CAPRI shown in Table 1.1 [50].

Although the CAPRI criteria helps evaluate a docking model, it is a complicated approach with different thresholds for different metrics and quality categories. A better quality measure named DockQ [9] assesses the quality of a docked model using a score between 0 and 1. The lower limit zero indicates low quality and the upper limit one indicates the highest quality. This approach helps classify as well as order the protein-protein models of varying quality for a given target very easily. DOCKQ uses $\mathrm{F_{nat}}$, LRMS, iRMS to output a score. The score is calculated by scaling the LRMS and iRMS values:

$$\mathrm{RMS_{scaled}(RMS)} = \frac{1}{1 + \left(\frac{\mathrm{RMS}}{a}\right)^2}. \tag{1.1}$$

The $\mathrm{LRMS_{scaled}}$ is computed by using the LRMS value and the constant a equal to 8.5Å. Similarly, the $\mathrm{iRMSscaled}$ values is computed using the iRMS value and the constant a equal to 1.5Å. Both the a values for LRMS and iRMS are optimized and are constants [9]. The final step is to average $\mathrm{F_{nat}}$, $\mathrm{LRMS_{scaled}}$, and $\mathrm{iRMS_{scaled}}$:

$$\mathrm{DOCKQ} = \left(\frac{\mathrm{F_{nat}} + \mathrm{LRMS_{scaled}} + \mathrm{iRMS_{scaled}}}{3}\right). \tag{1.2}$$

The classification of the docked protein-protein models into the CAPRI defined categories w.r.t the DOCKQ score is shown in the Table 1.1. This opens the avenue for machine learning algorithms to learn and score each decoy in the order of its quality w.r.t the native structure.

Table 1.1: Quality categories and corresponding parameter limits proposed by CAPRI and DOCKQ

Category	CAPRI limits	DOCKQ
Incorrect	F_{nat}<0.1 or (LRMS>10.0 and iRMS $\leq$ 4.0)	DOCKQ < 0.23
Acceptable	(F_{nat} $\geq$0.1 and F_{nat}<0.3) and (LRMS$\leq$10.0 or iRMS$\leq$4.0) or (F_{nat} $\geq$0.3 and LRMS>5.0 and iRMS>2.0))	0.23 $\leq$ DOCKQ < 0.49
Medium	F_{nat} $\geq$0.3 and F_{nat}<0.5) and (LRMS $\leq$ 5.0 or iRMS $\leq$ 2.0) or (F_{nat} $\geq$0.5 and LRMS>1.0 and iRMS>1.0)	0.49 $\leq$ DOCKQ < 0.8
High	F_{nat} $\geq$0.5 and (LRMS$\leq$1.0 or iRMS$\leq$1.0)	DOCKQ $\geq$ 0.8

Chapter 2

Traditional approaches in scoring docked protein-protein models

As described in the previous chapter, scoring is the most critical and challenging step that impacts the entire docking method's results. Looking at various procedures to solve this problem, the scoring problem can be solved using four approaches: Physics-based approaches, Interface-based approaches, Knowledge-based statistical potentials, and Machine Learning methods. Interface based approaches use the features that can be calculated at the interface of the two proteins. Knowledge-based statistical approaches use existing protein databases such as Protein Data Bank to calculate statistical observations. Machine Learning-based approaches are data-driven, and there is no assumption for a specific type of scoring function that drives the affinity.

2.1 Physics-based approaches

Physics-based approaches primarily use force fields to score the protein-protein models. This section discusses briefly one such approach. In Ranking Protein-Protein Docking Results using Steered Molecular Dynamics and Potential of Mean Force Calculations, Laura J. Kingsley, Juan Esquivel-Rodríguez, Ying Yang, Daisuke Kihara, and Markus A. Lill [54] proposed a scoring technique using physics-based approaches. The authors developed a novel technique that steered molecular dynamics (sMD) and umbrella sampling to identify the near-native complexes among the protein-protein docking models. The scoring technique they proposed takes place in two steps. The first step uses sMD to sub-sample docked interactions. sMD is a technique that estimates the amount of energy required to separate the molecules in docked protein-protein models, with the intuition that near-native or natives take high energies/forces to separate the proteins and low-energies for incorrect docked protein-protein models. In the second step, the sub-sampled docked models are re-ranked using the umbrella sampling and weighted histogram analysis method to give

the final predictions. Umbrella technique is computationally more expensive and can give more accurate quantification compared to sMD. Umbrella sampling is a technique that scores each of the docked models by calculating the potential of mean force (PMF) from overlapping MD trajectories.

The data to score all the docked protein-protein docked models are generated using the ZDock [18] docking algorithm. The ZDock algorithm takes two proteins in an unbound format as input and produces 54,000 docked protein-protein models. In the first step, sMD assigns scores for the docked models, and all the models are arranged in the descending order of the scores. The top-10 high scoring docked models are selected, re-scored, and re-ordered using umbrella sampling to give the final set of predictions. To test the proposed scoring technique, the authors choose ten diverse sets of protein-protein complexes. The method described above generates the ten best-docked models and evaluated against other scoring functions such as zrank1 [6], zrank2 [6], zdock [18], irad [103], and a custom potential based on van der Waals, electrostatics and knowledge-based terms [30]. All the docked models were evaluated using the interface RMSD (iRMSD), a standard metric used by the community to evaluate the quality. The results are compared at the two steps of the process. All the scoring functions, along with the top-10 picks from sMD (first step), identified good or acceptable quality within the top-10 ranks. Compared to the other methods the results from the first step were better. Since the second step was computationally intensive, the complexes in which the results were not great were selected to evaluate the second step, which includes sMD and umbrella sampling. The second step's results placed the best near-native complex at the top of the list and thus proved to be very effective and performed the best among other scoring functions. Although sMD performed better than other scoring techniques, the disadvantage of this approach is that it is computationally very expensive and cannot perform well on complexes that are sensitive to conformational changes.

2.2 Interface-based approaches

Interface-based approaches use features derived from the interface of the docked protein-protein model to score. The features capture at the interface are most commonly the shape-

complementarity and evolutionary information. This section talks about the scoring functions based on the most common interface-based feature. In Protein-protein docking using region-based 3D Zernike descriptors by Vishwesh Venkatraman, Yifeng D Yang, Lee Sael and Daisuke Kihara [101] proposes a docking and sampling technique called LZerD (Local 3D Zernike descriptor-based Docking program) that uses 3D Zerkine Descriptors (3DZD) [78]. The primary purpose of the 3DZD is to capture the shape complimentarity of the proteins whose information can be further used to determine the docked samples and rank them. The authors' scoring function is a geometric criterion composed of three measures: Term based on normals and 3DZD vector's local shape correlation, area of overlap, and excluded volume. The common term is obtained by combining the orientation of normals at the interface and the shape correlation of 3DZD vectors that rewards or penalizes the score of an orientation of a docked protein complex. The area of overlap for a docked protein complex is estimated by the surface area buried by one of the protein molecules (usually the ligand). It can be calculated by summing up the solvent-accessible surface areas (SASA) of both the protein molecules and subtracting it from the complex's SASA. The excluded volume is the repulsive space caused due to proximity (3 Å according to the CAPRI criteria [48]) between two atoms in the docked model. The above three scoring measures combined with the weights associated with each measure are trained using a genetic algorithm to maximize the function's overall fitness.

The data to train the above Genetic Algorithm consists of 29 protein complexes from the ZDOCK benchmark datasets 0.0 [20] and 1.0 [20]. The test dataset consists of a custom 76 bound-bound docking set, and unbound complexes from the ZDock Benchmark 2.0 [74]. The docked protein models are evaluated using the interface RMSD (iRMSD), ligand RMSD (lRMSD) as proposed in the CAPRI criteria [48], and the mean of the logarithm of the first hit [88]. The other scoring techniques that the authors compared against were ZDock(PSC) [19], Context Shapes [92], and PatchDock [91]. In the test set containing 76 bound-bound cases, CS performed the best, followed by LZerD, ZDock, and PatchDock. In ZDock Benchmark 2.0 test set, ZDock and LZerD performed significantly better than CS and PatchDock. Although LZerD's performance was com-

parable with ZDock in the top-500 cut-off, it performed better than ZDock in the top 1000, and 2000 cut off.

In InterEvScore: a novel coarse-grained interface scoring function using a multi-body statistical potential coupled to evolution, Jessica Andreani, Guilhem Faure, Raphael Guerois [5] proposed a scoring technique that combines multi-body statistical potential with interface contacts derived from Multiple Sequence Alignments (MSA) and information on the evolution of homologous interfaces [4]. The authors proposed this approach addressing the limitations in scoring mechanisms that do not account for multi-body interaction and unavailability of interface contact statistics. Multi-body potentials are included in the scoring functions in the form of two-body and three-body potentials based on residue information because they encapsulate the evolutionary information. Also, the two-body and three-body states are independent of the set of decoys' analytical reference state for a complex. In two-body interaction, pairs of residues each belonging to a protein pair in the protein complex are selected. In three-body interaction, three residue interactions are chosen so that all of them do not exist in the same protein molecule. The multi-body potentials are further combined with MSAs and evolutionary information. MSAs identifies the most probable orthologs for each partner protein in a wide range of species. The evolutionary information is combined with multi-body potentials by computing MSAs for the pair of residues between the chains. The authors also describe a version of the InterEvScore wherein evolutionary information was only used for inter-residues contacts present in apolar patches. Apolar patches are said to have a high conservation of inter-residue contacts [4].

The potentials are trained using the InterEvol database [31] consisting of 1,554 interfaces. These interfaces are reduced to 1,289 interfaces by filtering with <30% sequence identity against Benchmark 4.0 [46] and further clustering the remaining interfaces with sequence identity < 40%. Testing is performed on 176 complexes of Benchmark 4.0 [46]. For each of the complex in Benchmark 4.0, ZDock 3.0 [73] is used to generate 54,000 decoys. However, the evolutionary information could only be computed for 85 complexes, and 31 complexes did not have near-native decoys generated by ZDock. Altogether, the test dataset consisted of 54 complexes and 54,000 decoys for

each of the complexes. The decoys from the ZDock algorithm are evaluated using interface Root Mean Square Deviation (iRMSD). All the near-native decoys were considered hits if Cα iRMSD is below 2.5Å. InterEvScore's performance was compared against ZDock [73], ZRank [6], and SPIDER [52]. The performance is evaluated using four metrics: Success rate defined as the number of complexes with at least one hit in top N predictions, hit rate defined as the proportion of hits among the top N predictions, integrated success rate (ISR) [103], and integrated hit rate (IHR, similar to ISR but incorporates the hit rate). In all of the performance metrics on the test set, InterEvScore performed better than the other methods.

2.3 Knowledge-based statistical potentials

Knowledge-based statistical potentials use information from known protein complexes to score the docked protein-protein models. This section talks about one such technique. In the paper, An iterative knowledge-based scoring function for protein-protein recognition, Sheng-You Huang Xiaoqin Zou [43], proposed a distance-dependent knowledge-based scoring function called ITScore-PP. The authors circumvent the limitation in many knowledge-based scoring functions that have difficulty determining the pair potentials (number of atomic densities) in reference state by proposing a novel iterative method. The calculation of pair potentials in reference state is complicated because the proteins cannot achieve zero interaction between the atoms [97]. The iterative method proposed in the paper works on the idea that the pair potentials are adjusted in every iteration until the scoring function can distinguish between the docking poses and native bound complexes.

The training data is extracted from the Protein Data Bank [12] and filtered for dimeric structures, a minimum number of 10 residues per chain, at least 30 interacting residues (heavy atoms within 4.5Å), similar complexes with >70% sequence identity. The final filtered training set consists of 851 protein-protein complexes. For each of the complex in training set ZDock 2.1 [19], it was used to generate decoys. Testing of ITScore-PP for scoring decoys was done on three databases. The four databases are: benchmark by Weng and coworkers [21], ZDOCK decoy set [19, 18], and RosettaDock unbound perturbation decoy set [41]. The decoys' quality in train or

test set is assessed using the CAPRI competitions criteria [50, 71]. The success rate was used as a metric to compare different scoring functions. In the first test set, ITScore-PP performed better than ZDock 2.1 and ZDock 2.3 in both bound scoring (differentiating between the bound complex and its decoys) and unbound scoring (identifying the best decoys among a given set). In the test dataset ZDOCK 2.1 decoy set, the scoring functions ZDOCK 2.1, RDOCK [67], and ITScore-PP were compared, and the results show RDOCK performed the best, followed by ITScore-PP and ZDOCK 2.1. In the test dataset ZDOCK 2.3 decoy set, the scoring functions ZDOCK 2.3, RDOCK, ITSCore-PP, and EMPIRE were compared, and the results show EMPIRE [69] performing the best followed by ITScore-PP, RDOCK, and ZDOCK 2.3. In the Rosetta dock decoy test set, ITScore-PP and Rosetta Dock are compared, with Rosetta Dock having a higher success rate in the top-5 predictions and ITScore-PP dominating in all the predictions after top-5. Overall, we can see that knowledge-based statistical potential are comparable or better than other methods.

2.4 Machine learning based approaches

Unlike the above three methods: Physics-based, Interface-based, and Knowledge-based statistical potentials that are direct approaches meaning the features are mapped to scores using known functions, Machine learning based approaches are indirect where the function that maps the features to the scores is unknown and derived from the data. This section describes one such machine learning approach. In Finding correct protein–protein docking models using ProQDock, Sankar Basu, Björn Wallner [10], proposed a machine learning based scoring technique that uses Support Vector Machines (SVM) [11] to predict quality scores using the features computed. Thirteen features are derived from each of the docked protein-protein model that are based on structural information of the protein-protein interfaces, and the structural quality and integrity of the proteins in the protein complex. ProQDock uses a training set containing 73792 docked protein-protein models from CAPRI score set [65] and MOAL [75]. The test set consists of 25985 docked protein-protein models generated using SwarmDock [98] from 55 new complexes added to Benchmark 5 dataset [105]. In both the training and testing set, the quality of the docked protein-protein model

is measured using the DOCKQ score [9] and is used as the label for training. The performance of the ProQDock algorithm is compared against ZRANK [6], ZRANK2 [83] by computing the ROC curves, and the number of docked protein-protein models in 1, 5, 10, and 100 ranks that have acceptable or better quality. ProQDock performed the best in both the evaluation metrics. The ROC-AUC of ProQDock, ZRANK, and ZRANK2 are 0.87, 0.75, and 0.71 respectively, showing that ProQDock performs significantly better. ProQDock shows that combining features high-level features i.e., features at protein-level with machine learning algorithms helps in identifying the good quality docking models better than any of the discussed methods.

Machine learning algorithms come in different forms from simple linear regression to complex Artificial Neural Network. Various approaches in Machine learning have been applied to Scoring docked protein-protein models, most notably Artificial Neural Networks. The scoring methods that use Artificial Neural Networks will be discussed in Chapter 4.

Chapter 3

Artificial Neural Networks

Artificial Neural Networks are loosely modeled by biological neurons' connectivity and impulse action in various organisms' nervous systems. A biological neuron comprises distinct parts: dendrites, soma (cell body), and axon. The dendrites receive information from other biological neurons, pass the information to soma, accumulate information, and finally propagate to other biological neurons through the axon. Electrical signals, a.k.a action potential, usually transmit the information among biological neurons. The communication between biological neurons occurs when the dendrite of a biological neuron latches onto the axon of another biological neuron. This junction between two biological neurons is called a synapse. Figure 3.1 shows a biological neuron with n number of input synapses at the dendrites and m number of output synapses at the axon terminals. The biological neuron processes the input signals that arrive through the dendrites to emit the output signals through the axon terminals. The processing of the input signals occurs at the neurons' soma and myelinated axon trunk, which can inhibit or excite the signal strength. This interconnectivity between neurons creates a complex network of interactions that can perform a broad spectrum of actions for different organisms.

Inspired by the biological neuron mechanism, a simple artificial neuron was designed using mathematics. Similar to how a biological neuron communicates using electrical signals, an artificial neuron communicates using mathematical signals. It takes mathematical input from other artificial neurons, combines, and passes to other artificial neurons. In simple mathematical terms an artificial neuron can be represented by a function f in the equation $y = f(x_1, \ldots, x_n|\Theta)$ where $x_1, \ldots, x_n$ are the inputs, y is the output, and Θ is the complete set of parameters. An artificial neuron maps the inputs to the outputs using three basic steps. The first step is to determine the importance of each input by scaling the input. The scaling variable for an input x_i is denoted by w_i and is referred to as weight. The second step aggregates all the scaled inputs, and the result is called the signal. Finally, the third step applies a non-linear function σ, a.k.a activation function,

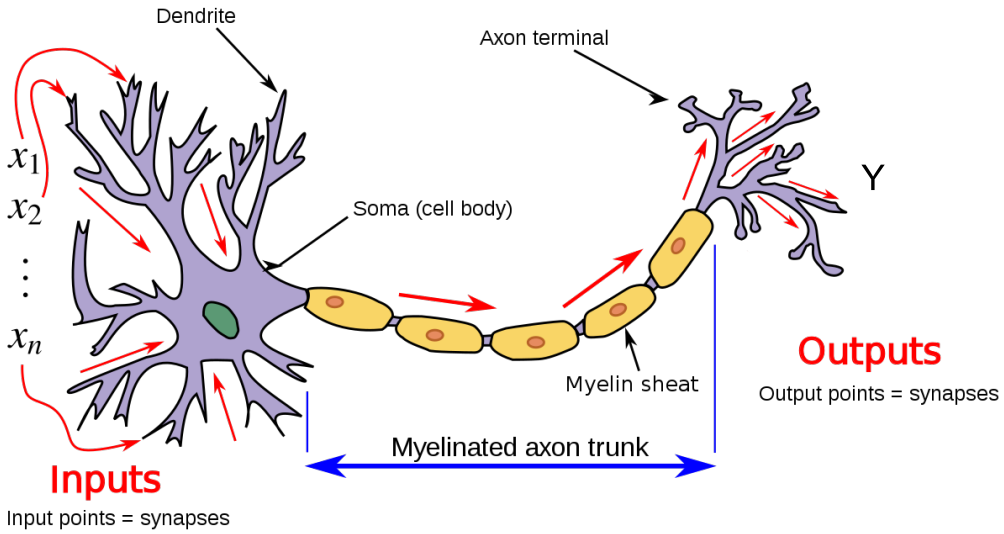


Figure 3.1: Structure of the Biological neuron showing the dendrites, soma, and axon. Attached to the axon is a myelin sheath that acts as an insulator and increases the electrical signal's speed of transmission. Mathematically, the biological neuron can be represented by a function f that transforms the inputs $x_1, \ldots, x_n$ to y. It is under a Creative Commons Attribution-Share Alike 3.0 Unported license. File: https://commons.wi kimedia.org/wiki/File:Neuron3.png#/media/File:Neuron3.svg, License: https://creativecom mons.org/licenses/by-sa/3.0/deed.en

to the output of the second step to get the final outputs called the activation. The activation function's role is to mimic the biological neuron's excitation and inhibition of electrical signals. A bias variable represented using b is added in the second step most of the time to help shift the activation function. The above three steps can be explained using the following equation:

$$y = \sigma\left(\sum_{i=1}^{n} w_i x_i + b\right), \tag{3.1}$$

where y is the output signal from the neuron, x_i is one input signal among among n input signals, σ is the activation function, w_i is the parameter to learn the weight of each input connection, and b is the bias parameter.

Another inspiration from biological neurons is the communication between them that forms an intricate network. This is mimicked in artificial neurons by propagating mathematical signals between artificial neurons in a network. This network of artificial neurons is commonly referred

to as the Artificial Neural Network (ANN). Inspired by biological neurons' arrangement, many architectures of Artificial Neural Networks were designed, each for a specific need and purpose. Further in this thesis, as common parlance in machine learning literature, neural network is used as a replacement for Artificial Neural Network. The term "biological neural network" will be mentioned explicitly.

Among the neural network architectures, the most common and simplest is the Feed Forward Neural Network (FFNN) illustrated in Figure 3.2. FFNN are acyclic, meaning the network doesn't have any cycles. The artificial neurons in an FFNN layer take input from the previous layers, perform mathematical operations on the input and provide output to the subsequent layer as shown in Figure 3.2. Using the FFNN, we introduce the notations of neural network. Generally, any neural network comprises two or more layers where each layer encompasses one or more neurons, also known as units.

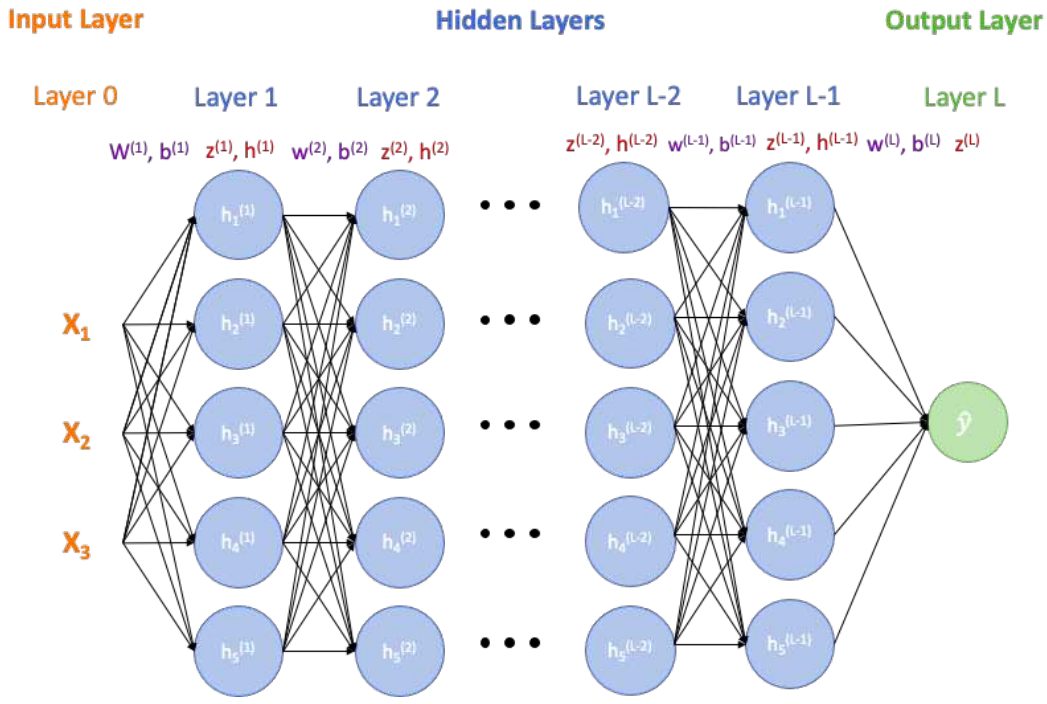


Figure 3.2: A feed forward neural network (FFNN) with L layers. The diagram shows three fundamental layers: input layer in orange color, hidden layers in blue color, and output layer in green color. x is the input. For each hidden layer l, $w^{(l)}$ and $b^{(l)}$ are the learnable parameters, $z^{(l)}$ and $a^{(l)}$ are the intermediate values. The intermediate outputs for unit i and hidden layer l can be identified using $z_i^{(l)}$ $a_i^{(l)}$. The output is $\hat{y}$.

Any neural network architecture comprises three fundamental layers: an input layer, hidden layer, and output layer. These layers are indexed from layer 0 to L, with 0 being the input layer, L being the output layer and 1 to L-1 called the hidden layers. Therefore, a given neural network with L layers has L-1 hidden layers. The hidden layers' learn representations that map the inputs to correct outputs. In FFNN, all hidden layers use dense layer. In the dense layer, each unit receives information from all units in the previous layer and sends information to all the units in the next layer. The computation of signal and activation in a dense layer l is shown below:

$$\mathbf{Z}^{(l)} = \mathbf{W}^{(l)} h^{(l-1)} + \mathbf{b}^{(l)} \tag{3.2}$$

$$h^{(l)} = g^{(l)} \left(\mathbf{Z}^{(l)} \right), \tag{3.3}$$

where $h^{(l-1)}$ is the activation from the layer $(l-1)$, $\mathbf{W}^{(l)}$ is the learning parameter weight matrix for the layer (l) whose dimension is $n^{(l)} \times n^{(l-1)}$, $\mathbf{b}^{(l)}$ is the bias vector for the layer l whose dimension is $n^{(l)}$, $\mathbf{Z}^{(l)}$ is the signal for the layer l, $g^{(l)}$ is the activation function for the layer l. The most common activation functions include the logistic function: $g(x) = 1/(1+\exp^{-x})$, hyperbolic tangent: $g(x) = \tanh x$, and ReLu: $g(x) = max(0, x)$. The computation of the output of a layer and passing it to the subsequent layer in an ANN is called forward propagation.

Although the design of ANNs is conceptually simple, the challenge is to find good set of parameters that model a continuous function with minimal error. The ideal set of parameters are determined using an algorithm called gradient descent [89] that involves three steps: computing the error using loss functions, gradient w.r.t each parameter, and updating the parameters. The error between the ground truth values a.k.a labels ($Y = \{y_1, \ldots, y_m\}$) and the predicted values ($\hat{Y} = h^{(l)} = \{\hat{y}_1, \ldots, \hat{y}_m\}$) is calculated using a loss function $L(\hat{Y}, Y)$. A variety of loss functions are available depending on the type of problem. For example, regression problems use the L1 and L2 loss functions, classification problems use cross-entropy loss function [40]. The gradient w.r.t each of the parameter is calculated by computing the derivative of loss function w.r.t to the parameter ($\mathbf{dW}^{(l)}, \mathbf{db}^{(l)}$). Finally, the parameter is updated by taking a step in the direction of the

gradient and a hyper-parameter called learning rate (α) influences the step size by linearly scaling it.

Algorithm 1: Gradient descent

Input: $\mathbf{X}$, Label: $\mathbf{Y}$, Parameters: $\mathbf{\Theta}$, Number of epochs: t

Repeat for t epochs {

Compute predictions: $\hat{\mathbf{Y}} = f(\mathbf{X}, \mathbf{\Theta})$

Compute Loss: $J(\Theta) = L(\hat{\mathbf{Y}}, \mathbf{Y}) = \frac{1}{m} \sum_{i=1}^{m} L(y_i, y_i)$

Compute gradients: $d\mathbf{W}^{(l)} = \frac{\partial J}{\partial \mathbf{W}_i}$, $d\mathbf{b}^{(1)} = \frac{\partial J}{\partial \mathbf{b_i}}$

Take a step: $\mathbf{W}^{(l)} = \mathbf{W}^{(l)} - \alpha d\mathbf{W}^{(l)}$, $\mathbf{b}^{(l)} = \mathbf{b}^{(l)} - \alpha d\mathbf{b}^{(l)}$

}

Gradient descent is available in three different variants depending on the number of examples used to compute the gradients of the loss function: Batch gradient descent (GD), stochastic gradient descent, and mini-batch gradient descent. Batch gradient descent computes the gradient of the loss function by considering the entire training dataset. The disadvantage with batch gradient descent is that convergence is slow and also the whole algorithm needs to be retrained if new examples are added to the training set. In stochastic gradient descent (SGD), gradient of the loss function is computed for each of the training example. Although SGD is fast the disadvantage is that the convergence is unstable. Mini-batch gradient descent takes the best of both batch gradient descent and SGD by computing the gradients of the loss functions for a subset of the training data. This stabilizes the convergence because the parameter updates are of low variance.

In fitting ANNs, backpropagation [90] is an efficient method to compute gradients of loss function w.r.t the learning parameter. The training of a neural network takes place in two steps, forward propagation, and backpropagation. Forward propagation takes input data and outputs the predictions, while backward propagation computes the gradient of the loss function w.r.t each of the parameter and updates the parameters. Initialization of the learning parameters, weight, and bias is done by drawing numbers from random distributions based on empirical or theoretical justification

[38]. Training takes place in epochs, where an epoch is a complete pass through the entire dataset. The neural network is trained until the loss function is optimized or until a fixed number of epochs.

Neural networks are prone to overfitting. Overfitting occurs when a neural network fails to generalize to unseen data. A neural network starts to overfit when it is overtrained, i.e., training beyond achieving the ideal set of parameters or using more than the sufficient number of parameters [87, 23]. To avoid the risk of overfitting many regularization techniques are used including Dropout [93], early stopping [76], and weight decay [56].

A neural network can be trained in a supervised or unsupervised fashion [39]. In supervised learning, a neural network is trained to correct itself to predict the labels provided. In contrast, in unsupervised learning, a neural network learns the probablity distribution of the dataset since the labels are not provided. This allows unsupervised algorithms to synthesize or denoise data [39]. Training neural networks for complex tasks demands sophisticated hardware such as Graphics Processing Units (GPUs) or the new Tensor Processing Units (TPUs). With its increased accessibility, neural network architectures are being designed with many hidden layers and hidden units. These multi-layered neural network models are termed Deep Learning models by the machine learning community. In the last decade, most of the advancements in neural networks are due to image and speech analysis developments. One such class of neural network that helped significantly is the Convolutional Neural Network (CNN) which is explained in the following section.

3.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a class of deep neural networks primarily designed to solve problems related to visual image data but with its success extended to different avenues such as speech and text [39]. CNNs use representations from local region of the inputs to make predictions. The idea to learn and predict using local regions was inspired by a biological experiment where a neuron responds to very small regions of the visual field and the visual cortex [45]. This biological experiment paved the way for the initial CNN algorithm called neocognitron [34] which introduced two different layers in CNNs called convolutional layer and downsampling layer.

In the convolutional layer, a kernel containing a set of units is applied to the receptive field of the inputs to output the activations. Applying the same kernal with different sets of weights for each pass helps in learning more useful representations. The set of weights used by the kernel for each pass through the receptive field is commonly referred to as the filter. One of the downsampling layer is the average pooling layer where the activations in each of the receptive fields are averaged. Inspired by the neocognitron downsampling layer, another method called cresceptron [110] was proposed, which introduced the max-pooling layer. Max-pooling performs downsampling by grabbing the maximum activation in the receptive field. Later on, more sophisticated and efficient convolution and pooling operations were proposed to tackle the challenging tasks of achieving shift-invariance and low computational requirements [106, 60, 61, 96].

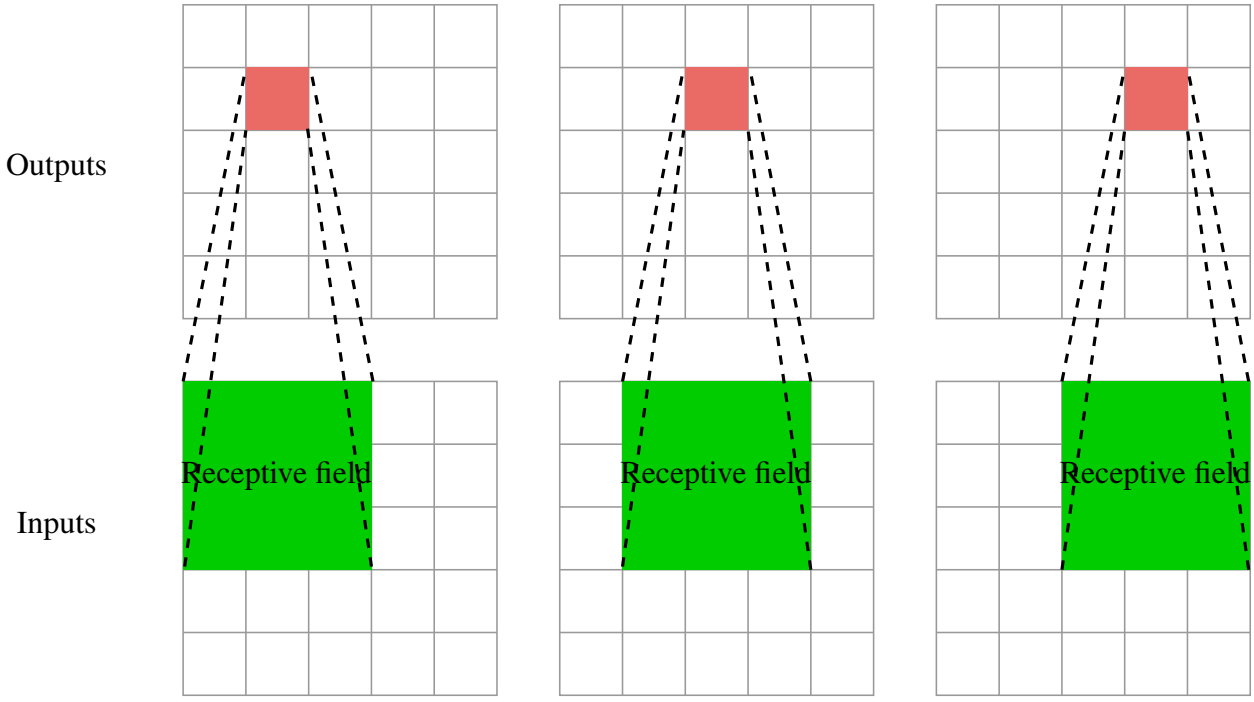


Figure 3.3: Application of strided convolution where a filter of the size of the receptive field 3×3 is applied on the input to get an output unit. All output units are computed by sliding the receptive fields and performing the convolution operation.

Most of the CNN architectures use two major building blocks: The convolutional layer and the pooling layer. A simple convolutional layer performs strided convolution using one or more filters

on an input to produce the output as shown in the Figure 3.3. A strided convolution is a convolution operation where the kernel size is smaller than the image and the kernel slides throughout the image. This type of convolution brings three key features: sparse interactions, parameter sharing, and equivariant representations [39]. Unlike traditional neural networks, where every input unit connects to every output unit, the sparse representation of connectivity uses a kernel smaller than the input to reduce the number of connections. This allows convolution to learn features in local regions, and indirectly interact with other regions when multiple convolutions are applied. With parameter sharing, instead of learning a set of parameters for every location, only one set of parameters is used that can perform multiple functions. Equivariant representation is a consequence of the kernels parameter sharing; where if there's a change in the input, the output also changes. This can be mathematically explained using a function $f(x)$ that's said to be equivariant if $f(g(x)) = g(f(x))$, where $g(x)$ is a function that translates the input. Convolution operation is equivariant to linear translational operations but operations that distort the input by scaling and rotating are not equivariant.

Pooling is another important layer in the CNN. The pooling layer's primary purpose is to reduce the dimension of the input or hidden representations, thereby decreasing the number of parameters in the following layers. The most common pooling operations are average pooling and max pooling. Average pooling works by sliding a kernel-like window over the hidden representations and averaging all the activations inside the window. Similarly, in max-pooling, the maximum activation among the activations in a window is passed as output. Averaging the activations or picking the maximum activation helps summarize the input or hidden representations to the following layers and make the neural network robust to translations. Thus, helping the network learn and make predictions based on invariant information rather than learning position-specific features from the convolution.

3.2 Graph Neural Network

In the past decade, most of the neural networks were designed to operate on data that are grid-like structures. Although grid structures like image, speech, or text can accurately capture relevant information, they may not be the best to encapsulate data from irregular domains such as social networks, knowledge graphs, or biological networks [59, 42, 32]. To solve this problem, a data structure called a graph capable of modeling the information of objects (nodes) and their relationships (edges) is used. Graph data structures are successful because of their power to express information. Graph Neural Networks (GNNs) is all about applying neural networks to the non-euclidean domain of graphs. Some of the common tasks of graph neural networks are to classify nodes, predict links, cluster data or classify entire graphs.

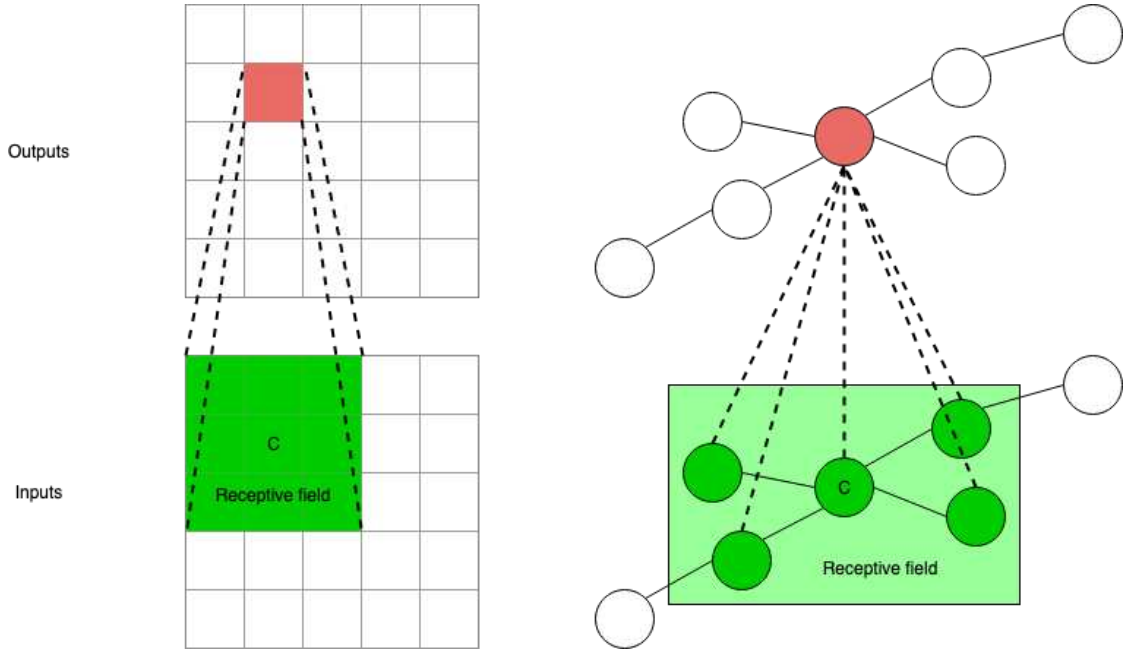


Figure 3.4: Comparision of the convolution operation between the CNN on the left and the GCN on the right. The CNN aggregates all the hidden representation surrounding a central pixel C in a receptive field shown in green to output the final representation of the central pixel C show in red color. Similarly, the GCN aggregates the hidden representations of the neighboring nodes around a central node to output the central nodes final representation shown in red color.

Graph convolution network (GCN) is one type of GNN inspired by CNNs discussed in Section 3.1. Graph Convolution approaches initially proposed by Duvenaud et al. [28] apply convolution

on a node and spatially nearby neighbor nodes of a graph. Similar to how the convolution operation in CNN aggregates information from the surrounding pixels around a pixel, the Spatial GCN aggregates information from the neighboring nodes around a central node. Figure 3.4 compares the convolution operation in CNN and GCN. Although most commonly GCNs are customized according to the problem, the general mathematical notation is shown in the equation below:

$$h_i^{(l)} = \sigma\left(\mathbf{W}^{(l)} h_i^{(l-1)} + \frac{1}{|N_i|}\sum_{j \epsilon N_i} \mathbf{W}^{(l)} h_j^{(l-1)} + b^{(l)}\right), \tag{3.4}$$

where for the layer l, $\mathbf{W}^{(l)}$ is the weight matrix, $b^{(l)}$ is the bias vector, $h_i^{(l)}$ is the hidden node embedding of node i, N_i is the set of neighbors of node i, and σ is the activation function. From here on Spatial GCN will be referred to as GCNs.

Graph Attention networks (GAT) are an extension of GCNs where the neighbor nodes' participation w.r.t the central node is weighted. In GCN, every central node attends its neighbor nodes with equal importance, whereas GAT gives more importance to the relevant neighbor nodes. GAT introduced by Veličković et al. [100] determines the neighbors' importance using one of the most successful strategies in sequence-based tasks called self-attention [99]. Self-attention is based on attention mechanism [7, 36] that computes the importance of each of the input w.r.t other inputs to generate its new representation. The importance is calculated by doing a cross-product of the inputs w.r.t to itself and then applying softmax to get the probabilities of each input w.r.t other inputs. This importance, a.k.a attention coefficient e_{ij} of each of the central nodes w.r.t its neighboring nodes in a graph is calculated by performing a shared attentional mechanism a:

$$e_{ij} = a(\mathbf{W}h_i, \mathbf{W}h_j), \tag{3.5}$$

where a is the function that computes dot product, $\mathbf{W}$ is the shared weight matrix. The attention coefficients around a central node are further normalized using a softmax function to make the attention coefficients comparable:

$$\alpha_{ij} = \text{softmax}_j\left(e_{ij}\right) = \frac{\exp\left(e_{ij}\right)}{\sum_{k \epsilon N_i} \exp\left(e_{ik}\right)}, \tag{3.6}$$

Finally, all the nodes hidden features are updated using the normalized attention coefficients shown below:

$$h_i^{(l)} = \sigma\left(\mathbf{W}^{(l)} h_i^{(l-1)} + \frac{1}{|N_i|} \sum_{j \epsilon N_i} \alpha_{ij} \mathbf{W}^{(l)} h_j^{(l-1)} + b^{(l)}\right), \tag{3.7}$$

where the notation is as explained in equation (3.4). The GAT can be applied on graphs of different sizes and structure because the importances are calculated within a variable neighborhood size. Furthermore, this implementation is computationally efficient because it can be parallelized for each node.

Chapter 4

Application of Neural Networks in scoring docked protein-protein models

Scoring using machine learning methods construct the functional forms by mapping the protein-protein complexes' structural features to their binding affinities. This makes it a different approach compared to other classical scoring mechanisms discussed in Chapter 2. Results show machine learning methods consistently outperform the classical scoring algorithms on a diverse set of protein-protein complexes [109, 17].

In protein docking model evaluation by 3D deep convolutional neural networks [109], Xiao Wang, Genki Terashi, Charles W Christoffer, Mengmeng Zhu, Daisuke Kihara proposed a scoring technique that uses 3D convolutional networks. This paper's main idea is to represent the interface of protein docking models as a 3D grid made up of voxels and input it into the 3D CNN to distinguish the incorrect decoys from the correct ones. The input data is a cube of dimension $20 \times 20 \times 20$ Å or $40 \times 40 \times 40$ Å centered on the interface of the chains in a docked protein model. The authors define the interface as the set of heavy atoms that are at a distance of 10 Å or less between the chains of a docked protein model. The cube placed at the interface is filled with smaller cubes called voxels of either $1 \times 1 \times 1$ Å for a cube of side 20 Å or $2 \times 2 \times 2$ Å for a cube of side 40 Å. These voxels encapsulate the number of atoms of a specific atom type, such as carbon, oxygen, nitrogen, and other atoms, separately at the interface. In addition to the atom types, two other features, GOAP [115] and ITScore [43] encapsulates the energy contributions from different atoms at the interface of the docked protein model. The neural network architecture consists of two 3D Convolutional layers, two fully connected layers, and a final output layer that uses a sigmoid function to compute the input docked protein model's probability of being a good quality model. Each of the 3D convolutional layers applies a filter of 3*3*3 on the output from the previous layer, with the first convolution using 100 filters followed by a max-pooling layer and the

second convolution using 200 filters followed by a max-pooling layer. The authors train the neural network with cross-entropy as the loss function, and Nadam [27] as the optimizer. Regularization techniques, such as Dropout and L2 regularization, are used to reduce overfitting.

The authors use 178 protein-protein complexes that have, on average, 54,000 docked models per complex from ZDock Docking Benchmark Dataset 4.0 [46] to train the neural network. All the docked protein models are classified into correct and incorrect docked models using the lRMSD, iRMSD, and FNAT thresholds defined in the CAPRI criteria [66]. Each of the correct docked protein models in the training set is augmented by 24 rotations. The training set is divided into four groups using TM-score [114] and trained using 4-fold cross-validation. The trained neural network model is tested on the DockGround benchmark dataset [70]. The model's performance on the test set is compared against other scoring techniques such as GOAP, ITScore, ZRANK [6], ZRANK 2.0 [83]. Hit Rate, defined as the fraction of complexes for which the scoring technique was able to rank a correct decoy in top-p ranks, is used as a metric to evaluate and compare scoring functions' performance. The results show that the authors proposed a model (Dove) to perform the best, followed by irad, ZRANK2, ZRANK, ITScore, and GOAP.

In Energy-based Graph Convolutional Networks for Scoring Protein Docking Models (EGCN) [17], Yue Cao and Yang Shen proposed a graph convolution technique that learns interaction energies of 3D protein-protein complexes to score the docked protein models. All the docked protein models are represented using graphs with residues as nodes and the edges as inter or intramolecular residue contacts. Four features are used to represent a node where the first three represent the side-chain pseudo atoms charge, distance to α-carbon atom, and nonbonded radii. The fourth feature is the solvent-accessible surface area (SASA). Each edge is represented using 11 features indicating the pairwise distances between a combination of 6 different atom types between two residues. An edge is defined between two atoms in different residues if the distance is within 12Å. The energy-based graph convolution (EGC) layer models the binding energy. In EGC, a learnable weight is used for every combination of node and edge feature and is shared across all the atoms in a docked model. The EGC layer computes hidden node representations for each

of the proteins by combining the information of a protein and other proteins using a convolution operation. The convolution operation aggregates the hidden node information across all node features, edge features, and the neighboring nodes around a central node scaled up by inverse of the distance between the nodes. The neural network architecture proposed by the authors consists of three EGC layers, multi-head attention, and three fully connected layers. The neural network is trained with binding affinities as the label, mean square error as the loss function, and optimized using the Adam optimizer.

The neural network model is trained and validated on the 50 protein-protein complexes from the protein benchmark 4.0 [46] and is split in the ratio 4:1, respectively. For each protein complex in the training set, docked models are generated using the ClusPro docking software. The neural network models are finally evaluated on three test sets: remaining 107 protein-protein complexes from the benchmark 4.0 [46], 14 previous CAPRI targets, CAPRI Score_set [65]. All docked protein models are evaluated for ground truth quality as acceptable using the CAPRI criteria of using only IRMSD < 4Å. The performance of EGCN was compared against other scoring mechanisms such as iRAD [103] and RF [16]. Enrichment Factor (EF), defined as the number of acceptable docked protein models in top p% ranked by a scoring function divided by that of the random order, is used to evaluate and compare the performance of EGCN against other scoring functions. In the first and second test sets, EGCN performs better, followed by RF and iRAD. In the third test set, EGCN and iRAD are comparable, whereas RF performed a little worse than both. Overall, we can see that Machine Learning based scoring functions perform comparable to or better than other classical scoring functions. The work proposed in this thesis neither uses the existing approaches nor improves the existing methods in scoring docked protein-protein models. Instead we take a unique approach to represent data and apply neural network techniques. Most of the existing approaches do not use raw features, instead they work on high-level features that are feature engineered. Also, existing methods use small datasets that may not help learn a wide distribution of protein-protein complex structures. Finally, most of existing approaches do not use state-of-the-art neural network

techniques. In this thesis, we work with data that uses raw features, a very large dataset, and the proposed neural-network techniques use state-of-the-art techniques in deep learning.

Chapter 5

Proposed Neural Network techniques

Graphs neural networks are now commonly used to predict properties of molecular structures. 3D CNNs are an alternative deep learning approach [109, 24]. 3D CNNs, as explained in the previous chapter, learn from the embedding of the molecule in a 3D grid to perform various tasks. Although 3D CNNs were applied on molecular structure data and demonstrated good results [109, 24], there are limitations to this technique. First, 3D CNNs are not invariant to the orientation of its input molecule. Therefore, data augmentation needs to be performed by rotating the input. Second, CNNs typically need many layers to learn the structure of the data, which demands many parameters and a long training time. In comparison, graph neural networks use a powerful graph data structure that provides flexibility in incorporating domain knowledge, designing GNN layers, and model interpretability. Molecular graphs where edges represent distances between atoms are invariant to rotations, making GNNs invariant by construction. Furthermore, GNNs are usually known to give good results using three to four layers [111, 116], therefore requiring shorter training time. Although the disadvantage of representing proteins as a graph is that there could be a loss of spatial information because the captured graph is an abstract representation of the protein. In this thesis, we would like to see if the advantages of GNNs outweigh the disadvantages. Before discussing the GNNs used to solve this problem, we will first discuss we represent docked protein-protein models using graphs.

5.1 Docked protein-protein models as a Graph

Proteins can be represented at multiple levels: atom, residue, or the protein itself. A protein can be represented in mathematical terms by an undirected, unweighted graph G with a set of nodes, $V = \{v_1, v_2, \ldots, v_n\}$ and a set of edges, $E = \{e_1, e_2, \ldots, e_m\}$, where any two nodes are connected by at most one edge. Borrowing ideas from other papers [109, 32] this thesis constructs the graph of the docked protein-protein model at the atomic level. Our primary motivation for this decision

is to see if raw features at the atom level are sufficient for neural networks to model protein-protein interactions, removing the need for feature engineering. When extracting the graph at the atomic level from the docked protein-protein model, we only consider the interface between the proteins instead of considering all the atoms. This is because the interface is pivotal for the protein-protein interaction, as explained in chapter 1. We define the interface as shown in the Figure 5.1. All residues that are a distance of at most 10Å between chains are extracted, and the atoms in these residues are called interface atoms. Similarly, all residues that are at a distance of at most 10Å from the interface atoms in the same chain are extracted, and the atoms in these residues are called neighboring atoms.

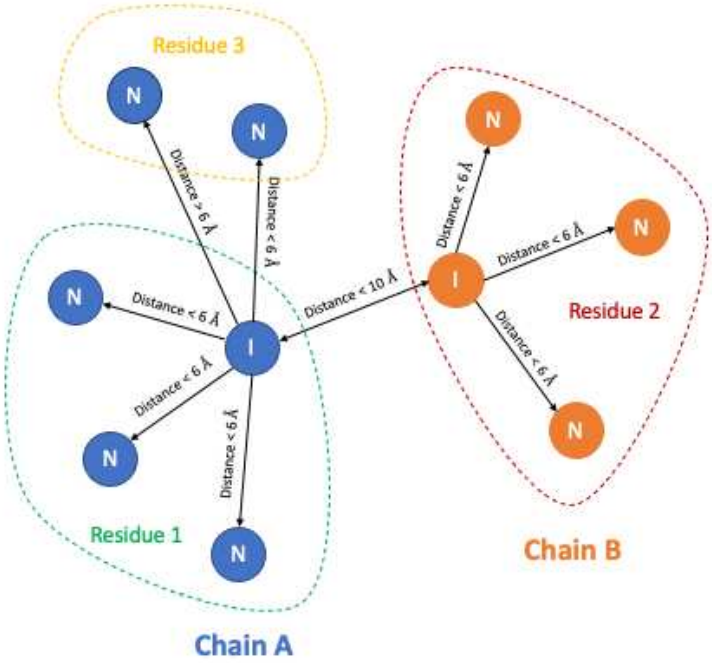


Figure 5.1: Graph extracted from the interface of the protein complex. The protein complex consists of two chains A and B, shown in blue and orange. The atoms belonging to residues 1, 2, and 3 are circled in green, red, and yellow. The atoms represent the nodes of the graph, and the edges between the nodes show the distance. All interface atoms are labeled as I, and the neighboring atoms are labeled as N.

Altogether, in our Graph G extracted from the docked protein-protein model, the node captures the feature vectors associated with the atom, and the edges are weighted according to distance. The node vector is a one-hot encoding of the 11 different atom types borrowed from work on ranking

proteins [24]. The 11 different atoms types are shown in the Table 5.1. The edge weight captures the distances as follows:

$$E_{ij} = e^{-\frac{1}{2}\left(\frac{d}{\sigma}\right)^2}. \tag{5.1}$$

The abstract representation of the docked protein-protein model using graphs brings several ad-

Figure 5.2: Interface extracted from a docked protein-protein model (model.000.00.pdb) generated using ClusPRO docking software from the unbound proteins of the complex 1ACB. The atoms from the entire docked protein-protein model are shown in red color. The extracted interface is shown using yellow and green colors with yellow color representing the interfacing atoms and the green color representing the neighboring atoms of the interfacing atoms.

vantages while solving this problem. Firstly, the change in coordinate system does not influence the graph representation. Secondly, neither the node nor edge features are tied to the coordinate system. Although the edge feature - distance, is calculated using the coordinate system, it is a scalar and is independent of the coordinate system. Therefore, the graphs' rich information about the atoms involved and their local neighborhood helps the convolution operations in graph neural networks leverage this information to solve this thesis's problem. An example of an interface graph extracted from a protein-protein interaction model is shown in Figure 5.2.

Table 5.1: Atom types used in constructing node features.

Type	Description	Atoms
1	Sulfur/selenium	CYS:SG, MET:SD, MSE:SE
2	Nitrogen (amide)	ASN:ND2, GLN:NE2, backbone N (including N-terminal)
3	Nitrogen (aromatic)	HIS:ND1/NE1, TRP:NE1
4	Nitrogen (guanidinium)	ARG:NE/NH*
5	Nitrogen (ammonium)	LYS:NZ
6	Oxygen (carbonyl)	ASN:OD1, GLN:OE1, backbone O (except C-terminal)
7	Oxygen (hydroxyl)	SER:OG, THR:OG1, TYR:OH
8	Oxygen (carboxyl)	ASP:OD*, GLU:OE*, C-terminal O, C-terminal OXT
9	Carbon (sp2)	ARG:CZ, ASN:CG, ASP:CG, GLN:CD, GLU:CD, backbone C
10	Carbon (aromatic)	HIS:CG/CD2/CE1, PHE:CG/CD*/CE*/CZ, TRP:CG/CD*/CE*/CZ*/CH2, TYR:CG/CD*/CE*/CZ
11	Carbon (sp3)	ALA:CB, ARG:CB/CG/CD, ASN:CB, ASP:CB, CYS:CB, GLN:CB/CG, GLU:CB/CG, HIS:CB, ILE:CB/CG*/CD1, LEU:CB/CG/CD*, LYS:CB/CG/CD/CE, MET:CB/CG/CE, MSE:CB/CG/CE, PHE:CB, PRO:CB/CG/CD, SER:CB, THR:CB/CG2, TRP:CB, TYR:CB, VAL:CB/CG*, backbone CA

5.2 Proposed Graph Convolutional Networks

The simple idea of graph convolution from Equation (3.4) can be improved further in multiple ways. One way to learn better representations is by distinguishing the central node from the neighboring nodes using different weight matrices, one for the central node and the other for the neighboring nodes. This distinction can be combined with the domain knowledge of protein-protein interactions that the neighboring atoms closest to the central atom contribute a lot to the central atom. The importance of a neighbor can be incorporated into the GCN by scaling each of the neighbors' hidden representations by the edge weight between the central node and the neighboring node, which reflects distance. These ideas are incorporated in the following:

$$h_i^{(l)} = \sigma\left(\mathbf{W}_c^{(l)} h_i^{(l-1)} + \frac{1}{|N_i|}\sum_{j \epsilon N_i} \mathbf{W}_n^{(l)} h_j^{(l-1)} \bigodot E_{ij} + b^{(l)}\right), \tag{5.2}$$

where for the layer l, $\mathbf{W}_c^{(l)}$ and $\mathbf{W}_n^{(l)}$ are the weight matrix for learning representations w.r.t central node and neighboring nodes respectively, $b^{(l)}$ is the bias vector, $h_i^{(l)}$ is the hidden node embedding of node i, N_i is the set of neighbors of node i, and σ is the activation function. E_{ij} is the value in the edge matrix between node i and node j. From now on in this thesis, equation (5.2) will be referred to as baseline GCN or bGCN.

While finding the set of neighboring nodes in bGCN, we account for both interface and neighboring atoms and apply the same filters without distinction. Instead, the graph convolution could potentially learn better representations by applying different sets of filters to the nodes in the interface and protein graphs as shown in the equation below:

$$\begin{aligned} h_i^{(l)} = \sigma\Bigg(&\mathbf{W}_c^{(l)} h_i^{(l-1)} + \frac{1}{|N_i^{(int)}|}\sum_{j \epsilon N_i^{(int)}} \mathbf{W}_{int}^{(l)} h_j^{(l-1)} \bigodot E_{ij} \\ &+ \frac{1}{|N_i^{(prot)}|}\sum_{j \epsilon N_i^{(prot)}} \mathbf{W}_{prot}^{(l)} h_j^{(l-1)} \bigodot E_{ij} + b^{(l)}\Bigg) \end{aligned} \tag{5.3}$$

where, $N_i^{(int)}$ is the number of atoms surrounding node i in the interface graph, $N_i^{(prot)}$ is the number of atoms surrounding node i in the protein graph, the filter $\mathbf{W}_n^{(l)}$ in (5.3) is replaced with two filters $\mathbf{W}_{int}^{(l)}$ and $\mathbf{W}_{prot}^{(l)}$ for nodes in the interface and the protein graph respectively. The architecture defined by (5.3) will be referred to as GCN from now on.

5.3 Proposed Graph Attention Networks

The idea of differentiating the neighbor nodes in the interface and protein graphs is extended to Graph Attention Networks where two set of attention coefficients are computed: one w.r.t the interface graph and the other w.r.t the protein graph. Each of the sets of the scaled attention coefficients are further scaled using the edges and normalized using the softmax function as shown below:

$$\alpha_{ij,int}^{(l)} = \frac{\exp\left(\text{LeakyReLU}\left(a_{int}^{(l)}\left[\mathbf{W}_{int}^{(l)}h_i^{(l-1)} \parallel \mathbf{W}_{int}^{(l)}h_j^{(l-1)}\right] \bigodot E_{ij}\right)\right)}{\sum_{k\epsilon N_i^{(int)}} \exp\left(\text{LeakyReLU}\left(a_{int}^{(l)}\left[\mathbf{W}_{int}^{(l)}h_i^{(l-1)} \parallel \mathbf{W}_{int}^{(l)}h_k^{(l-1)}\right] \bigodot E_{ik}\right)\right)} \tag{5.4}$$

$$\alpha_{ij,prot}^{(l)} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}_{prot}^{(l)}\left[\mathbf{W}_{prot}^{(l)}h_i^{(l-1)} \parallel \mathbf{W}_{prot}^{(l)}h_j^{(l-1)}\right] \bigodot E_{ij}\right)\right)}{\sum_{k\epsilon N_i^{(prot)}} \exp\left(\text{LeakyReLU}\left(\mathbf{a}_{prot}^{(l)}\left[\mathbf{W}_{prot}^{(l)}h_i^{(l-1)} \parallel \mathbf{W}_{prot}^{(l)}h_k^{(l-1)}\right] \bigodot E_{ik}\right)\right)} \tag{5.5}$$

where $\alpha_{ij,int}^{(l)}$ and $\alpha_{ij,prot}^{(l)}$ is the nomalized attention coefficient for the neighboring nodes in the interface graph and protein graph respectively, $\parallel$ is the concatenation operation, $a_{int}^{(l)}$ and $a_{prot}^{(l)}$ are the attention matrices for the interface nodes and the neighboring nodes respectively for layer l. Finally, each node vector is updated with a non-linear function applied over a linear combination of the node features using the attention coefficients:

$$\begin{aligned} h_i^{(l)} = \sigma\Bigg(&\mathbf{W}_c^{(l)}h_i^{(l-1)} + \frac{1}{|N_i^{(int)}|}\sum_{j\epsilon N_i^{(int)}} \alpha_{ij,int}^{(l)}\mathbf{W}_{int}^{(l)}h_j^{(l-1)} \\ &+ \frac{1}{|N_i^{(neigh)}|}\sum_{j\epsilon N_i^{(neigh)}} \alpha_{ij,neigh}^{(l)}\mathbf{W}_{neigh}^{(l)}h_j^{(l-1)} + b^{(l)}\Bigg) \end{aligned} \tag{5.6}$$

The self-attention in GAT can be stabilized by performing multi-head attention [99]. The intuition behind multi-head attention is to model information from different representation subspaces. Multi-head attention applied on GAT in equation (5.6) is shown below:

$$h_i^{(l)} = \|_{k=1}^{K} \sigma \left(\mathbf{W}_c^{(l,k)} h_i^{(l-1)} + \frac{1}{|N_i^{(int)}|} \sum_{j \epsilon N_i^{(int)}} \alpha_{ij,int}^{(l,k)} \mathbf{W}_{int}^{(l,k)} h_j^{(l-1)} + \frac{1}{|N_i^{(neigh)}|} \sum_{j \epsilon N_i^{(neigh)}} \alpha_{ij,neigh}^{(l,k)} \mathbf{W}_{neigh}^{(l,k)} h_j^{(l-1)} + b^{(l)} \right), \tag{5.7}$$

where K is the total number of attention heads, k is the attention head, $\mathbf{W}_c^{(l,k)}$ is the central node weight matrix, $\mathbf{W}_{int}^{(l,k)}$ is the weight matrix for neighboring nodes in interface graph, $\mathbf{W}_{neigh}^{(l,k)}$ is the weight matrix for neighboring nodes in protein graph, $\alpha_{ij,int}^{(l,k)}$ is the attention coefficient between node i and neighboring node j in the interface graph, and $\alpha_{ij,neigh}^{(l,k)}$ is the attention coefficient between node i and neighboring node j in the protein graph. The architecture defined by (5.7) will further be referred to as GAT.

5.4 Graph Pooling

Graph pooling helps represent the nodes of the graph at the graph level. Depending on the pooling strategy, Graph pooling can be divided into two types: Global pooling and Hierarchical pooling. Global pooling is the transformation of all the embeddings of the nodes of a graph into a single representation. Some of the notable graph pooling techniques are:

- **Global max pooling:** The max pool operation is performed by picking the maximum activation from each channel of the node embeddings in a graph as shown below:

$$\mathbf{r}_{max} = \max_{i=1}^{n_f^{(l)}} \mathbf{X}_i^{(l)}, \tag{5.8}$$

  where $\mathbf{X}^{(l)}$ is the node feature matrix and $n_f^{(l)}$ is the number of features in the node embedding matrix in layer l.

- **Global average pooling:** The average pooling is computed by taking a mean of the node embeddings across each channel in a graph as shown below:

$$\mathbf{r}_{avg} = \frac{1}{n_f^{(l)}} \sum_{i=1}^{n_f^{(l)}} \mathbf{X}_i^{(l)}. \tag{5.9}$$

- **Global sum unit vector pooling:** Here a global representation is computed by summing all the node embeddings and computing a unit vector in that direction as shown below:

$$\mathbf{r}_{norm} = \frac{\sum_{i=1}^{n_f^{(l)}} \mathbf{X}_i^{(l)}}{\sqrt{\sum_{i=1}^{n_f^{(l)}} (\mathbf{X}_i^{(l)})^2}}. \tag{5.10}$$

- **Self-Attention Graph Pooling (SAGPool)** [62]: The main idea of SAGPool is to apply self-attention over all the nodes in a graph to compute scores which represents the importance. Then top-k nodes are picked and a sub-graph involving these nodes are preserved.

Other global pooling algorithms include Set2Set [102] and SortPool [113]. The advantage with global pooling is that the pooling can be performed on graphs with variable number of nodes. However the disadvantage is that it doesn't account for the topology of the graph.

Hierarchical pooling is another form of graph pooling that considers the topology of the graph. Considering the topology of the graph helps in capturing the hierarchical representations that are very crucial in understanding the structure of the graph. Thus the main motive of hierarchical pooling is to involve topology while pooling the node features of the graph. Some of the more popular hierarchical include ASAP [85].

Chapter 6

Data and training

This chapter outlines the data and methods used for training. The data section talks about the data processing pipeline right from identification of protein complexes to the final train, validation, and test sets. The training section discusses the sampling technique, detailed model architecture, model selection, and model testing.

6.1 Data

This section discusses the data processing pipeline. The data processing pipeline comprises of three steps: Finding protein complexes, generating decoys using docking softwares, and separation of the complexes into train, validation, and test sets. Each of the step is discussed in more detail in the following subsections.

6.1.1 Protein-Protein interactions databases

A large pool of protein-protein complexes are collected from three databases: Docking Benchmark Dataset (DBD) version 5.0 (DBD5) [104], Dockground Unbound Set 4 [58], and ProPairs [57]. DBD5 consists of 230 high-quality 3D structures of protein-protein complexes and their component unbound structures. DBD5 is an update to DBD version 4.0 (DBD4) [46] where 55 new protein-protein complexes were added to the existing 175 protein-protein complexes in DBD4. These 55 new protein complexes added in DBD5 are used for testing. Dockground Unbound Set 4 consists of 396 protein-protein complexes. ProPairs consists of 2959 complexes. From this large collection of protein-protein complexes, only those are selected if they satisfy certain criteria. The first criterion are that the protein-protein complex should be a dimer, i.e., the protein-protein complex should strictly contain two chains. The second criteria are that for a protein-protein complex, both the bound structure and the unbound structure of its components should be available. The third criteria are that the protein-protein complex cannot be a homomer, i.e., each of the proteins

in the protein-protein complex needs to be distinct. Finally, in the fourth and final criterion, no protein-protein complex should be redundant among the collection of protein-protein complexes. A protein-protein complex is deemed redundant if the sequence identity between any two protein-protein complexes is greater than 90%. In that case, one of the protein-protein complexes will be eliminated randomly. All sequence identity computations are performed using the CD-hit software [44]. After filtering the protein complexes using the above criteria, there are a total of 425 complexes that are used for the next steps.

6.1.2 Decoy generation using docking softwares

For each of the protein-protein complex from the above step, protein-protein models are generated using one or more of the following docking softwares: ClusPro [55], FroDock [35], ZDock [84], and Haddock [26, 37]. Using the ClusPro docking server and FroDock, protein-protein decoys are generated using the unbound protein structures of the proteins in protein-protein complexes. Pre-computed decoys generated using the ZDock and Haddock docking algorithms on DBD5 complexes are borrowed. Further for each of the protein-protein decoy, graphs are constructed as described in Section 5.1 and the DOCKQ quality scores are calculated as detailed in Section 1.2.4. The decoys for which either the graph construction or calculation of DOCKQ scores failed are discarded from the decoy set. Also, the decoys of a docking software for a protein-protein complex are discarded if none of the decoys have a DOCKQ quality score greater than 0.45 (i.e. medium quality or above). The summary of the total number of protein-protein decoys for which both the graph and DOCKQ scores are available are shown below grouped by the doking software.

Table 6.1: Summary of number of decoys per docking software

Docking software	Number of decoys	Number of complexes
ClusPro	40861	420
FroDock	38783	392
ZDock	28090	28
Haddock	11599	29

6.1.3 Train, Validation, and Test sets

The protein-protein complexes in the dataset are separated into train, validation, and test set using the sequence identity thresholds between train-validation, train-test, and validation-test. The test set is designed to be challenging by including the new complexes added to DBD5, and similarly, in the validation set, possible DBD4 that could satisfy the thresholds are included. The train, validation, and test set is separated in such a way that the sequence identity between test and validation complexes is less than 25%, between test and train complexes is less than 25%, and between train and validation complexes is less than 40%. All SI computations are performed using the CD-hit tool [44]. Less sequence identity represents less similarity between the complexes, thus making the problem more challenging. Therefore, the test set should be most challenging in theory compared to the validation set when the machine learning model is trained on the train set. The final dataset consists of 388 protein-protein complexes in the train set, 20 protein-protein complexes in the validation set, and 17 protein-protein complexes in the test set. The number of complexes, number of decoys, number of decoys per DOCKQ category in each of the train, validation, and test sets are shown below:

Table 6.2: Distribution of the complexes and decoys in train, validation, and test sets

	Train	validation	Test
Number of complexes	388	20	17
Number of decoys	99182	5051	15525
Medium & high quality decoys	2081	132	470
Acceptable decoys	4306	211	843
Incorrect decoys	92795	4708	14212

6.2 Training

This section talks about the methods used to train the experiments. Each of the experiments fetches data to the neural network using a multi-level sampler. The data fetched from the train dataset is used for training the neural network. The trained model is inferenced on the validation set to select the appropriate parameters using the model selection strategy. Finally, the optimized model is evaluated and compared against other methods on the test set. The multi-level sampler, neural network architecture, model selection strategy, and the evaluation metric will be outlined in the following subsections.

6.2.1 Multi-level sampler

Binning is a sampling technique that samples equal proportion of examples per class from an unbalanced dataset. Neural networks do not distinguish between classes of examples that are shown while training, therefore in unbalanced datasets where there are minority classes having very few examples and majority classes having many examples, the neural networks are overwhelmed with learning representations from majority classes that they either learn little to nothing about the minority classes. This limitation is addressed using many approaches including binning. Binning samples equal number of examples per class in a batch thus helps neural network observe an equal distribution of classes.

As shown in the Table 6.2, the three quality categories: "Medium high quality decoys", "acceptable decoys", and "Incorrect decoys" have an unequal distribution of examples. The majority class are the Incorrect decoys whereas the minority classes are the acceptable, medium, and high quality decoys. If the neural network is trained on this unbalanced dataset, it wouldn't be able to differentiate between the quality categories. This problem is addressed using a multi-level sampler that samples equal number of decoys from each of the quality categories. The multi-level samples uses two steps to sample the decoys: 1) Pick a complex from the dataset by weighted sampling where there is a higher chance for complexes with more number of mid-high quality decoys to be picked more number of times. 2) For the complex picked from step 1, randomly pick a docking

software that's available and draw an equal number of decoys from each of the quality "Medium high quality decoys", "acceptable decoys", and "Incorrect decoys". If in case "Medium high quality decoys" are unavailable in the docking software for a sampled complex then the native complex structure is considered. Also, if decoys are unavailable for any of the quality category, then non-redundant docking decoys from other quality categories are sampled in the order of preference from high quality to low quality categories. The sampling process is summarized in the figure below.

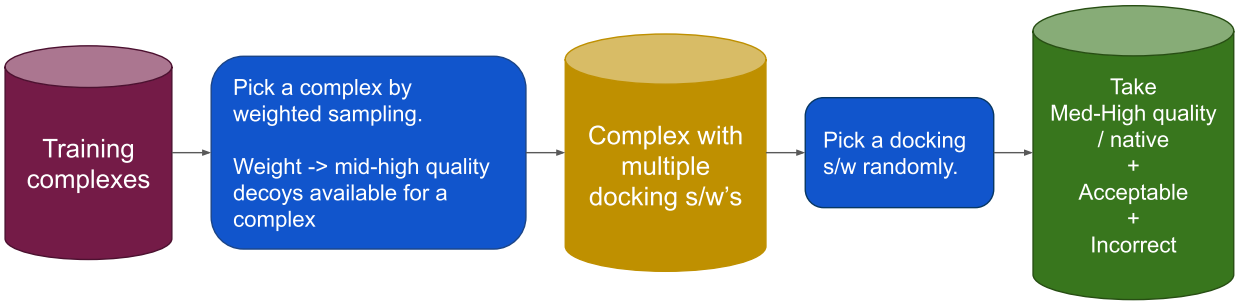


Figure 6.1: Multi-level sampling process.

6.2.2 Model architecture

The general scheme followed in designing the model architecture is to feed the input interface graph of a docking decoy to graph neural networks followed by a graph pooling layer, fully connected layers, and a single output unit that predicts the quality score. The motivation behind the architectural design is to learn useful representations from the input graph using graph neural networks and transform the graph into euclidean space where traditional neural networks can be applied to make predictions. Three variety of graph neural networks are used bGCN, GCN, and GAT. Two kinds of pooling layer- Normalize pooling, and SAGPool are used. The diagram of the model architecture from input to output is shown in Figure 6.2.

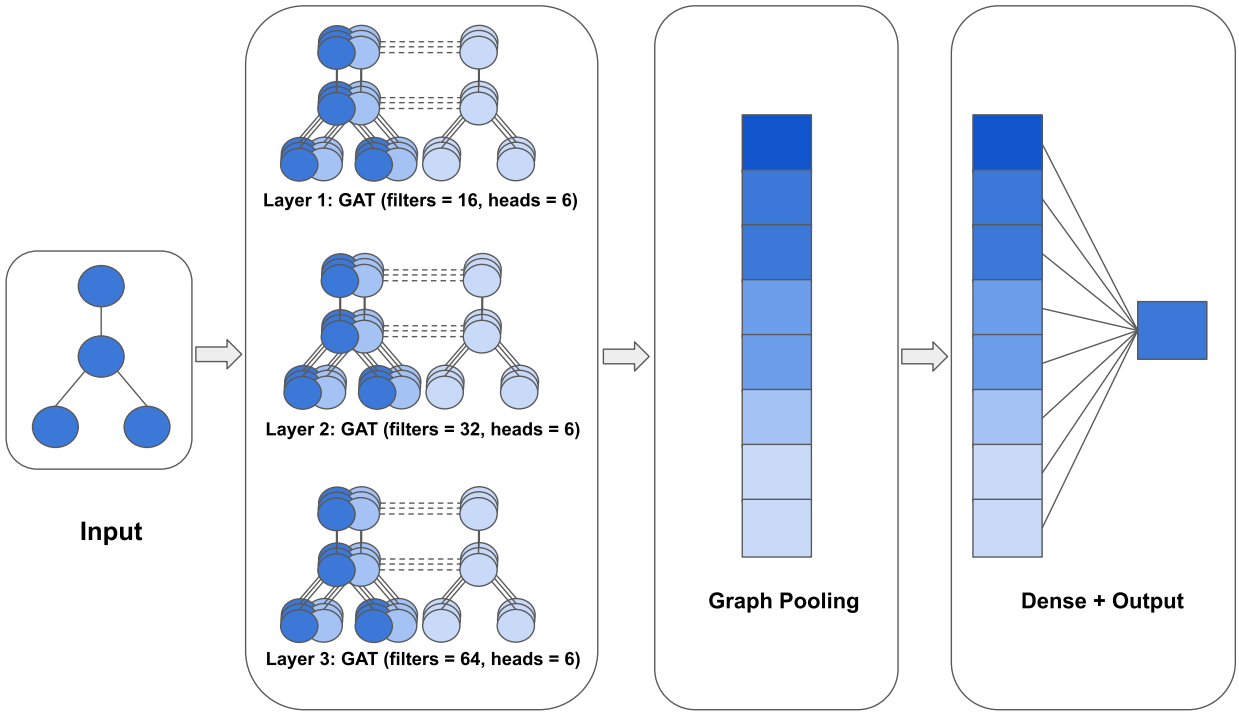


Figure 6.2: Model architecture showing the propogation of inputs through multiple graph convolutional layers, a graph pooling layer, a dense layer and finally a single neuron that produces the predicted score.

6.2.3 Model training

The neural network is trained in mini-batches, where a mini-batch consisting of 15 docking decoys is fed to the neural network using the multi-level sampler that samples five protein complexes and one docking decoy from each of the three quality categories. The output of the neural network is the predicted quality score for each of the input in the mini-batch. A loss function that measures the error between the predicted quality score and the label DockQ quality score is calculated using regression loss function or ranking loss function for the mini-batch. Using regression and ranking loss functions helps identify the near-native docking decoys as well as order them w.r.t the quality. We use two regression loss functions L1-loss (L_1) and L2-loss(L_2):

$$L_1 = \frac{1}{n}\left(\sum_{i=1}^{n}(y_i - \hat{y}_i)\right), \tag{6.1}$$

$$L_2 = \frac{1}{n}\left(\sum_{i=1}^{n}(y_i - \hat{y}_i)^2\right), \tag{6.2}$$

where n is the number of decoys in the mini-batch, y_i is the label DOCKQ value and $\hat{y}_i$ is the predicted score for the decoy i. Another loss function called the ranking loss function ($\mathrm{L_{rank}}$) or hinge loss as shown below is used inspired by Derevyanko et al. [24]

$$\mathrm{L_{rank}} = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} \max\left[0, 1 - b_{ij} \cdot (\hat{y_i} - \hat{y_j})\right], \tag{6.3}$$

where a_{ij} is one if $|y_i - y_j| > \epsilon$ else 0, where ϵ is 0.1 and the value of b_{ij} is +1 if $y_i \leq y_j$ or -1 otherwise. The loss function is optimized using the optimizers Adam [53] or AMSGrad variant of Adam [86]. The model is trained for 200 epochs where an epoch consists of 500 mini-batches. To mitigate the problem of overfitting while training the neural networks regularization techniques such as Dropout [94], L1 and L2 regularization [77] and early stopping were used. All the neural network models are implemented using PyTorch [81] and trained on either NVIDIA TITAN V or TITAN X GPU cards. The time taken to train the model varies from 25 minutes to 6 hours depending the type of graph convolution, the number of parameters, network size, and GPU card. Typically, both baseline-GCN and GCN trains under 30 minutes while GAT depending on the number of attention heads takes 6 hours.

6.2.4 Metrics

Among the algorithms for scoring protein-protein docking models, the evaluation metric is not standardized. The two most common metrics among these scoring algorithms [17, 109, 10] are Enrichment factor and DockQ quality measure, and ranking ability using top-n ranks. Enrichment factor is a comparative evaluation metric that tells how better the scoring algorithm ranks when compared to a random ranking order. Although both enrichment factor and correlation coefficient give an idea about the robustness of the scoring algorithm, these metrics do not directly tell how effective the scoring algorithm is in identifying the near-native decoys. One of the simple ways to capture this information is to find the number of complexes having near-natives above a given quality in top-n ranks when the decoys are ranked in the order of the score predicted by scoring algorithms. In this thesis, we use this metric and show the results of the number of complexes

for which the decoys above a certain quality threshold are in top-1, 5, 10, and 50 as shown in Figure 7.1. The three quality thresholds defined are: DOCKQ > 0.23 a.k.a acceptable and above quality, DOCKQ > 0.49 a.k.a medium and high quality, and DOCKQ > 0.65 a.k.a high and above quality. This metric will further be referred to as "top-n near-native" to make it simpler. In addition to evaluating the results on near-native, the performance of the model in ranking the native structure is considered. This is done by predicting the score of the native structure and ranking it among the pool of docking decoys for a complex to find if its within top-1, 5, 10, and 50 ranks. This metric using natives will be referred to as "top-n natives". All the results in thesis are evaluated using the top-n near-native and native metric.

6.2.5 Model Selection

Model selection is the process of finding the model with a set of hyper-parameters that gives the best results. Two types of searches are performed in model selection: architectural search and hyper-parameters search. Architectural search was performed manually using intuition as the basis by experimenting different proposed GNNs and varying the number of GNN and fully connected layers. The observations from the architectural search experiments were that GATs perfomed the best compared to baseline-GCN and GCN. Also, the performance of the model did not increase with more than 3 layers of GNNs (bGCN, GCN) and 1 fully connected layer. Using more than 3 layers of GAT couldn't be experimented because of lack of accessibility to machines having sufficient GPU memory. The GNNs consisting of baseline-GCNs, GCNs, and GATs with 16, 32, and 64 filters in each of the layer gave the best results. In experiments with GATs, using 6 multi-head attention gave the best results and more than 6 attention heads couldn't be experimented because of memory constraints.

We perform hyper-parameter search on learning rate of the optimizer, weight initialization, activation function, and regularization parameters such as dropout, L2 penalty, and optimizer. The search space we defined is shown in Table 6.3. From the search space ten sets of hyper-parameters are randomly sampled to find the best combination. The best combination is identified by picking

the experiment that has the highest near-native complexes in top-20 ranks and high quality threshold. Although more number of trials need to be done, due to memory constraints the number of random samples were limited to ten. Among the search space for each of the hyper-parameters the best value is indicated in the table below.

Table 6.3: Hyper-parameters and their search space used in model selection. The best combination is listed in the last column.

Hyper-parameter	Search space	Best value
Learning rate	[0.00001, 0.00005, 0.0001, 0.0003, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1]	0.0001
Weight initialization	[Kaiming, xavier]	kaiming
Activation function	[ReLU, leaky ReLU]	leaky ReLU
Dropout	[0.1, 0.2, 0.3, 0.4, 0.5, 0.6]	0.3
L2 penalty	[0.00001, 0.00005, 0.0001, 0.0003, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1]	0.0001
Optimizer	[Adam, AMSGrad variant of Adam [86]]	Adam

Chapter 7

Results

This chapter first lists a set of experiments to explain the performance of the proposed graph neural network techniques on validation and test set. Finally, the best performing graph neural network is compared against other scoring functions such as ZDock, Haddock, FroDock, and DOVE on the test set.

In the results that follow, the performance of each method is measured using either one or both of the following metrics based on the ranking of docking decoys: 1) top-near-native complexes: number of complexes having high-quality and above docking decoys in top-1, 5, 10, 20, and 50 ranks and 2) top-n native complexes: number of complexes having natives in the top-1, 5, 10, 20, and 50 ranks. The performance of a scoring algorithm should be better on the top-n native complexes than the top-n near-native complexes because the quality separation between near-native and native is large. Therefore, identifying natives should be an easier task than identifying near-natives. When computing the results on the entire validation or test datasets, for each of the complex the docking decoys are chosen from the docking software that has the best quality decoy, i.e., the docking software having a docking decoy with the highest DockQ value. Otherwise, the results are evaluated on an explicitly mentioned docking software.

7.1 Comparision of proposed Graph Neural Networks

From many experiments conducted in model selection, we choose eight GNNs as shown in the Table 7.1. In our preliminary experiments we identified A as the best experiment on validation set and conducted an ablation study by varying the type of graph convolution, graph pooling, loss function, and the label. The experiments are compared by looking at the near-native complexes and native complexes.

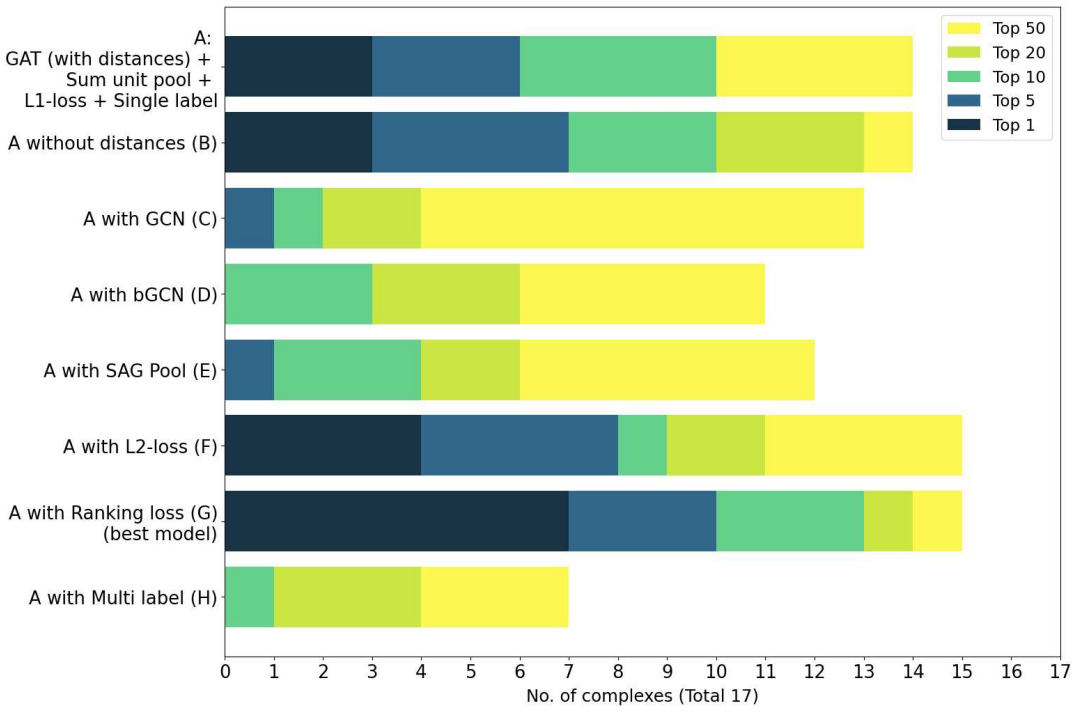
As discussed in the previous chapter, the architecture of the proposed GNN consists of an input layer, three baseline-GCN, GCN or GAT layers, graph pooling layer, a fully connected layer, and

Table 7.1: List of experiments

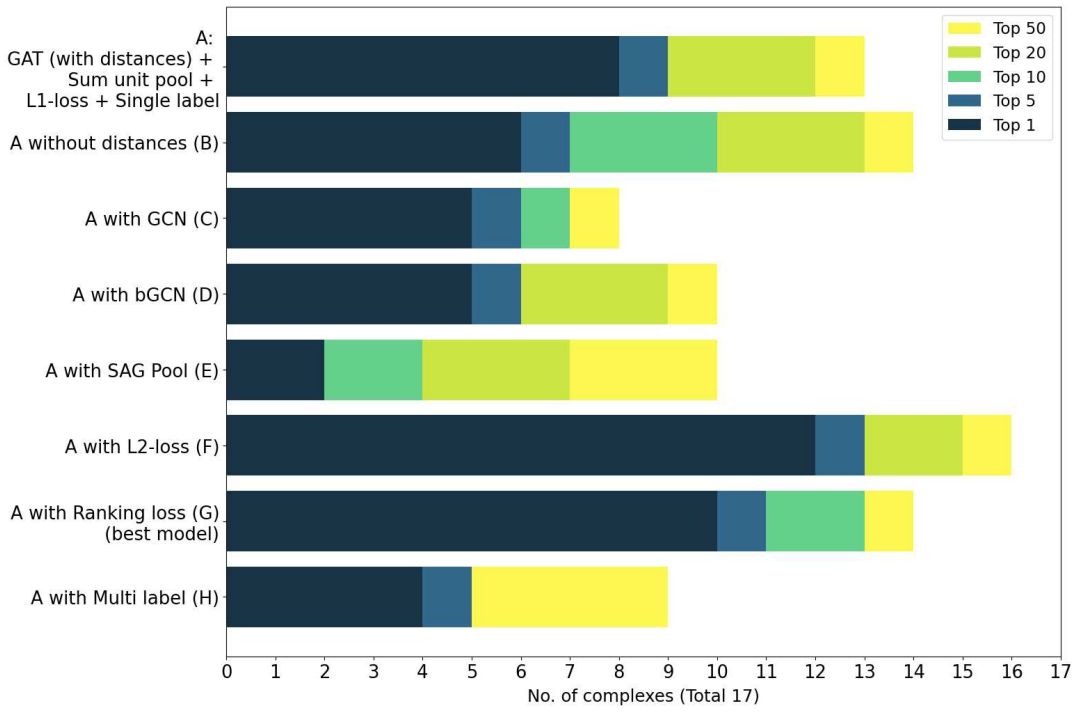
Name	Experiment
A	GAT (with distances) + Sum unit pool + L1-loss + Single label
B	GAT (without distances) + Sum unit pool + L1-loss + Single label
C	GCN (with distances) + Sum unit pool + L1-loss + Single label
D	baseline-GCN (with distances) + Sum unit pool + L1-loss + Single label
E	GAT (with distances) + SAG pool + L1-loss + Single label
F	GAT (with distances) + Sum unit pool + L2-loss + Single label
G	GAT (with distances) + Sum unit pool + Ranking loss + Single label
H	GAT (with distances) + Sum unit pool + L1-loss + Multi label

an output layer that predicts one or four different scores. Mentioning baseline-GCN, GCN, or GAT in an experiment indicates three such layers with 16, 32, and 64 filters and 6 attention heads if GAT is used. In order to understand the importance of distances in baseline-GCN, GCN, or GAT, two options are used "with distances" or "without distances". The option "with distances" indicates the use of distances to scale the hidden representation between two nodes as defined in bGCN, GCN, or GAT. Whereas, "without distances" indicates that the hidden representation between two nodes is not scaled by the distance. In this case, E_{ij} is equal to one in bGCN, GCN, or GAT. Single label in the experiment indicates use of DockQ as the label, whereas multi-label indicates the use of DockQ and three components of DockQ: Fnat, lRMSD, and iRMSD.

Figure 7.1 shows the results of the experiments mentioned in Table 7.1 on the test set. Among the experiments, experiment G performed the best in identifying the most number of near-native complexes in top-1, 5, 10, 20, and 50 ranks. Whereas, experiment F was able to identify more number of native complexes in top-1, 5, 10, 20, and 50 ranks. Comparing different building blocks:



(a)



(b)

Figure 7.1: Comparing top-n ranking performance of the experiments listed in Table 7.1 on the test set. Reported are the number of complexes having (a) high quality and above decoys and (b) natives, in top-p ranks.

bGCN, GCN, and GAT using experiments A, C, and D respectively, the results show the order of performance is GAT, GCN, and bGCN. This justifies the arguments presented for the design of GCN and GAT. The results on experiments A and B show that infact not using distances in GAT can improve the performance contrary to the intuition expressed in designing bGCN. The experiments A and E compare the graph pooling layer- Sum unit pooling and SAG Pool, and the results show Sum unit pooling performs better than SAG Pool. Among the loss functions- L1, L2, and ranking loss experimented in A, F, and G respectively, the order of performance is Ranking loss, L2 loss, and L1 loss. Finally, experiment G and H shows that single label performs better than multi-label.

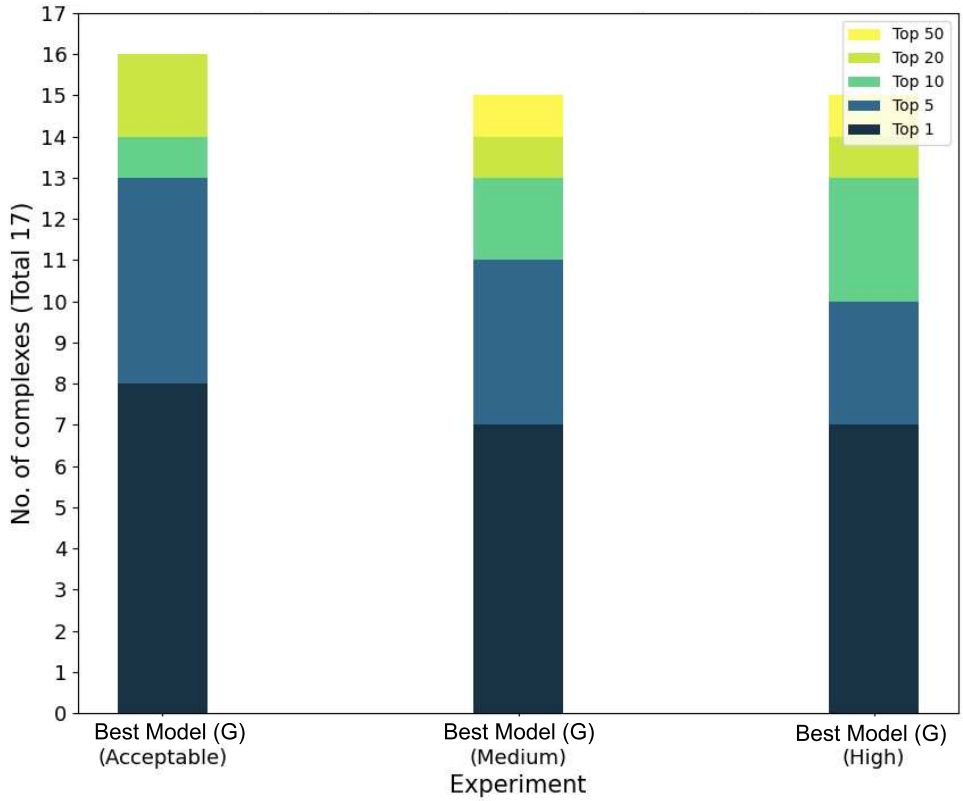


Figure 7.2: Comparing top-n ranking performance of the best model at different quality thresholds on the test set. Reported are the number of complexes having docking decoys above the quality threshold in top-p ranks.

Next, the performance of the best model - experiment G is compared at three quality thresholds: acceptable quality and above, medium quality and above, and high quality and above. The results from this analysis on the test set are shown in Figure 7.2. As the DockQ threshold is relaxed the problem is easier to solve. Therefore the near-native complexes identified in top-1, 5, 10, 20, and 50 are the same or more as the quality threshold moves from high to medium and medium to acceptable.

7.2 Comparison against other methods

In this section the best experiment from the proposed GNNs is compared against other protein-protein docking scoring techniques. The other scoring techniques used in comparison are: ZDock [73], FroDock [35], Haddock [26], and DOVE [109]. The comparison is drawn by computing the number of complexes having docking decoys above acceptable, medium, and high quality in top-1, 5, 10, 20, and 50 ranks, when ordered according to the scoring algorithms.

Figure 7.3 (a), Figure 7.3 (b), and Figure 7.3 (c) compares the best model (experiment G) against ZDock [73] on 10 complexes, FroDock [35] on 16 complexes, and Haddock [26] on 6 complexes respectively. In the first comparison against ZDock, our model is able to identify more near-native complexes in all three quality thresholds compared to ZDock. From the second comparison against FroDock, the results show that our best model is able to identify more number of near-native complexes in top-1, 5, and 10 ranks. Both perform the same on top-20 ranks, and in top-50 ranks FroDock performs slightly better. The third comparison against Haddock shows that our model performs better in top-5, 10, 20, and 50 ranks. In top-1 ranks both methods are unable to identify at least one near-native complex. From the above comparisons we can conclude that our best model performs better than the three docking + scoring methods.

Next, we make a comparison against Dove [109] that uses 3D CNNs to score docking decoys as explained in Chapter 4. The results for Dove [109] on the test set for near-native complexes in top-1, 5, 10, 20, and 50 ranks at acceptable and high thresholds are obtained from Bontha [14]. Among the complexes in the test set, results were computed for 12 complexes. The results in

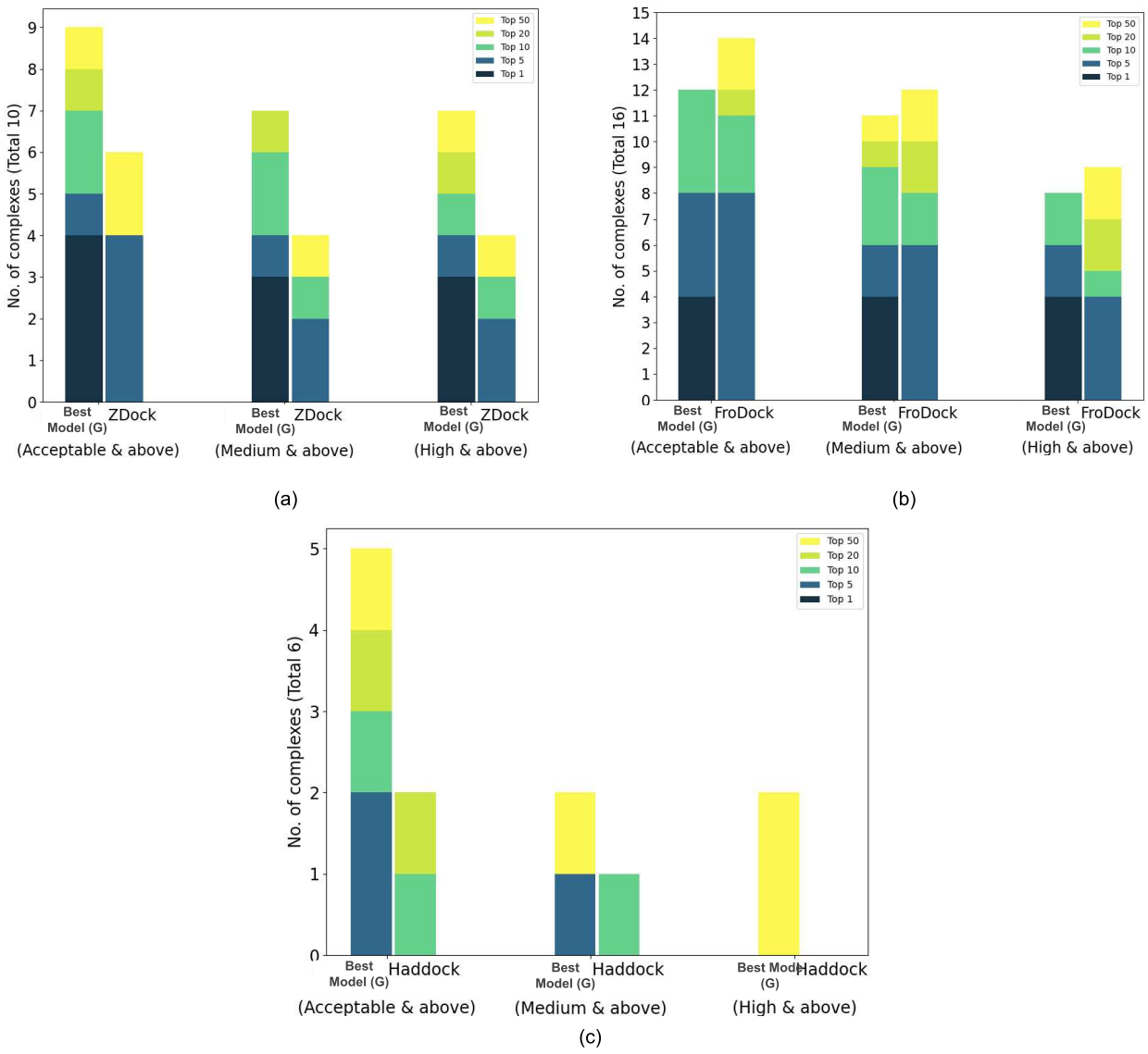


Figure 7.3: Comparing top-n ranking performance of the best model against (a) ZDock, (b) FroDock, and (c) Haddock docking + scoring methods at different quality thresholds on the test set. Reported are the number of complexes having docking decoys above the quality threshold in top-1, 5, 10, 20, and 50 ranks.

Figure 7.4 show that our best model performs better than Dove in all top-1, 5, 10, 20, and 50 ranks across acceptable and high quality thresholds.

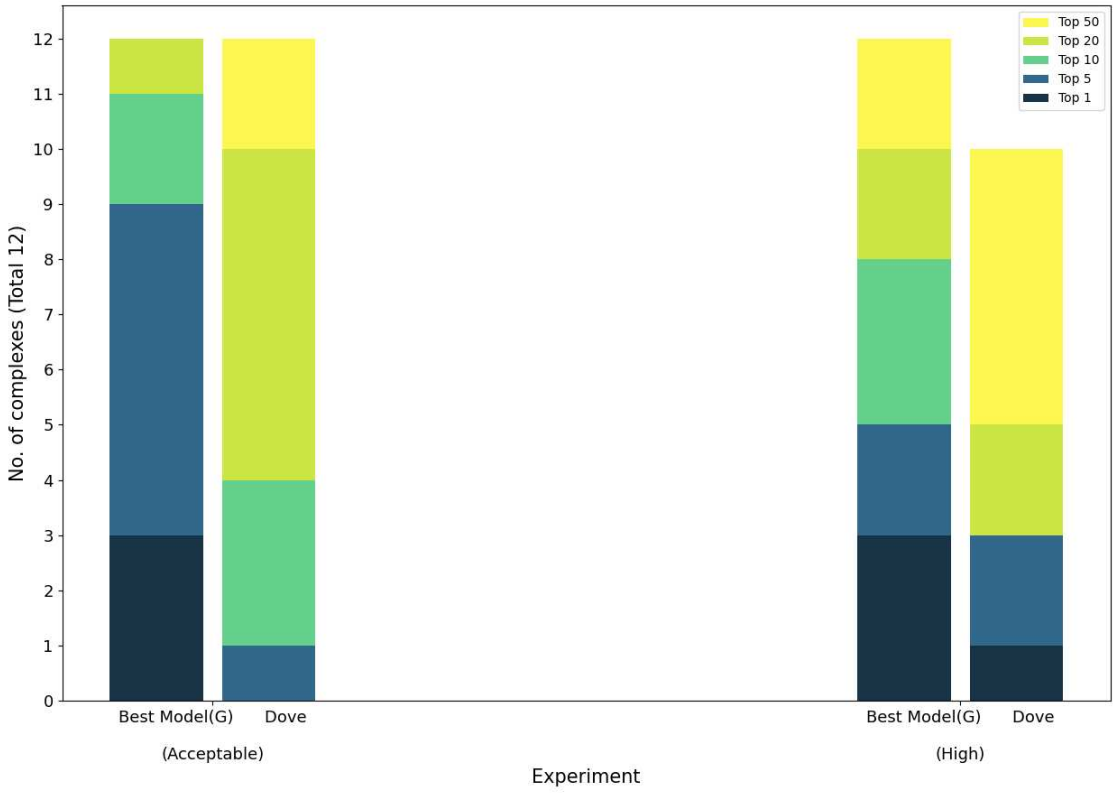


Figure 7.4: Comparing top-n ranking performance of the best model vs Dove[109] at different quality thresholds on the test set. Reported are the number of complexes having docking decoys above acceptable and high quality threshold in top-p ranks.

Our final comparison is against the 3D CNN based scoring method proposed by Bontha [14]. This method uses the dataset and sampling techniques proposed in this thesis and only differs by the neural network technique and architecture. Therefore, comparing both the methods demonstrates how good 3D CNNs and GNNs are at scoring docking decoys. The results in the Figure 7.5 show that our best model performs better than 3D CNNs in all top-1, 5, 10, 20, and 50 ranks across acceptable, medium, high quality thresholds and natives. From this analysis, we can conclude that between the best models trained using GNNs and 3D CNNs to score docked protein-protein models, GNNs perform better. Note that we did not compare our model against EGCN since they did not publish their code or detailed enough results.

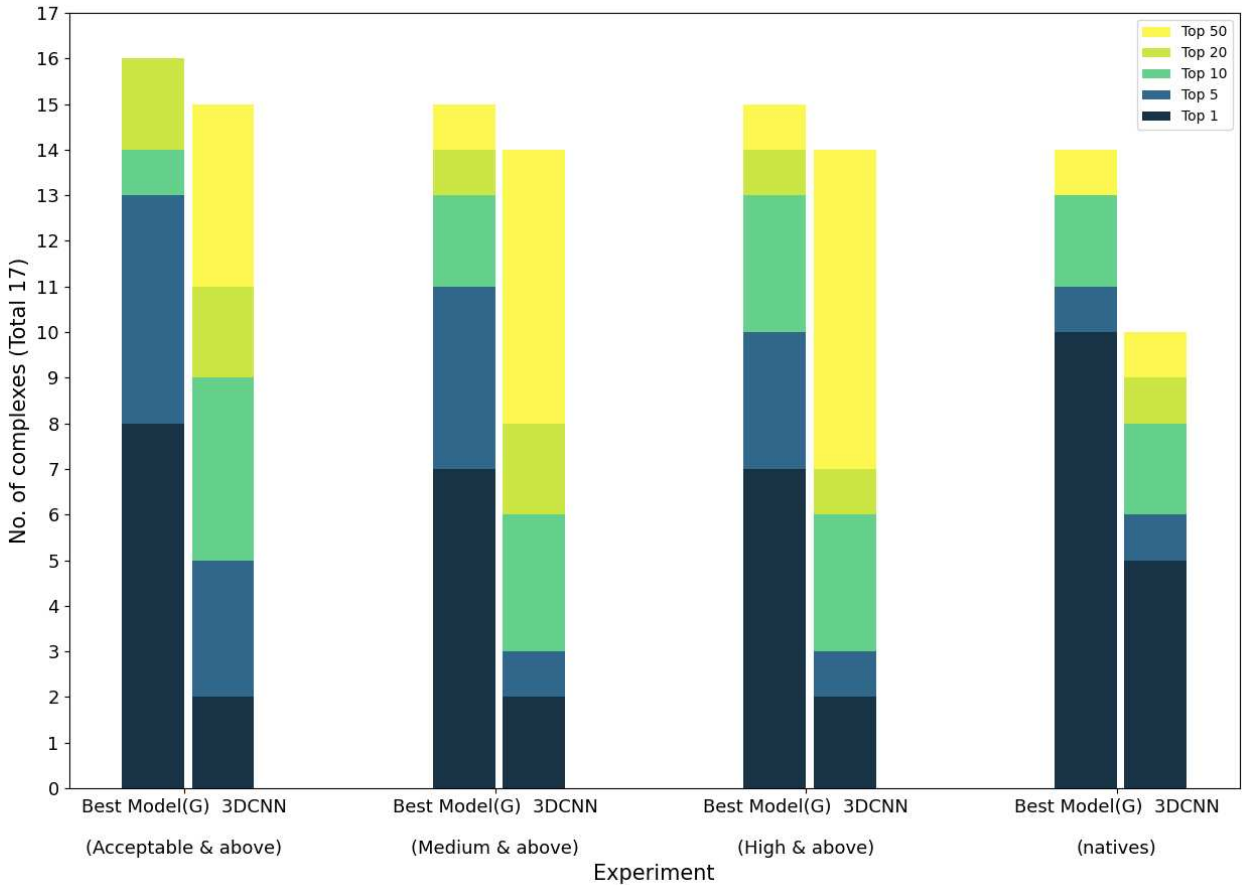
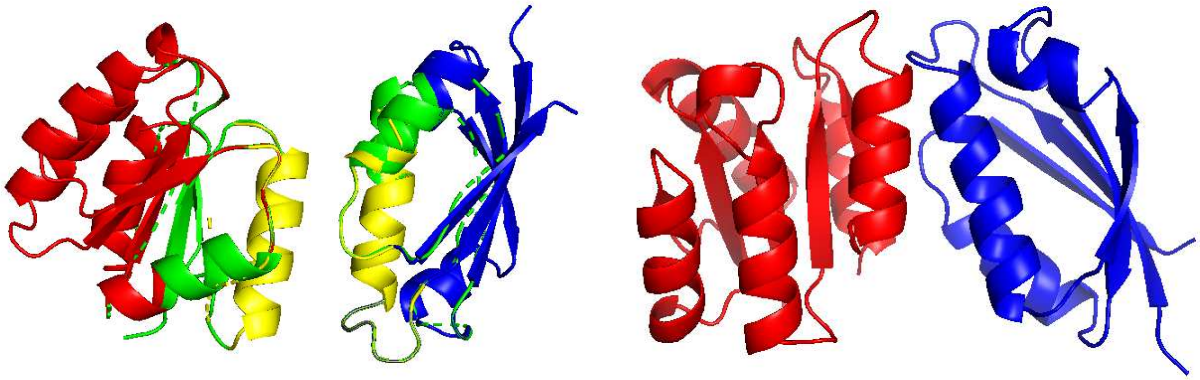


Figure 7.5: Comparing top-n ranking performance of the best model vs 3DCNN at different quality thresholds on the test set. Reported are the number of complexes having docking decoys above acceptable and high quality threshold in top-p ranks.

7.3 Misclassifications

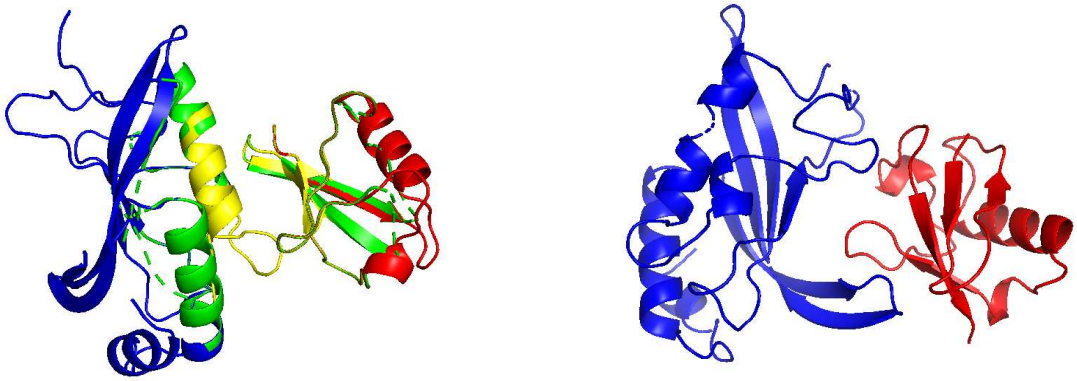
In this section, we will be investigating the false positives, i.e., the docking solutions for which the scoring technique ranked them in top ranks, but in reality, they are of low quality. Investigating these misclassifications can help us understand why our scoring technique performed poorly and could provide ideas for possible new features.



Complex: 1OUS
Docking s/w: FroDock

Complex: 1OUS
Native

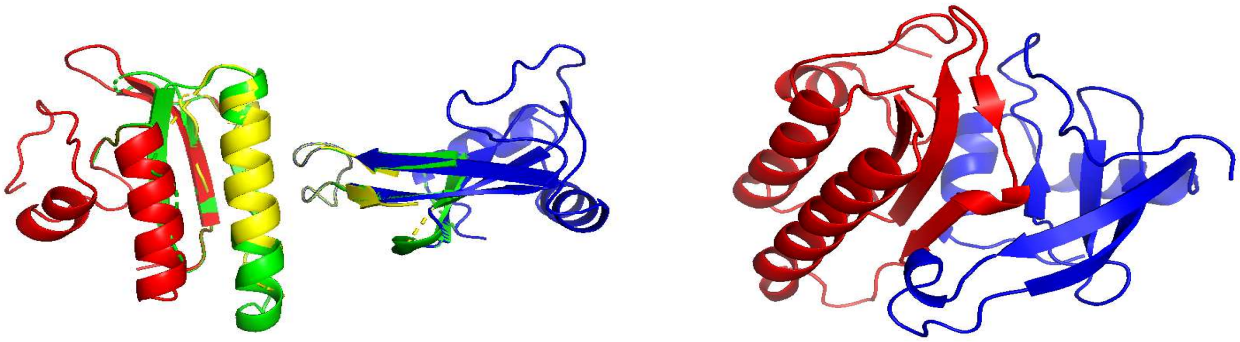
(a)



Complex: 1S1Q
Docking s/w: Haddock

Complex: 1S1Q
Native

(b)



(c)

Figure 7.6: Misclassified docking solution for the protein complex (a) 1OUS generated using FroDock docking software (b) 1S1Q generated using Haddock docking software (c) 2FHZ generated using ClusPro docking software, and their corresponding native complex

From Figure 7.6 we make two observations. First, the size of the interface of the docking solution is very small, and second, the interface of the docking solution is not compact compared to the interface of its native structure. Both the problems can be solved by either feature engineering or modifying the pooling layer to capture the size of the interface. The feature engineering approach is to calculate the interface area or in graph pooling layer we can aggregate the hidden representations of the nodes in the interface graph.

7.4 Conclusion

In this thesis we presented a graph neural network based scoring technique to rank protein-protein docking models. We present graph convolution techniques that use standard GCNs and compare them to graph attention networks. Our experiments demonstrated the benefit of using graph attention. In addition, a detailed ablation study showed the effectiveness of the chosen architecture. We also compare our best model against other scoring methods and demonstrate that our model performs better than other traditional and machine learning based scoring techniques such as ZDock [73], FroDock [35], Haddock [26], and two 3D CNN approaches [109, 14]. Two important takeaways from our comparison are as follows: 1) It is possible to obtain good results

without the need for extensive feature engineering; 2) Among deep learning approaches, GNNs out-performed 3D CNNs.

Chapter 8

Future work

In this section we discuss ideas to further improve the performance of our model. The two most important aspects of designing accurate machine learning models are the data and machine learning technique. We mention sources to increase the size of our dataset and potential new features that can be incorporated. We also briefly describe possible ways to improve the models themselves. We think that additions to both data and graph neural network technique along with robust model selection will help improve performance. Finally, discuss further comparison with the state of the art directions for model interpretation.

In this thesis we used 11 atom types (see Table 5.1) to represent a node in our graph, but results by a member of the lab [14] show that using only four atom types (C, N, O, and other) improves the performance of the model. We would like to see if using four atom types would improve our model's performance as well. Next, our current model uses atom-level features and does not capture residue-level information. We plan to capture residue-level information by creating a graph from the protein-protein interface, where the nodes in the graph represent the residues and their features and the edges represent the distance between two residues. We plan to experiment with three different types of residue features. First, a residue in the graph can be represented using one-hot encoding of the 21 different amino acid residues. Second, we will use the ProtTrans [29] embeddings, which are generated in a similar way to the commonly used BERT embeddings [25]. These embeddings have been shown to capture the function of amino acids, and the shape of proteins [29]. Finally, we plan to integrate the work from our research group by Soumyadip on ranking protein models by using the embedddings generated from his ranking technique. Using the embeddings from his research is beneficial because the forces that guide protein folding also guide protein-protein interactions. Thus identifying a good interface is analogous to identifying a good protein. Both the ProtTrans and protein scoring embeddings will allow for transfer learning which compensates for the lack of data.

The current dataset can be increased in size by adding additional complexes and decoys from different docking software. Our current test consists of only 17 complexes and can be further increased by adding complexes from the CAPRI score-set [65] and MOAL dataset [75]. More decoys can be added to the train and test set from SwarmDock [98] decoys on the Docking Benchmark 4 and 5 datasets.

The analysis we plan to conduct includes examining the misclassifications of docking decoys by our best model by identifying the characteristics of false positives. Next, using the probabilities from our graph attention layer, we would like to recognize the atom-atom interactions that help determine the best decoy and visualize them on the protein-protein interface. Further, we would like to make comparisons against the state-of-the-art scoring techniques such as ProQDock [10], and SwarmDock [98].

Bibliography

[1] Fayyaz ul Amir Afsar Minhas, Brian J Geiss, and Asa Ben-Hur. "PAIRpred: Partner-specific prediction of interacting residues from sequence and structure: Interface Prediction Using PAIRpred". eng. In: *Proteins, structure, function, and bioinformatics* 82.7 (2014), pp. 1142–1155.

[2] Grigoris Amoutzias and Yves Van de Peer. "Single-Gene and Whole-Genome Duplications and the Evolution of Protein–Protein Interaction Networks". In: *Evolutionary Genomics and Systems Biology*. John Wiley Sons, Ltd, 2010. Chap. 19, pp. 413–429. ISBN: 9780470570418. DOI: https://doi.org/10.1002/9780470570418.ch19.

[3] Grigoris Amoutzias and Yves Van de Peer. "Single-Gene and Whole-Genome Duplications and the Evolution of Protein–Protein Interaction Networks". eng. In: *Evolutionary Genomics and Systems Biology*. Hoboken, NJ, USA: John Wiley Sons, Inc, 2010, pp. 413–429. ISBN: 9780470195147.

[4] Jessica Andreani, Guilhem Faure, and Raphaël Guerois. "Versatility and invariance in the evolution of homologous heteromeric interfaces". In: *PLoS Comput. Biol., 2012, vol. 8* (2012), e1002677.

[5] Jessica Andreani, Guilhem Faure, and Raphael Guerois. "InterEvScore: a novel coarse-grained interface scoring function using a multi-body statistical potential coupled to evolution". In: *Bioinformatics* 29.14 (May 2013), pp. 1742–1749. DOI: 10.1093/bioinformatics/btt260.

[6] Pierce B and Weng Z. "ZRANK: reranking protein docking predictions with an optimized energy function". In: *Proteins, 2007, vol. 67* (2007), pp. 1078–1086.

[7] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473* (2014).

[8] Caterina Barillari et al. “Classification of water molecules in protein binding sites”. In: *Journal of the American Chemical Society* 129.9 (2007), pp. 2577–2587.

[9] Sankar Basu and Björn Wallner. “DockQ: A Quality Measure for Protein-Protein Docking Models”. eng. In: *PloS one* 11.8 (2016), e0161879–e0161879.

[10] Sankar Basu and Björn Wallner. “Finding correct protein-protein docking models using ProQDock”. eng. In: *Bioinformatics (Oxford, England)* 32.12 (2016), pp. i262–i270.

[11] Asa Ben-Hur and Jason Weston. “A User’s Guide to Support Vector Machines”. eng. In: *Data Mining Techniques for the Life Sciences*. Methods in Molecular Biology 609 (2009), pp. 223–239.

[12] Helen M Berman et al. “The protein data bank”. In: *Nucleic acids research* 28.1 (2000), pp. 235–242.

[13] Aleksandar Bijelic and Annette Rompel. “Ten Good Reasons for the Use of the Tellurium-Centered Anderson-Evans Polyoxotungstate in Protein Crystallography.” In: *Accounts of chemical research vol. 50,6* (2017), pp. 1441–1448.

[14] Mridula Bontha. “Quality Assessment of Docked Protein Interfaces using 3D Convolution”. MA thesis. Colorado State University, 2021.

[15] “Fundamentals of protein structure”. it. In: *Structural Bioinformatics, chapter 2*. Ed. by Philip E. Bourne and Helge Weissig, pp. 15–39.

[16] Yue Cao and Yang Shen. “Bayesian Active Learning for Optimization and Uncertainty Quantification in Protein Docking”. eng. In: *Journal of chemical theory and computation* 16.8 (2020), pp. 5334–5347.

[17] Yue Cao and Yang Shen. “Energy-based graph convolutional networks for scoring protein docking models”. In: *Proteins: Structure, Function, and Bioinformatics* 88.8 (2020), pp. 1091–1099. DOI: https://doi.org/10.1002/prot.25888.

[18] Rong Chen, Li Li, and Zhiping Weng. “ZDOCK: An initial-stage protein-docking algorithm”. eng. In: *Proteins, structure, function, and bioinformatics* 52.1 (2003), pp. 80–87.

[19] Rong Chen and Zhiping Weng. “A novel shape complementarity scoring function for protein-protein docking”. eng. In: *Proteins, structure, function, and bioinformatics* 51.3 (2003), pp. 397–408.

[20] Rong Chen and Zhiping Weng. “Docking unbound proteins using shape complementarity, desolvation, and electrostatics”. eng. In: *Proteins, structure, function, and bioinformatics* 47.3 (2002), pp. 281–294.

[21] Rong Chen et al. “A protein–protein docking benchmark”. eng. In: *Proteins, structure, function, and bioinformatics* 52.1 (2003), pp. 88–91.

[22] Michael E Cusick et al. “Interactome: gateway into systems biology”. eng. In: *Human molecular genetics* 14.suppl-2 (2005), R171–R181.

[23] P. J. Dalianis, S. G. Tzafestas, and G. Anthopoulos. “A study of the generalization capability versus training in backpropagation neural networks”. In: *Proceedings of IEEE Systems Man and Cybernetics Conference - SMC*. Vol. 4. 1993, 485–490 vol.4. DOI: 10.1109/ICSMC.1993.390760.

[24] Georgy Derevyanko et al. “Deep convolutional networks for quality assessment of protein folds”. eng. In: *Bioinformatics* 34.23 (2018), pp. 4046–4053.

[25] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423.

[26] Cyril Dominguez, Rolf Boelens, and Alexandre M. J. J Bonvin. “HADDOCK: A Protein-Protein Docking Approach Based on Biochemical or Biophysical Information”. eng. In: *Journal of the American Chemical Society* 125.7 (2003), pp. 1731–1737.

[27] Timothy Dozat. “Incorporating Nesterov Momentum into Adam”. In: 2016.

[28] David K Duvenaud et al. “Convolutional Networks on Graphs for Learning Molecular Fingerprints”. In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes et al. Vol. 28. Curran Associates, Inc., 2015.

[29] Ahmed Elnaggar et al. “ProtTrans: Towards Cracking the Language of Life’s Code Through Self-Supervised Deep Learning and High Performance Computing”. In: *arXiv e-prints*, arXiv:2007.06225 (July 2020), arXiv:2007.06225.

[30] Juan Esquivel-Rodríguez, Yifeng David Yang, and Daisuke Kihara. “Multi-LZerD: Multiple protein docking for asymmetric complexes”. eng. In: *Proteins, structure, function, and bioinformatics* 80.7 (2012), pp. 1818–1833.

[31] Guilhem Faure, Jessica Andreani, and Raphaë Guerois. “InterEvol database: exploring the structure and evolution of protein complex interfaces”. In: *Nucleic Acids Res., vol. 40* (2012), pp. D847–D856.

[32] Alex Fout et al. “Protein Interface Prediction using Graph Convolutional Networks”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017.

[33] Francesco Frigerio et al. “Crystal and molecular structure of the bovine -chymotrypsin-eglin c complex at 2.0 Å resolution”. eng. In: *Journal of molecular biology* 225.1 (1992), pp. 107–123.

[34] K. Fukushima. “Neocognitron”. In: *Scholarpedia* 2.1 (2007). revision #91558, p. 1717. DOI: 10.4249/scholarpedia.1717.

[35] José Ignacio Garzon et al. “FRODOCK: a new approach for fast rotational protein–protein docking”. eng. In: *Bioinformatics* 25.19 (2009), pp. 2544–2551.

[36] Jonas Gehring et al. *A Convolutional Encoder Model for Neural Machine Translation*. cite arxiv:1611.02344Comment: 13 pages. 2016.

[37] C Geng, LC. Xue, and A Bonvin. *Docking models for Docking Benchmark 4, 5 and CAPRI score_set.* 2019. DOI: 10.15785/SBGRID/684.

[38] Xavier Glorot and Yoshua Bengio. “Understanding the difficulty of training deep feedforward neural networks”. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Yee Whye Teh and Mike Titterington. Vol. 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: JMLR Workshop and Conference Proceedings, 13–15 May 2010, pp. 249–256.

[39] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[40] J Goodman. “Classes for fast maximum entropy training”. eng. In: *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*. Vol. 1. IEEE, 2001, 561–564 vol.1. ISBN: 0780370414.

[41] Jeffrey J Gray et al. “Protein–Protein Docking with Simultaneous Optimization of Rigid-body Displacement and Side-chain Conformations”. eng. In: *Journal of molecular biology* 331.1 (2003), pp. 281–299.

[42] Takuo Hamaguchi et al. “Knowledge Base Completion with Out-of-Knowledge-Base Entities: A Graph Neural Network Approach”. jpn. In: *Transactions of the Japanese Society for Artificial Intelligence* 33.2 (2018), F-H72_1–10.

[43] Sheng-You Huang and Xiaoqin Zou. “An iterative knowledge-based scoring function for protein–protein recognition”. eng. In: *Proteins, structure, function, and bioinformatics* 72.2 (2008), pp. 557–579.

[44] Ying Huang et al. “CD-HIT Suite: a web server for clustering and comparing biological sequences”. eng. In: *Bioinformatics* 26.5 (2010), pp. 680–682.

[45] D. H. Hubel and T. N. Wiesel. “Receptive fields and functional architecture of monkey striate cortex”. In: *The Journal of Physiology* 195.1 (1968), pp. 215–243. DOI: https://doi.org/10.1113/jphysiol.1968.sp008455.

[46] Howook Hwang et al. “Protein-protein docking benchmark version 4.0”. In: *Proteins, 2010, vol. 78* (2010), pp. 3111–3114.

[47] Andrea Ilari and Carmelinda Savino. “A Practical Approach to Protein Crystallography.” In: *Methods in molecular biology (Clifton, N.J.) vol. 1525* (2017), pp. 47–78.

[48] Joël Janin. “Assessing predictions of protein–protein interaction: The CAPRI experiment”. eng. In: *Protein science* 14.2 (2005), pp. 278–283.

[49] Joël Janin. “Wet and dry interfaces: the role of solvent in protein–protein and protein–DNA recognition”. eng. In: *Structure* 7.12 (1999), R277–R279.

[50] Joël Janin et al. “CAPRI: A Critical Assessment of PRedicted Interactions”. eng. In: *Proteins, structure, function, and bioinformatics* 52.1 (2003), pp. 2–9.

[51] Susan Jones and Janet M. Thornton. “Principles of Protein-Protein Interactions”. eng. In: *Proceedings of the National Academy of Sciences - PNAS* 93.1 (1996), pp. 13–20.

[52] Raed Khashan, Weifan Zheng, and Alexander Tropsha. “Scoring protein interaction decoys using exposed residues (SPIDER): A novel multibody interaction scoring function based on frequent geometric patterns of interfacial residues”. eng. In: *Proteins, structure, function, and bioinformatics* 80.9 (2012), pp. 2207–2217.

[53] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015.

[54] Laura J Kingsley et al. “Ranking protein–protein docking results using steered molecular dynamics and potential of mean force calculations”. eng. In: *Journal of computational chemistry* 37.20 (2016), pp. 1861–1865.

[55] Dima Kozakov et al. “The ClusPro web server for protein-protein docking”. eng. In: *Nature protocols* 12.2 (2017), pp. 255–278.

[56] Anders Krogh and John Hertz. “A Simple Weight Decay Can Improve Generalization”. In: *Advances in Neural Information Processing Systems*. Ed. by J. Moody, S. Hanson, and R. P. Lippmann. Vol. 4. Morgan-Kaufmann, 1992, pp. 950–957.

[57] Florian Krull et al. “ProPairs: A Data Set for Protein-Protein Docking”. eng. In: *Journal of chemical information and modeling* 55.7 (2015), pp. 1495–1507.

[58] Petras J Kundrotas et al. “Dockground: A comprehensive data resource for modeling of protein complexes”. eng. In: *Protein science* 27.1 (2018), pp. 172–181.

[59] David Lazer et al. “Life in the network: the coming age of computational social science”. eng. In: *Science (American Association for the Advancement of Science)* 323.5915 (2009), pp. 721–723.

[60] Y. LeCun et al. “Backpropagation Applied to Handwritten Zip Code Recognition”. In: *Neural Computation* 1.4 (1989), pp. 541–551. DOI: 10.1162/neco.1989.1.4.541.

[61] Y. Lecun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[62] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. “Self-attention graph pooling”. English. In: *36th International Conference on Machine Learning, ICML 2019*. 36th International Conference on Machine Learning, ICML 2019. International Machine Learning Society (IMLS), 2019, pp. 6661–6670.

[63] Marc F Lensink, Sameer Velankar, and Shoshana J Wodak. “Modeling protein-protein and protein-peptide complexes: CAPRI 6th edition: Modeling Protein-Protein and Protein-Peptide Complexes”. eng. In: *Proteins, structure, function, and bioinformatics* 85.3 (2017), pp. 359–377.

[64] Marc F Lensink and Shoshana J Wodak. “Docking, scoring, and affinity prediction in CAPRI”. eng. In: *Proteins, structure, function, and bioinformatics*. Proteins 81.12 (2013), pp. 2082–2095.

[65] Marc F Lensink and Shoshana J Wodak. “Score_set: A CAPRI benchmark for scoring protein complexes”. eng. In: *Proteins, structure, function, and bioinformatics* 82.11 (2014), pp. 3163–3169.

[66] Marc F Lensink et al. “The challenge of modeling protein assemblies: the CASP12-CAPRI experiment”. eng. In: *Proteins, structure, function, and bioinformatics* 86.S1 (2018), pp. 257–273.

[67] Li Li, Rong Chen, and Zhiping Weng. “RDOCK: Refinement of rigid-body protein docking predictions”. eng. In: *Proteins, structure, function, and bioinformatics* 53.3 (2003), pp. 693–707.

[68] Ye Li, Xianren Zhang, and Dapeng Cao. “The role of shape complementarity in the protein-protein interactions”. eng. In: *Scientific reports* 3.1 (2013), pp. 3271–3271.

[69] Shide Liang et al. “A simple reference state makes a significant improvement in near-native selections from structurally refined docking decoys”. eng. In: *Proteins, structure, function, and bioinformatics* 69.2 (2007), pp. 244–253.

[70] Shiyong Liu, Ying Gao, and Ilya A Vakser. “Dockground protein-protein docking decoy set”. eng. In: *Bioinformatics* 24.22 (2008), pp. 2634–2635.

[71] Raúl Méndez et al. “Assessment of CAPRI predictions in rounds 3–5 shows progress in docking procedures”. eng. In: *Proteins, structure, function, and bioinformatics* 60.2 (2005), pp. 150–169.

[72] C.L. Meyerkord and H. Fu. *Protein-protein interactions: Methods and applications: Second edition*. Jan. 2015, pp. 1–613. DOI: 10.1007/978-1-4939-2425-7.

[73] Julian Mintseris et al. “Integrating statistical pair potentials into protein complex prediction”. In: *Proteins, 2007, vol. 69* (2007), pp. 511–520.

[74] Julian Mintseris et al. “Protein–protein docking benchmark 2.0: An update”. eng. In: *Proteins, structure, function, and bioinformatics* 60.2 (2005), pp. 214–216.

[75] Iain H Moal et al. “The scoring of poses in protein-protein docking: current capabilities and future directions”. eng. In: *BMC bioinformatics* 14.1 (2013), pp. 286–286.

[76] N. Morgan and H. Bourlard. “Generalization and Parameter Estimation in Feedforward Nets: Some Experiments”. In: *Advances in Neural Information Processing Systems 2*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1990, pp. 630–637. ISBN: 1558601007.

[77] Andrew Y. Ng. “Feature Selection, L1 vs. L2 Regularization, and Rotational Invariance”. In: *Proceedings of the Twenty-First International Conference on Machine Learning*. ICML ’04. Banff, Alberta, Canada: Association for Computing Machinery, 2004, p. 78. ISBN: 1581138385. DOI: 10.1145/1015330.1015435.

[78] Marcin Novotni and Reinhard Klein. “3D zernike descriptors for content based shape retrieval”. eng. In: *Proceedings of the eighth ACM symposium on solid modeling and applications*. SM ’03. ACM, 2003, pp. 216–225. ISBN: 1581137060.

[79] Yanay Ofran and Burkhard Rost. “Analysing Six Types of Protein–Protein Interfaces”. eng. In: *Journal of molecular biology* 325.2 (2003), pp. 377–387.

[80] Hahnbeom Park, Hasup Lee, and Chaok Seok. “High-resolution protein–protein docking by global optimization: recent advances and future challenges”. eng. In: *Current opinion in structural biology* 35 (2015), pp. 24–31.

[81] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035.

[82] Eric F Pettersen et al. “UCSF Chimera—A visualization system for exploratory research and analysis”. eng. In: *Journal of computational chemistry* 25.13 (2004), pp. 1605–1612.

[83] Brian Pierce and Zhiping Weng. “A combination of rescoring and refinement significantly improves protein docking performance”. eng. In: *Proteins, structure, function, and bioinformatics* 72.1 (2008), pp. 270–279.

[84] Brian G Pierce, Yuichiro Hourai, and Zhiping Weng. “Accelerating protein docking in ZDOCK using an advanced 3D convolution library”. eng. In: *PloS one* 6.9 (2011), e24657–e24657.

[85] Ekagra Ranjan, Soumya Sanyal, and Partha P. Talukdar. “ASAP: Adaptive Structure Aware Pooling for Learning Hierarchical Graph Representations”. In: *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*. AAAI Press, 2020, pp. 5470–5477.

[86] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. “On the Convergence of Adam and Beyond”. In: *International Conference on Learning Representations*. 2018.

[87] R. Reed. “Pruning algorithms-a survey”. In: *IEEE Transactions on Neural Networks* 4.5 (1993), pp. 740–747. DOI: 10.1109/72.248452.

[88] David W Ritchie, Dima Kozakov, and Sandor Vajda. “Accelerating and focusing protein-protein docking correlations using multi-dimensional rotational FFT generating functions”. eng. In: *Bioinformatics* 24.17 (2008), pp. 1865–1873.

[89] Sebastian Ruder. “An overview of gradient descent optimization algorithms”. In: *arXiv preprint arXiv:1609.04747* (2016).

[90] D. Rumelhart, Geoffrey E. Hinton, and R. J. Williams. “Learning representations by back-propagating errors”. In: *Nature* 323 (1986), pp. 533–536.

[91] D Schneidman-Duhovny et al. “Taking geometry to its edge: fast unbound rigid (and hinge-bent) docking.” In: *Proteins 2003, 52* (2003), pp. 107–112.

[92] Z Shentu et al. “Context shapes: Efficient complementary shape matching for protein-protein docking.” In: *Proteins 2008, 70* (2008), pp. 1056–1073.

[93] Nitish Srivastava et al. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *J. Mach. Learn. Res.* 15.1 (Jan. 2014), pp. 1929–1958.

[94] Nitish Srivastava et al. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958.

[95] Andras Szilagyi and Yang Zhang. “Template-based structure modeling of protein–protein interactions”. eng. In: *Current opinion in structural biology* 24 (2014), pp. 10–23.

[96] Mingxing Tan and Quoc Le. “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks”. In: *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97. PMLR, Sept. 2019, pp. 6105–6114.

[97] Paul D Thomas and Ken A Dill. “Statistical Potentials Extracted From Protein Structures: How Accurate Are They?” eng. In: *Journal of molecular biology* 257.2 (1996), pp. 457–469.

[98] Mieczyslaw Torchala et al. “SwarmDock: a server for flexible protein-protein docking”. eng. In: *Bioinformatics (Oxford, England)* 29.6 (2013), pp. 807–809.

[99] Ashish Vaswani et al. “Attention is All you Need”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017.

[100] Petar Veličković et al. “Graph Attention Networks”. In: *International Conference on Learning Representations*. 2018.

[101] Vishwesh Venkatraman et al. “Protein-protein docking using region-based 3D Zernike descriptors”. eng. In: *BMC bioinformatics* 10.1 (2009), pp. 407–407.

[102] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. “Order matters: Sequence to sequence for sets”. In: *International Conference on Learning Representations (ICLR)*. 2016.

[103] Thom Vreven, Howook Hwang, and Zhiping Weng. “Integrating atom-based and residue-based scoring functions for protein–protein docking”. eng. In: *Protein science* 20.9 (2011), pp. 1576–1586.

[104] Thom Vreven et al. “Updates to the Integrated Protein–Protein Interaction Benchmarks: Docking Benchmark Version 5 and Affinity Benchmark Version 2”. In: *Journal of Molecular Biology* 427.19 (2015), pp. 3031–3041. DOI: https://doi.org/10.1016/j.jmb.2015.07.016.

[105] Thom Vreven et al. “Updates to the Integrated Protein–Protein Interaction Benchmarks: Docking Benchmark Version 5 and Affinity Benchmark Version 2”. eng. In: *Journal of molecular biology* 427.19 (2015), pp. 3031–3041.

[106] Alexander H. Waibel et al. “Phoneme recognition using time-delay neural networks”. In: *IEEE Trans. Acoust. Speech Signal Process.* 37 (1989), pp. 328–339.

[107] Hao-Ching Wang et al. “Staphylococcus aureus protein SAUGI acts as a uracil-DNA glycosylase inhibitor”. eng. In: *Nucleic acids research* 42.2 (2014), pp. 1354–1364.

[108] Hong-Wei Wang and Jia-Wei Wang. “How cryo-electron microscopy and X-ray crystallography complement each other”. In: *Protein Science* 26.1 (2017), pp. 32–39. DOI: https://doi.org/10.1002/pro.3022.

[109] Xiao Wang et al. “Protein docking model evaluation by 3D deep convolutional neural networks”. In: *Bioinformatics* 36.7 (Nov. 2019), pp. 2113–2118. DOI: 10.1093/bioinformatics/btz870.

[110] J. J. Weng, N. Ahuja, and T. S. Huang. “Learning recognition and segmentation of 3-D objects from 2-D images”. In: *1993 (4th) International Conference on Computer Vision*. 1993, pp. 121–128. DOI: 10.1109/ICCV.1993.378228.

[111] Z. Wu et al. “A Comprehensive Survey on Graph Neural Networks”. In: *IEEE Transactions on Neural Networks and Learning Systems* 32.1 (2021), pp. 4–24. DOI: 10.1109/TNNLS.2020.2978386.

[112] Changhui Yan et al. “Characterization of Protein–Protein Interfaces”. eng. In: *The Protein Journal* 27.1 (2008), pp. 59–70.

[113] Muhan Zhang et al. “An End-to-End Deep Learning Architecture for Graph Classification”. In: *AAAI*. 2018.

[114] Yang Zhang and Jeffrey Skolnick. “Scoring function for automated assessment of protein structure template quality”. eng. In: *Proteins, structure, function, and bioinformatics* 57.4 (2004), pp. 702–710.

[115] Hongyi Zhou and Jeffrey Skolnick. “GOAP: A Generalized Orientation-Dependent, All-Atom Statistical Potential for Protein Structure Prediction”. eng. In: *Biophysical journal* 101.8 (2011), pp. 2043–2052.

[116] Jie Zhou et al. *Graph Neural Networks: A Review of Methods and Applications*. cite arxiv:1812.08434. 2018.

Appendix A

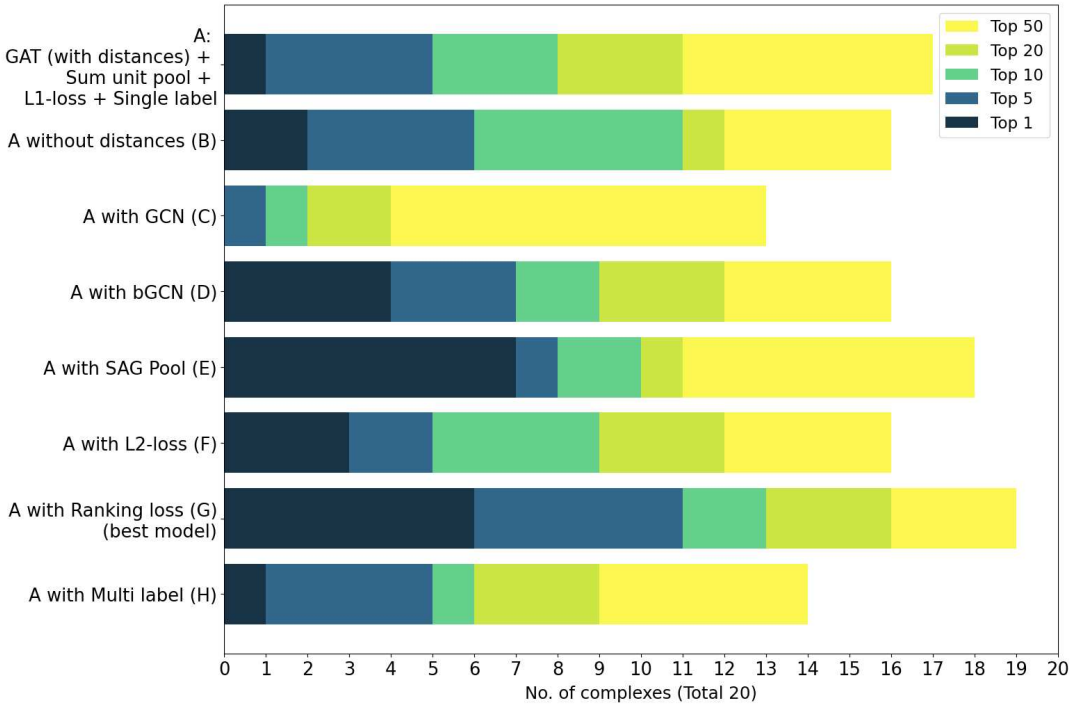
Supplementary Data

A.1 Results on validation set

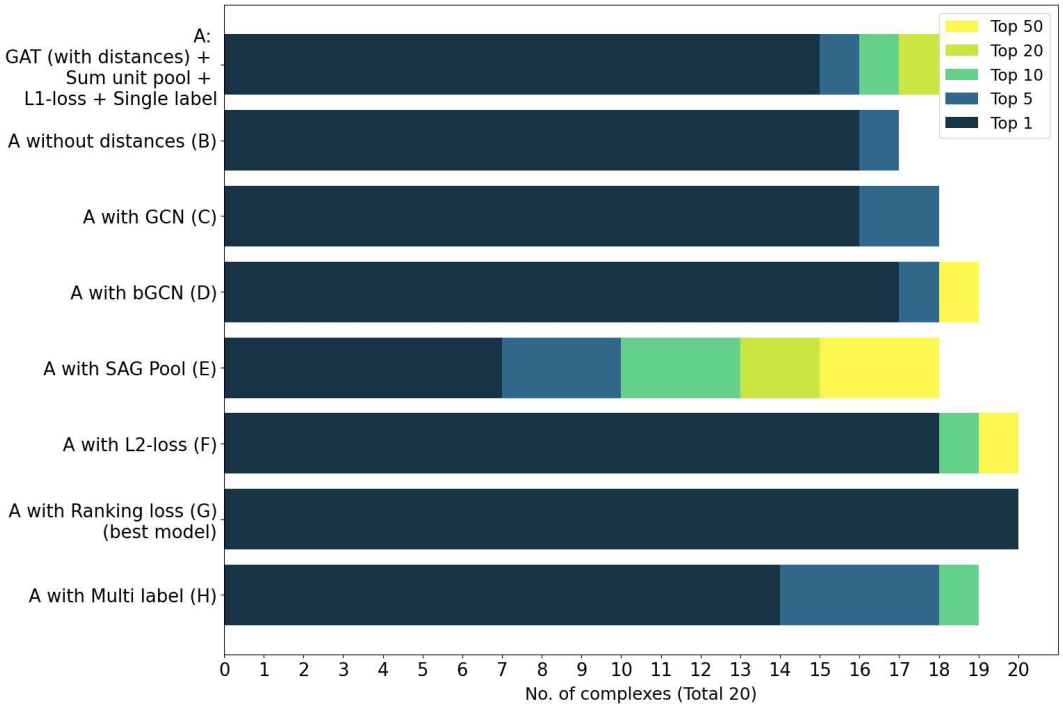
In Figure A.1 we examine the performance of the experiments listed in Table 7.1 on validation data to find the best model as mentioned in the model selection strategy (refer subsection 6.2.5). In top-n natives, experiment G performs the best in all top-1, 5, 10, 20, and 50 ranks. Comparing experiments in top-20 high quality and above docking decoys we observe experiment G performs the best. Therefore experiment G is considered as best model and is used in all comparisons on the test set in chapter 7.

A.2 Loss curve

The graph in Figure A.2 shows the loss values of the best model for each epoch. From the graph, we can infer that our model benefits from the training data until epoch 75 and flattens out afterward, indicating a possibility of overfitting. This supports our model selection strategy that selected epoch 62 as the best epoch.



(a)



(b)

Figure A.1: Comparing top-n ranking performance of the experiments listed in Table 7.1 on validation set. Reported are the number of complexes having (a) high quality and above decoys and (b) natives, in top-p ranks.

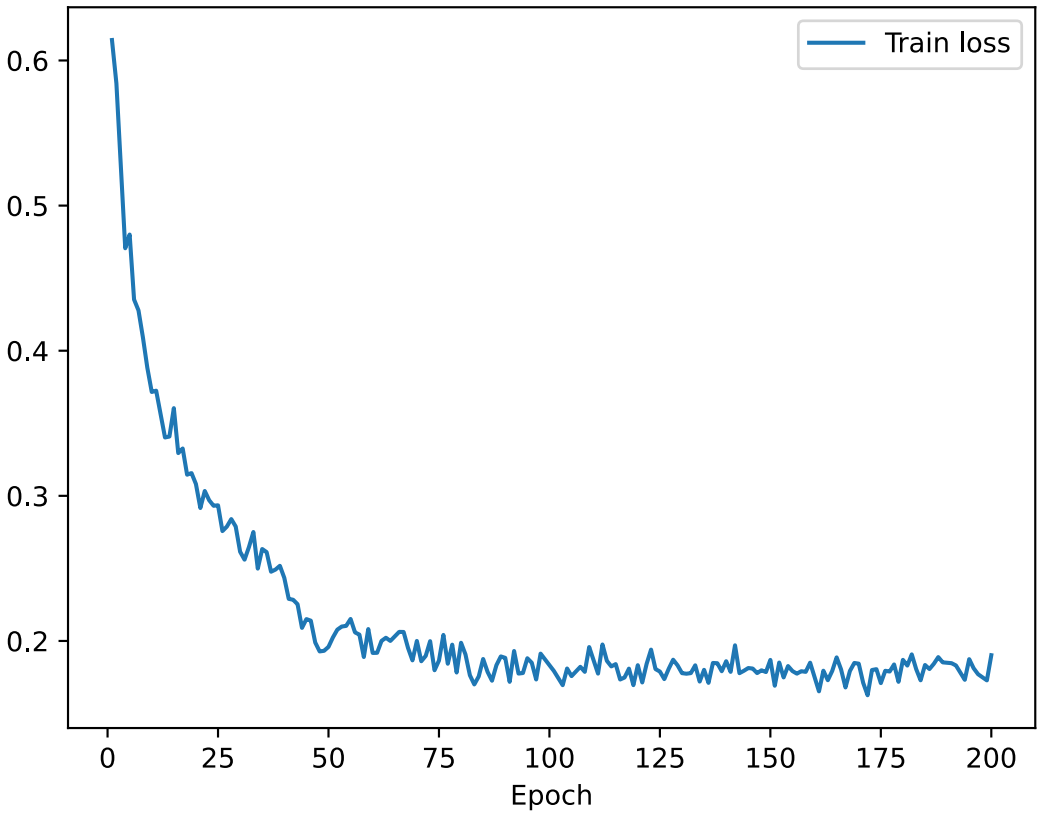


Figure A.2: Loss curve of the best model during training.