THESIS

QUALITY ASSESSMENT OF DOCKED PROTEIN INTERFACES USING 3D CONVOLUTION

Submitted by

Mridula Bontha

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Spring 2021

Master's Committee:

Advisor: Asa Ben-Hur

J Ross Beveridge
Emily J. King

ABSTRACT

QUALITY ASSESSMENT OF DOCKED PROTEIN INTERFACES USING 3D CONVOLUTION

Proteins play a vital role in most biological processes, most of which occur through interactions between proteins. When proteins interact they form a complex, whose functionality is different from the individual proteins in the complex. Therefore understanding protein interactions and their interfaces is an important problem. Experimental methods for this task are expensive and time consuming, which has led to the development of docking methods for predicting the structures of protein complexes. These methods produce a large number of potential solutions, and the energy functions used in these methods are not good enough to find solutions that are close to the native state of the complex. Deep learning and its ability to model complex problems has opened up the opportunity to model protein complexes and learn from scratch how to rank docking solutions. As a part of this work, we have developed a 3D convolutional network approach that uses raw atomic densities to address this problem. Our method achieves performance which is on par with state-of-art methods. We have evaluated our model on docked protein structures simulated from four docking tools namely ZDOCK, HADDOCK, FRODOCK and ClusPro on targets from Docking Benchmark Data version 5 (DBD5).

ACKNOWLEDGEMENTS

I would firstly like to thank god for providing me the strength to overcome all the hurdles during my masters. Thanks to my parents for their love and encouragement throughout my life. Without their support, it would not have been possible to achieve my goals. I want to thank my advisor, Dr. Asa Ben-Hur, for providing me the valuable advice to progress with my research and continually motivating me to come up with creative ideas. I would also like to thank my colleagues Yashwanth, Don, and Fahad, whose valuable feedback has helped me to make my work more presentable. I would extend my gratitude to the Department of Computer Science at CSU and the CSU Graduate School for initiating, commissioning, and supporting this project. I am thankful to my friend Uday Ranparia for constantly supporting me through the hard times and helping me overcome the stressfull times. Finally, I would like to thank Leif Anderson, who created and supported the previous LaTeX template for several years.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Protein-protein interactions play a crucial role in cellular mechanisms of all living organisms. Any deviation from regular protein-protein interactions can lead to disease, which makes them an important object of study. Protein-protein interactions result from physical contact between two or more proteins. The resulting interface of the protein complex can be determined accurately with experimental methods but these experimental methods are time consuming and expensive. This brings a need for development of computational methods. Docking is one such simulation method to predict the possible bound structure of the complex resulting from the interaction of the participating proteins. A docking simulation leads to a large number of possible solutions, which creates a need for ranking them according to their proximity to the native structure. There are many docking tools available today, and all of them are associated with a scoring function for assessing the quality of the interface.

As a part of our research, we propose and evaluate a simple approach to represent the interface of a complex and use 3D convolutional networks to determine the quality of the predicted interface. With the emergence of Convolution Neural Networks (CNNs), there has been a massive shift in the field of machine learning. The availability of improved computing resources and a vast amount of labeled image data has facilitated the usage of CNNs in computer vision. Inspired by the dramatic success of CNNs in computer vision, We are borrowing the idea of representing a protein-protein interface as a 3D grid of atomic densities of the constituting atoms; similar to a 3D grid representation of RGB values of the pixels composing the image.

This thesis presents a computational technique for the quality assessment of a docked protein-protein structure. Recent work has demonstrated the potential for modeling protein 3D structure using 3D convolutional neural networks. The motivation behind these methods is to avoid having to engineer features, and instead, rely on the network to learn relevant features directly from the 3D structure of the protein.

The outline of the thesis includes introduction to proteins, protein-protein interactions and how are they studied. Chapter 2 discusses various experimental and computational methods for studying protein docking. Chapter 3 explains previous applications of machine learning and deep Convolutional Neural Networks. Chapter 4 discusses some existing 3D-CNN bases methods for modeling protein structures. Chapter 5 describes the proposed methodology. Chapter 6 explains the experimental setup, which includes details about data sources, and data modeling. Chapter 7 discusses results of various experiments carried out as a part of this thesis. Finally, Chapter 8 digs into some of the potential areas of future work.

1.1 Proteins

Proteins are complex, large molecules that are responsible for various crucial activities in the body. Body functions such as, regulation of tissues and organs occur due to the interaction of proteins at the cellular level [20]. Enzymes are proteins that accelerate the chemical reactions taking place in the body. For example, enzymes help in digesting the food and regulating metabolism. Hormones are proteins that help coordinate by message passing between various parts of the body [21]. The phenomenon mentioned above occurs as a result of the continuous production of the proteins inside the cells. This process is known as protein synthesis, and it involves two significant steps. The first step is the transcription; during transcription, the enzyme RNA polymerase utilizes the DNA as a template to produce a pre-mRNA transcript. This pre-mRNA is further processed to form a more developed mRNA molecule. The second step is a translation, where mRNA gets translated into an amino acid sequence of the protein (See fig 1.1 for reference).

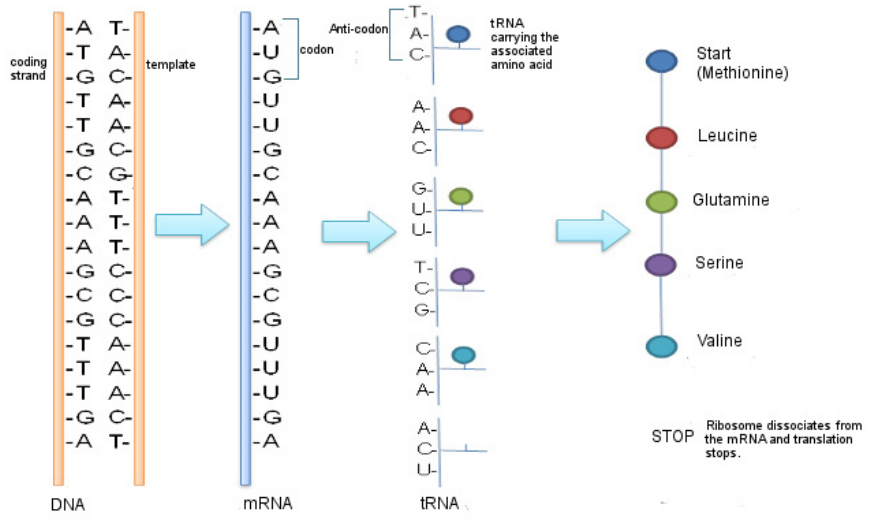


Figure 1.1: Diagram of the central dogma, DNA to RNA to protein, illustrating the genetic code in hemoglobin protein. This happens to be the first few amino acids for the alpha subunit of hemoglobin. The sixth amino acid here (glutamic acid, "E") is mutated in sickle cell anemia versions of the molecule [2].

Amino acids are compounds that contain an alpha carbon at the center surrounded by an amine group, a hydrogen atom, and a carboxyl group and a sidechain (See fig 1.2 for reference). Amino-acids thread into polypeptide-chain as a result of linkage between an α-carbon atom of one amino-acid and an α-carbon of another amino-acid through the formation of a peptide bond [20]. A polypeptide-chain comprises a recurring unit called the backbone and a variable unit called the sidechain. The amino acid units participating in the polypeptide chain determine the sidechain. There are twenty-one different types of amino acids that could substitute sidechain and result in long amino acid sequences. Sidechains that are oppositely charged tend to attract each other. For example, hydrophilic side chains attract to sidechains with water molecules, and hydrophobic sidechains would repel from water molecules. The interactions between amino acids lead to the folding of amino acid chains, which attribute to the functioning and the three-dimensional structure of the comprising protein [22].

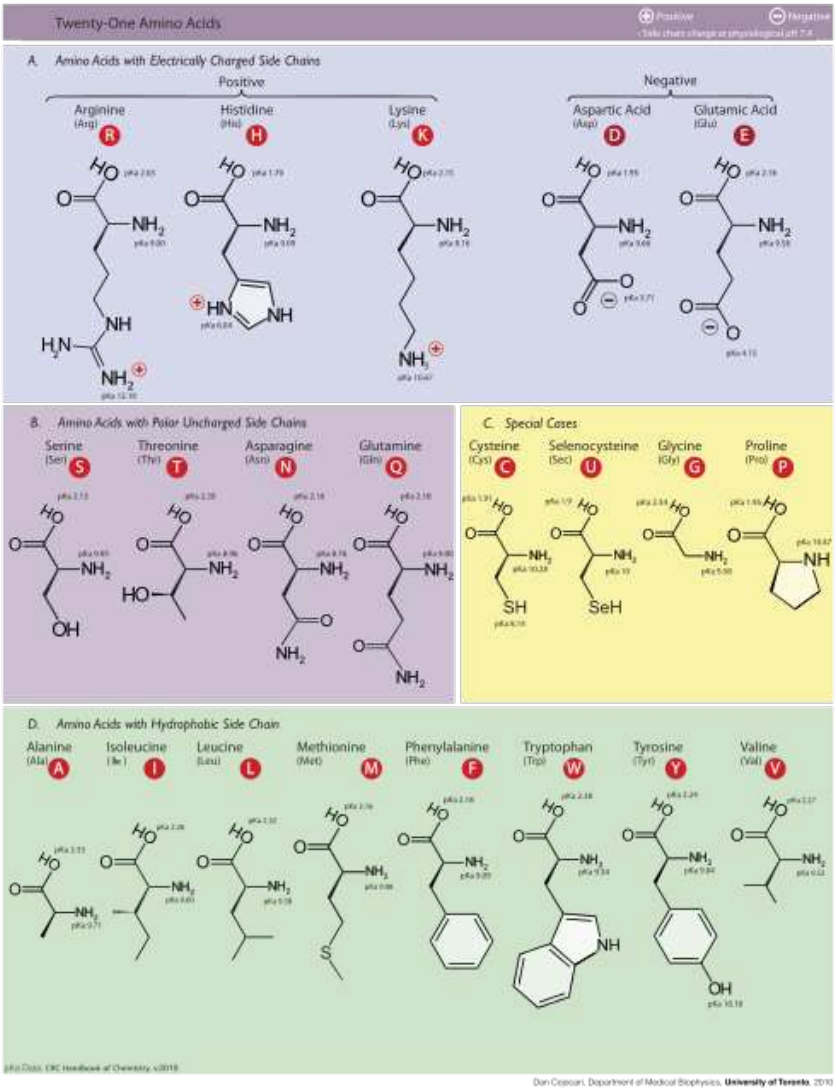


Figure 1.2: There are 21 types of amino acid encoded in eukaryotic DNA which are categorized by their electrical properties. The image provides the abbrebated and full-names of each of these amino acids. The structures of these compounds are oriented in such a way that the carboxyl group, α-carbon, and amine groups are on top, with the side chain extending downwards.

1.1.1 3-Dimensional structure of proteins

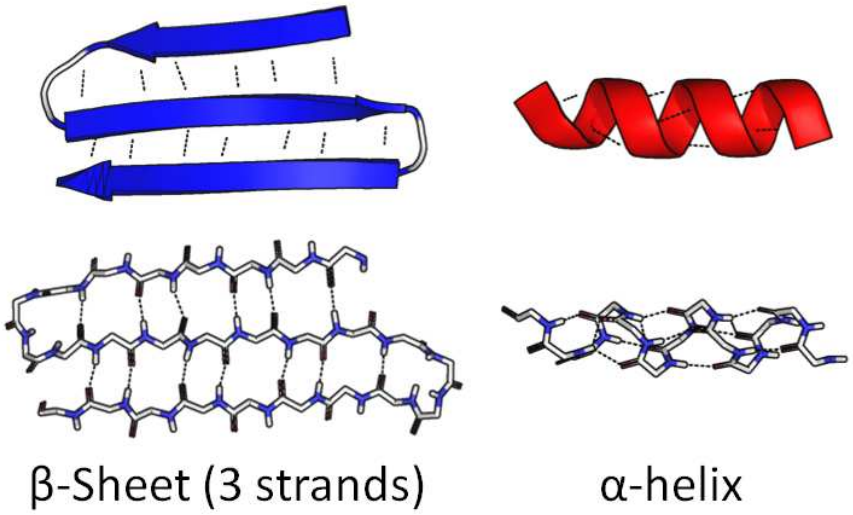


Figure 1.3: α-helix and β-sheets are the fundamental secondary structural elements in proteins. Image taken from [3].

The structure of a protein can be studied at three levels: namely primary structure, secondary structure, and tertiary structure. The amino acid sequence represents the primary structure. The secondary structure is the result of local interactions of the segments of the amino acid chain, and the tertiary structure is the result of the folding of the amino acid chain. α-helix and β-pleated sheets are the primary types of secondary structures [22] (See fig 1.3). The shape of both of these structures results from the hydrogen bonds forming between the amine group of one of the amino acids and carboxyl, another amino acid.

As depicted in figure 1.3, an α-helix results due to the bonding between the carbonyl of one of the residues with the amide group of an amino acid, which is four residues farther over the peptide chain. The helical turn in an α-helix structure, which also resembles a coiled ribbon is due to the repeated pattern of the bond between carboxyl and amide group between every four residues [21]. Each turn in an α-helix roughly consists of 3.6 residues. An α-helix is not stable by itself. It achieves a more stabilized structure through linking with other secondary structural elements.

On the other hand, a β-sheet results from the side by side alignment of two fragments of a polypeptide chain. A β-sheet attains its name due to its resemblance to a sheet-like structure knitted together with hydrogen bonds. The fragments of a β-sheet may be aligned parallel or anti-parallel (See Figure 1.4).

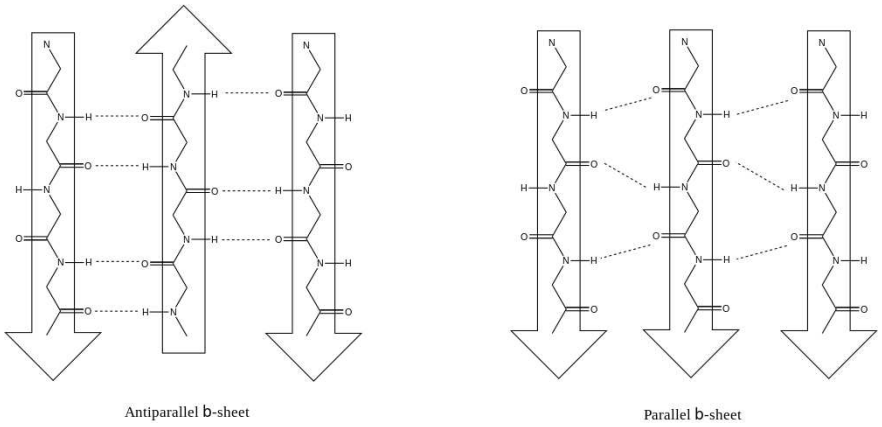


Figure 1.4: Parallel and Antiparallel β-strands [4]

The tertiary structure of a protein is determined by the interaction between elements of its secondary structure. In other words, a tertiary structure results from folding of different fragments of sheets and helices into a three-dimensional shape. A protein's most stable structure under a given physiological condition is called its native state [23]. Fig 1.5 represents a schematic diagram of various levels of the protein structure.

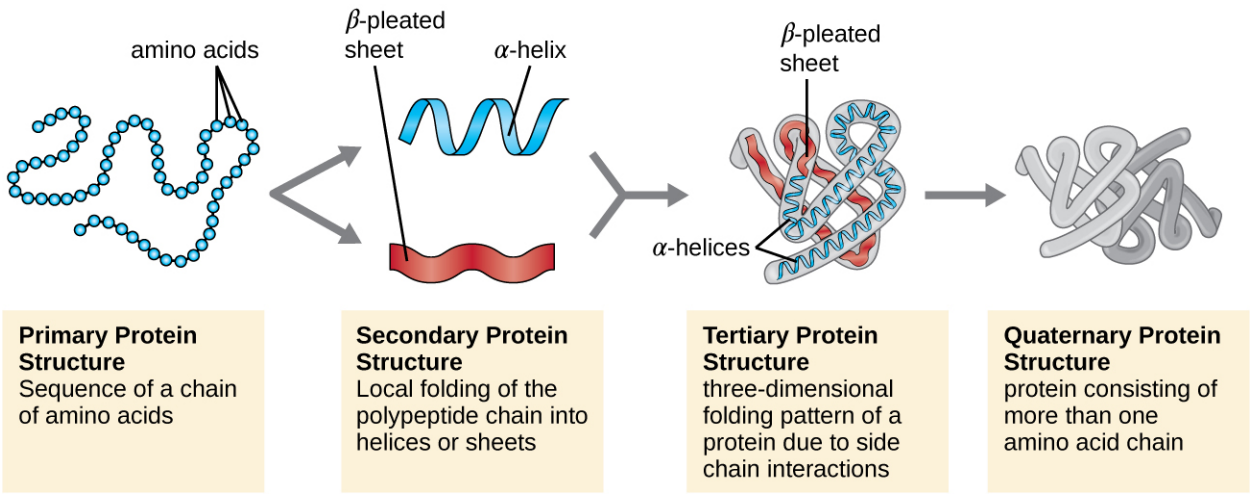


Figure 1.5: The four levels of protein structure. The final level of a protein structure is its quaternary structure that results from interaction of multiple protein structures. [5]

Finally, the quaternary structure results from the interaction of multiple protein chains that interact to form a complex. The resulting complex is referred to as the quaternary structure and each of the interacting protein molecules is called a subunit. Each of these subunits not only has their own primary sequence, secondary and tertiary structures but also a significant functionality that contributes to the overall functioning of the resulting complex. One such example of a complex is hemoglobin in which one of the proteins in the complex interacts with oxygen and the other one with carbon dioxide. This phenomenon of protein-protein interactions underly the millions of cellular level activities taking place in the body.

1.1.2 Protein-Protein interactions

As discussed in the previous section, a quaternary structure of a complex protein is a union of multiple protein chains in a compactly packed arrangement. Such an arrangement is a result of physical contact between atoms of the participating protein chains. Physical contacts here refer to interactions like electrostatic forces, hydrophobic or hydrophilic effects, which is driven by biochemical events. [24]. The phenomenon of interaction is known as protein-protein interaction (abbreviated as PPI), and the region of interaction is called a protein-protein interface.

Protein-protein interactions play a crucial role in many biological activities taking place in the body. It is important for the subunits to be organized in a particular fashion for the proper functioning of the entire unit. Any disruption in such interactions may lead to changes in the biological activity of the whole unit. This is often the case in a disease state. In a disease state there is often an alteration of the interaction network, leading to a change in the functioning of other proteins [25].

A close study of protein-protein interactions can yield vital information that could be useful for both identifying any abnormal body condition as well as for finding better curative drugs for treating such conditions [26].

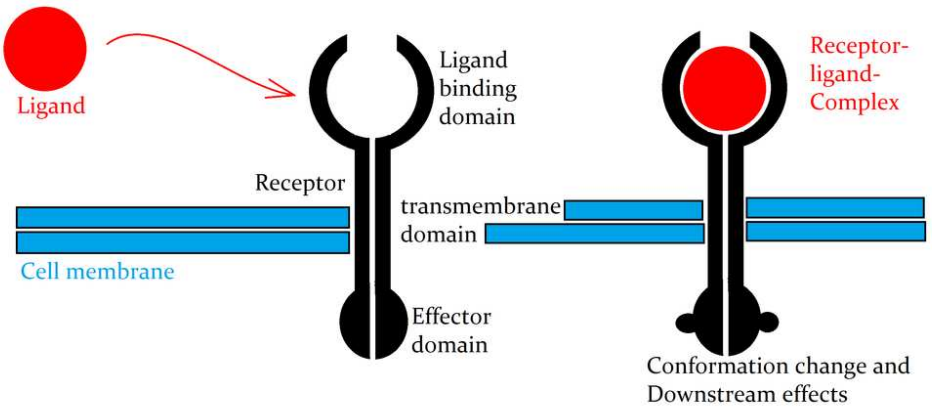


Figure 1.6: A ligand interacting with the receptor at the binding site. [6]

In a protein-protein complex, the partner with the longest amino acid chain is known as a **receptor**, whereas the shorter one is known as a **ligand**. Fig 1.6 represents an interaction between a protein and a ligand. The formation of protein complex results from the binding of a high affinity ligand at an adequate pose at the binding site of the protein [27]. Here, affinity refers to the tendency of binding between proteins as a result of chemically attractive force and the binding site refers to a small pocket like crevice in the 3D-structure of the protein. Here, a ligand could refer to an ion, a drug molecule, or another protein. This thesis focuses on interactions where both the receptor and ligands are proteins.

Several factors determine the binding affinity between proteins. Some of these factors include micro-environments that catalyze the protein-protein interaction, shape, and accessibility of the surfaces participating in the interaction.

The forces between the atoms of that are part of the interface determine the stability of the resulting complex. In general, a residue is considered as a part of the interface if it is in a direct contact with the residues of the partner protein. Here the term *contact* refers to any kind of physical or chemical force acting between the atoms constituing the residues at the interface. Furthermore, protein complexes can be categorized as homomers and heteromers based on the extent of similarity between the partnering proteins. A complex is said to be a homomer if it is composed of identical partners; otherwise, it is classified as a heteromer if it is composed of several different proteins.

Proteins can also be classified based on their ability to exist independently. Obligate complexes have partnering proteins that are quite unstable individually but attain stability when they partner with each other. Non-obligate or transient complexes are composed of proteins that can exist individually. Obligate interactions are usually perpetual, whereas non-obligate interactions are transient [28]. In this thesis we focus on non-obligate complexes.

Chapter 2

Methods to study protein-protein interactions

Many techniques have been developed for studying the interactions between the partnering proteins in a protein complex. In what follows we describe both experimental and computational methods.

2.1 Experimental methods

2.1.1 X-ray Crystallography

X-ray crystallography is the most accurate technique for discovering the structure of a protein. When applied to a complex, it provides detailed information about the interaction and its interface. X-ray crystallography uses of x-rays to diffract over the crystal structure of a protein or a complex to obtain extreme high-resolution microscopic information of its 3-dimensional structure. The wavelength of X-rays ranges from 0.1 nm to 1Å, which is roughly equal to the distance between interacting atoms in a molecule. The process begins with first purifying a protein sample crystallized at a higher concentration. The crystal and loop are cooled under constant liquid nitrogen flow to circumvent any impairment to the protein crystal. The atom of the crystal, when struck with an X-ray beam, produce a diffraction pattern [29]. A ray of X-ray is run through the crystallized protein, and a detector records the diffraction pattern. Several such diffraction patterns are recorded by rotating the crystal and directing the rays at different positions. An overlap between the incoming ray and an electron creates a secondary wave. This diffracted wave has the same wavelength as that of the incoming ray but in a different direction. These diffraction patterns are then used later to construct a protein structure or its interactions with other proteins in a protein complex. To build an electron density map from the various diffraction patterns, each point in these diffraction patterns is indexed, integrated, merged, and scaled, resulting in a single record for all the diffraction patterns associated with a structure [30].

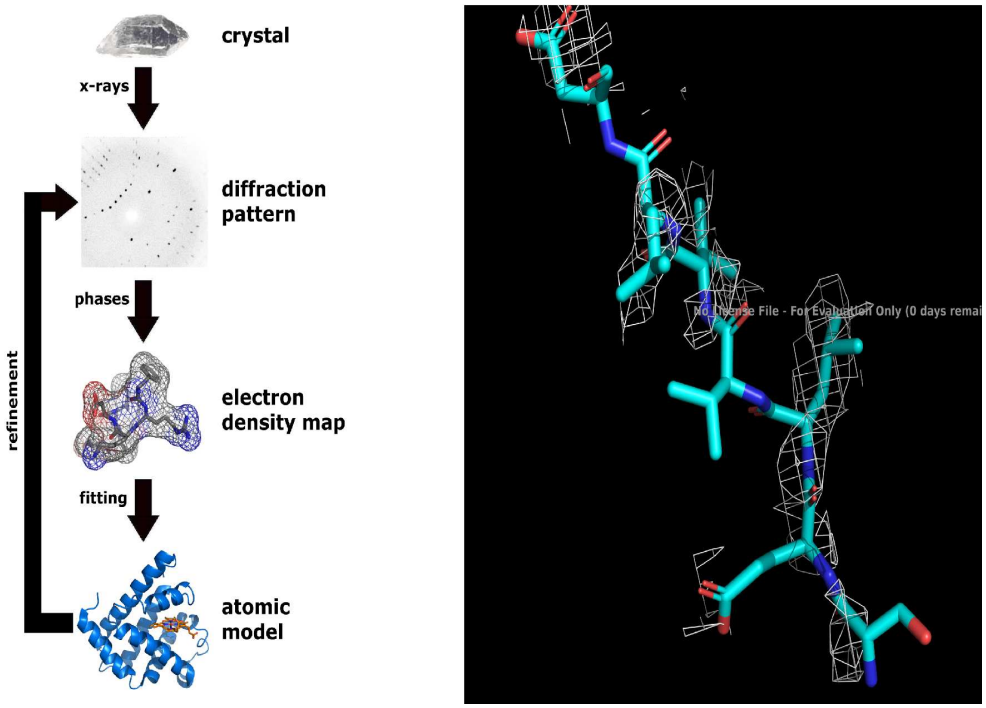


Figure 2.1: A schematic diagram representing X-ray crystallography [7] and A schematic diagram of a cross-section of protein 2BIW electron density map rendered using PyMol [8]

2.1.2 NMR spectroscopy

In the world of biology, some proteins may not crystallize or may not crystallize in a biologically relevant conformation. Such cases usually occur when the interactions between the proteins are weak. For such cases, X-ray crystallography may not be the right choice to study protein-protein interactions. In such instances, NMR spectroscopy may prove to a potential choice because it is capable of determining the three-dimensional structure of a protein complex without the need for crystallization. NMR spectroscopy of proteins involves the application of magnetic resonance spectroscopy to inspect the local magnetic fields around the nucleus of an atom. This method yields structure not only related information but also the composition of nucleic acids in the protein. The first step in this process is the sample preparation. An aqueous sample of highly purified protein is prepared and dissolved in a buffer solution. This sample is then placed in a strong magnetic field generated through radio frequency signals. The amount of penetration of these signals by the sample is measured, which yields the distances between atomic nuclei [31]. The shift in the atomic

nuclei's energy states is observed when the spin of the nucleus gets interrupted in the presence of a magnetic field.

2.2 Computational methods

Crystallographic methods provide the most accurate results, but most of these techniques are expensive, require a considerable amount of manual effort, and are not always successful. The limitations of experimental techniques bring in the need for computational methods.

The study of protein-protein interface prediction can be framed in two ways : partner-independent and partner-specific. The partner-independent technique focuses on only one of the partnering proteins. It involves the inspection of residues from the protein of interest which are interacting with the partnering protein. The partner-specific technique considers both the partnering proteins and includes assessing a pair of residues for their ability to be a part of the interface. Some of the most common protein-protein interface prediction techniques include template matching, machine learning-based methods, and docking.

Template matching and machine learning methods rely heavily on a pre-computed dataset of known interacting and non-interacting protein pairs [32]. These pairs constitute the positive and negative samples in the dataset. While creating a positive database is pretty straight forward, it is quite tricky to create a negative sample set. One of the most commonly used techniques to create negative datasets is pairing proteins from different cellular locations or randomly pairing proteins from the positive dataset while ignoring the interacting ones. The template-based matching methods utilize the similarity between the queried protein pairs' structure to a known protein pair database. For two proteins X and Y interacting with each other, If there exists a protein X' which is structurally similar to X and a protein Y', which is similar to protein Y, it suggests that proteins X' and Y' will also interact.

For machine learning methods, several ideas have been proposed using neural networks and SVMs [33]. In BLRM: A Basic Linear Ranking Model for Protein Interface Prediction [34], Shariat, Neumann, and Ben-Hur proposed a methodology for representing the docked interface.

Each residue pair from ligand and receptor chains are represented as a vector. The motivation here is to learn a ranking function mapped to these residue pairs. Further, the machine learning task is modeled as a classification problem where a positive prediction denotes a high-quality interface, and a negative prediction indicates a low-quality interface. Until the arrival of deep learning, machine learning methods for the study of protein interfaces required a major effort for feature engineering, i.e. design of characteristics of the protein structure that are predictive of binding.

In what follows we first describe docking methods which use the 3D conformation of the two partner proteins to predict the resulting protein complex.

2.2.1 Docking

Docking or molecular docking is a modeling technique that predicts the structure of a protein complex from its partners. Here, the partnering molecules could be a protein, ion, or a drug molecule. A docking problem can be considered analogous to putting together pieces of a jigsaw puzzle [35]. The most significant intermolecular forces between interacting atoms at the interface include electrical forces, Van Der Waals forces, and the strength of the Hydrogen bond. These forces are used to determine a score that measures the quality of the docked solution. This score, or energy function is the objective function used for predicting a large number of candidate solutions for the protein complex.

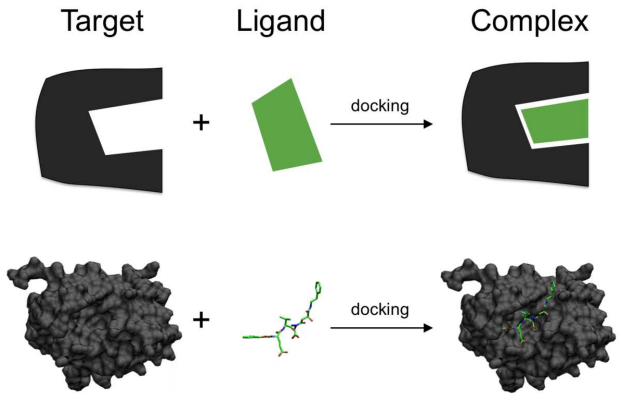


Figure 2.2: Docking as an analogy to fitting correct pieces of puzzle together. The part of the protein at which ligand attaches is known as binding site. A single protein can have multiple binding pockets each of which is associated with a binding-affinity for the docking ligand [9].

Docking techniques take the unbound structures of the interacting molecules to determine the possible bound configurations. Different docking tools use different scoring functions to assess the stability of a simulated solution. An unbound structure of a ligand or receptor refers to its free structure while in a non-interacting state. On the other hand, a bound structure of the same refers to its three-dimensional orientation while in an interacting state. The design of a protein-protein docking software can be considered as the prediction of complex structure given the structure of the individual unbound proteins. Figure 2.3 gives a schematic overview of the components that make up most docking methods. Let us discuss each of them in detail.

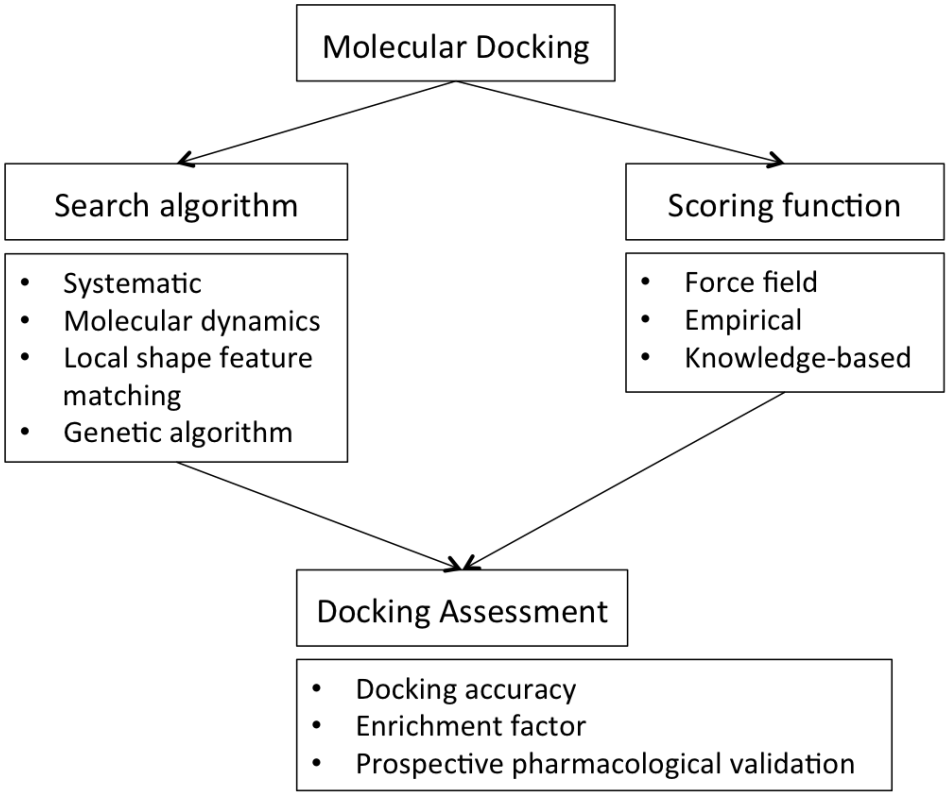


Figure 2.3: Components of docking methods [10].

2.2.1.1 Determining the structures

To perform docking, the first requirement is a three-dimensional structure of the participating proteins. The structures are usually determined using X-ray crystallography or NMR spectroscopy.

2.2.1.2 Exploring the search space

The search space is a set of all possible orientations of the ligand and the receptor relative to each other. In reality, it is impossible to exhaust the search space because it takes into account all possible translations and rotations of each of the partners. The size of the search space could be reduced by considering the entire conformation space of the ligand or vice-versa and modeling various receptor poses. Docking methods can be classified as rigid-body and flexible-body docking, depending on whether they allow for conformational changes in the two partners.

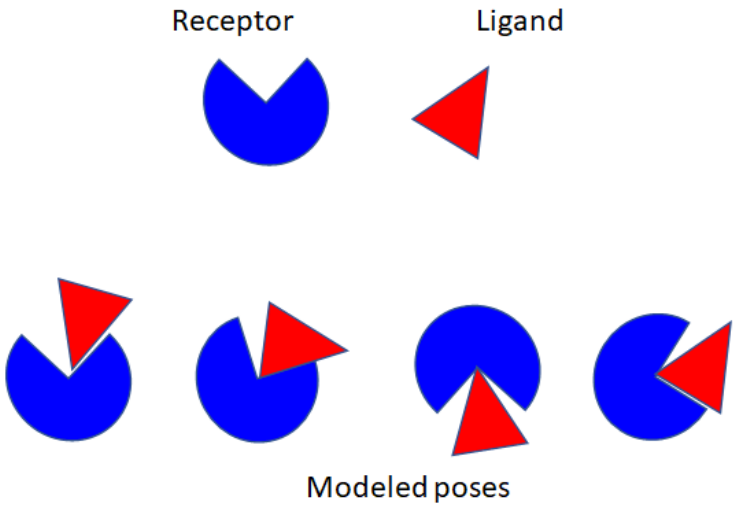


Figure 2.4: Docking mechanism - the receptor is shown in red and the ligand is shown in blue. Docking can be seen analogous to fitting two pieces of a jigsaw puzzle. These configurations are assessed using docking scores.

Fig 2.4 shows a schematic representation of rigid vs flexible docking. A rigid body docking does not allow alteration of bond angles, bond lengths, and torsion angles of one or both of the components in the docking process. In other words, a rigid-body docking algorithm performs docking by keeping either of the receptor or ligand (usually the receptor) as a rigid body and looking for possible orientation of the other component in the conformational search space. Rigid-body docking is more straightforward and faster; however, in reality, most of the protein-protein complexes are formed as a result of conformational changes in the interacting components. Flexible-body docking takes into account the conformational changes. Flexible docking allows complete move-

ment of the ligand and receptor in 3D space as well as movement of individual fragments of each of the components and docking these fragments to obtain the bound conformation.

2.2.1.3 Scoring functions

Exploring the search space yields many potential solutions, and almost all of them are not good. A docked solution is considered good if it can exist as a stable structure. To asses the quality of a docked solution produced by a docking algorithm, a scoring function is used. Most of the scoring functions are base on molecular level force fields that determine the energy of a simulated pose within the binding site. A lower energy implies a more stable configuration and thus a higher binding affinity between the two proteins after docking. Apart from energy functions, scoring functions can also be constructed based on the type of interactions between binding partners, statistical observations based in inter-molecular contacts as well as using machine learning where the functional form is inferred directly from the data [36].

2.2.1.4 Assessment of docking solutions

The final step in the docking pipeline is assessing the docked solutions using the scoring function formulated in the above step. This evaluation can be performed against many docking benchmark datasets. One of the most commonly referred docking datasets is DBD (Docking Benchmark Dataset). CAPRI (abbreviated for Critical Assessment of PRedicted Interactions) is a community-wide initiative for evaluating computational algorithms for predictions of experimentally determined 3D structures of protein complexes [37].

2.2.2 Statistical and Machine Learning methods

The problem of protein-protein interface prediction can be treated as a machine-learning classification problem if we can represent the interface with some features that describe the environment around it. Once an interface is modeled to fit as input for a machine learning algorithm, a number of methods such as SVM or Logistic Regression can be easily applied to predict if it is a true in-

terface or not. The following paragraph gives an overview of a number of machine-learning tools and techniques developed to predict a protein-protein interface.

SPPIDER uses the predicted and actual relative solvent accessibility differences for a residue participating at the interface [38]. SPPIDER's architecture consists of 10 output neurons. Each of these neurons produces a probability, and the final likelihood is calculated, considering the majority of votes from the ten results. While SPPIDER is a binding site classifier, PINUP is an algorithm for binding site prediction in monomer proteins that uses a scoring function based on a linear combination of the residue-energy score, residue conservation score, and residue interface propensity while optimizing two weight parameters [39]. PINUP considers an unbounded structure of a protein in PDB format and outputs predicted interface residues as well as the hotspot residues. Another similar method is Cons-PPISP, which is developed for predicting protein-protein interaction site detection [40]. For any protein structure, Cons-PPISP predicts residues that are a high likelihood to form an interface with another protein. The neural network input includes solvent accessibilities and position-explicit details of a residue in the interface region. The output is the probability of that residue to be a part of the binding site. A couple of methods have also been developed that use the combined score of the existing structure-based methods to predict a more refined probability of binding sites. Meta-PPISP [41] and CPORT [42] are two such methods that combine the raw scores from Cons-PPISP, PINUP, ProMate [39, 40, 43] and WHISCY, PIER, ProMate, cons-PPISP, SPPIDER, PINUP, [38–40, 43–45] respectively.

Much of the summary in the above section is compiled from article Computational prediction of protein interfaces: A review of datadriven methods [46]

2.3 Docking softwares used for constructing the dataset

As a part of this thesis, we have used docked-decoys from four docking tools namely ZDOCK, Frodock, ClusPro and HADDOCK. Each of them is discussed in detail next.

2.3.1 ZDOCK

ZDOCK is a protein-protein docking software available as both web-based serves as well as a standalone software that uses a rigid-body docking technique to predict structures of protein-protein complexes. ZDOCK uses the Fast Fourier Transform (FFT) to search across all possible binding models based on pairwise shape complementarity, electrostatic potential energy and desolvation energy [47]. ZDOCK is available both as a webserver and a stand alone software, and the predicted modeles are ranked according to the predicted score. A lower score corresponds to a higher rank.

2.3.2 Frodock

Frodock is a rigid body docking algorithm that utilizes 3D grid-based potentials and spherical harmonics approximations to boost the speed of the search process [48]. The resulting docked solution's binding energy is approximated as a correlation function based on the van der Waals, electrostatics, and the desolvation potential. The interaction energy is calculated using a fast and exhaustive rotational docking search complemented with a simple translational search. Similar to ZDOCK, Frodock is available as both an online web server and standalone software.

2.3.3 ClusPro

ClusPro is another automated rigid-body docking algorithm for predicting protein-protein interaction complexes. ClusPro is a fully automated web-based docking tool that accepts two proteins to be docking in their PDB formats and generates docking solutions clustered based on their electrostatic and desolvation free energies [49]. ClusPro uses the Fourier correlation algorithm for quickly filtering the pool of near-native solutions obtained. This algorithm uses a combination of desolvation and electrostatic potential, which is determined using Coulombic potential. Further, to reduce false-positives, the previous step's acceptable structures are clustered based on the pairwise RMSD between the calculated and the native binding-site.

2.3.4 HADDOCK

HADDOCK (High Ambiguity Driven protein-protein DOcKing) uses the information at the biochemical and biophysical level like chemical shift perturbation resulting from NMR titration and mutagenesis experiments [50]. This information is referred to as Ambiguous Interaction Restraints that steer the docking process. These restraints are calculated as ambiguous distances between all the residues participating in the interaction. Using these defined Ambiguous Interaction Restraints, HADDOCK searches through all possible solutions around the binding site to determine the most promising pair of residues from the passive and active residues. Here, the active residues refer to the residues that exhibit a considerable chemical shift perturbation and bear high solvent accessibility after the complex formation. Similar to ZDOCK and Frodock, HADDOCK is also available as a web-based tool and standalone software.

2.4 Assessment of docking solutions with DockQ score

DockQ is a continuous protein-protein docking model quality standard given as a function of three independent CAPRI standards, namely F_{nat}, $iRMS$, and $LRMS$ (described in detail below). The DockQ score can be used to determine the quality of a protein-protein decoy structure. While CAPRI uses each of the above-stated standards to assess a decoy structure's overall quality with respect to its native structure, DockQ serves as a single standard that considers all these standards. Fig 2.5 gives a schematic representation of the calculation of each of the parameters involved in determining the DockQ score.

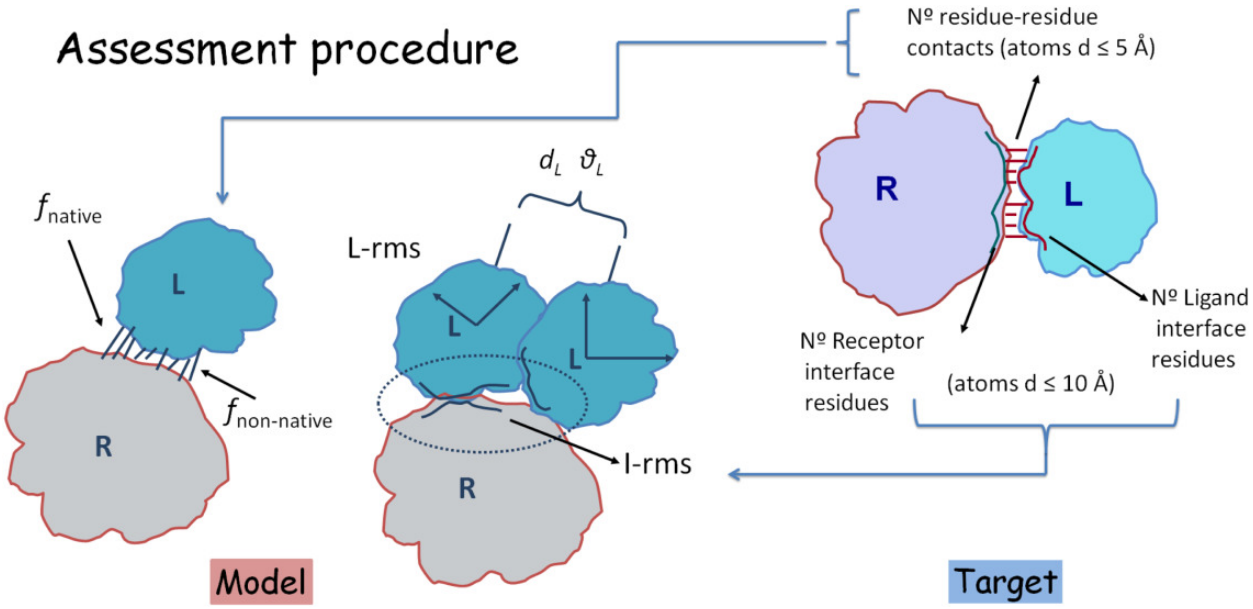


Figure 2.5: The different standards used for DockQ score calculation as defined by CAPRI [11].

For a given model and a target structure, with the receptor referenced as **R** and ligand as **L** we determine F_{nat}, $iRMS$, and $LRMS$ as follows:

1. F_{nat}: The ratio of total correct residue-residue contacts in the model structure interface and the total number of residue-residue contacts in the target structure interface.

2. $LRMS$: The backbone root-mean-square displacement of the ligand in the model structure with respect to the target structure's ligand, when the model ligand is superimposed over the target ligand.

3. $iRMS$: The backbone root-mean-square displacement of the interface residues in the model structure with respect to the target structure's interface residues, given that the model interface is superimposed over the target interface.

For calculating the $iRMS$, the interface is extracted by considering only the atoms from both the interacting partners which are at most 10Ådistance apart from each other. Given each of these parameters stated above, the DockQ score can be calculated as

$$DockQ = \frac{(F_{nat} + LRMS_{scaled,d_1} + iRMS_{scaled,d_2})}{3}. \tag{2.1}$$

Here $iRMS_{scaled}$ and $LRMS_{scaled}$ refer to scaled RMS values to avoid large random values. These values are calculated according to a scaling factor d using inverse scaling technique [51].

$$RMS_{scaled,d} = \frac{1}{1 + (\frac{RMS}{d})^2} \tag{2.2}$$

For our implementation of DockQ score we used scaling factors d_1 and d_2 as 1.5 and 8.5 respectively. These values were selected in order to fit the score range of LRMS and iRMS. Based on the DockQ scores, a model structure can be placed in one of four categories as defined in table 2.1.

Table 2.1: Quality-categories based on the DockQ score.

Quality	DockQ score range
Incorrect	$0 \leq DockQ < 0.23$
Acceptable	$0.23 \leq DockQ < 0.49$
Medium	$0.49 \leq DockQ < 0.80$
High	$DockQ \geq 0.80$

Since we are proposing to rank the quality of modeled structures(decoys), We would be using the DockQ scores of the modeled interfaces as labels for training our 3D-CNN model.

Chapter 3

Artificial Neural Networks

An Artificial Neural Network, abbreviated as ANN, refers to architectures inspired by the biological nervous system. Let's begin from the basic building block of a biological neural network called a neuron. A neuron is a special kind of nerve cell that communicates information across the body in chemical and electrical signals and there are several types of neurons serving different purposes from sensory input to motor information. There are two primary components of a neuron called the axon and soma. Neurons connect to form a network where one neuron's axon is attached to another neuron's soma. The information gets transmitted along this chain from one neuron to another. Each neuron is associated with an "action potential," that is, the amount of electric signal needed to excite that neuron. Analogously, an ANN is composed of many interconnected processing elements called neurons [52] which communicate with each other through electrical signals. An artificial neuron gets simulated the same way as a biological neuron by adding together the values of the inputs it receives [52] The activation function is similar to the action potential and determines a neuron's tendency to be triggered. The weight determines the strength of that particular node.

3.1 Feed-forward networks

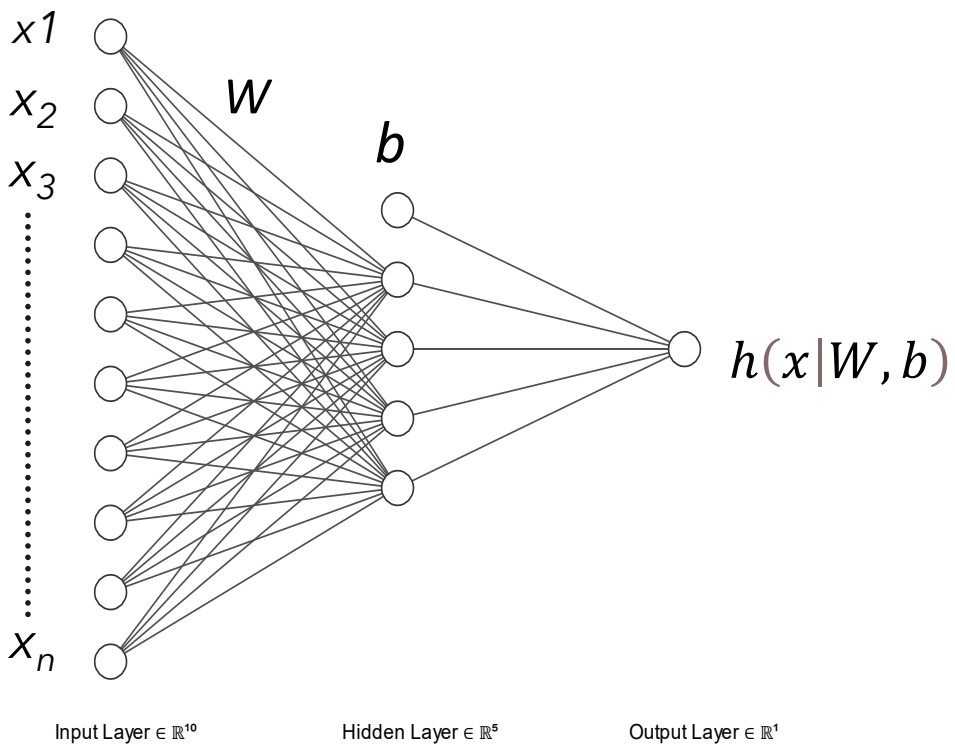


Figure 3.1: Feed-forward network with a single hidden layer. This image is generated using NN-SVG online tool [12]

Figure 3.1 depicts a feed-forward network.Feed-forward networks are neural networks with no cycles in the network, that is the data flows through the network in one direction only. A feed-forward network can be denoted by $f(x \mid \Theta)$ where x is the input vector and θ is the set of parameters. In feed-forward network, f is calculated on inputs of a sequence of layers composed of neurons and produce an output. Mathematically, we can express the output of a layer in a feed-forward network with a weight matrix W and bias b applied to an input vector x as:

$$h\left(x \mid W, b\right) = \sigma\left(Wx + b\right). \tag{3.1}$$

Here σ is the activation function and h is the output from the layer. Details about activation functions are discussed in the next subsection.

The flow of the data in a neural network architecture begins by feeding the data to the input layer and subsequently passing through the hidden layers followed by the final output layer. A feed-forward network is an artificial neural network in which the information propagates in only one (forward) direction. The number of neurons are fixed for the input and the output layers and are equal to the input data dimension and the number of outputs to be generated, respectively. The hidden layers can have any number of neurons, depending upon the complexity of the network we want to have.

3.1.1 Activation functions for neural networks

We use non-linear activation functions for hidden layers and a linear or non-linear activation or no activation on the output layer based on the nature of the problem. As in the classification case, a non-linear function such as softmax is used for the output layer to predict probability values and a linear function for regression problems. Some of the commonly used non-linear activation functions include the ReLU function $\mathrm{ReLU}(x) = \max(0, x)$, the logistic function $\sigma(x) = \frac{1}{1+e^{-x}}$ and the hyperbolic tangent function $\sigma(x) = \tanh(x)$.

3.2 Training of neural networks

Now once the data is propagated through feed-forward network, there is no way to know if the network is learning correctly or not because it lacks a feedback mechanism. The need for a feedback mechanism is achieved through the backpropagation technique [53]. In the backpropagation technique, the learning is associated with a penalty which is calculated using a loss or cost function. Whenever there is a deviation in the predicted and the actual values, the weights of the hidden layer units are adjusted so as to minimize this deviation and optimize the loss in prediction. This weight adjustment helps the internal hidden units to capture the patterns in the task and learn to represent important features corresponding to the task.

3.3 Optimizers

Optimizers are methods that help to optimize different attributes of the model simultaneously while optimizing the loss function. One of the oldest algorithms supporting this technique is Gradient Descent which was first proposed by Augustin-Louis Cauchy [54]. Gradient descent updates the weights by differentiating the loss with respect to all the neural network parameters and taking a step in parameter space opposite to the direction of the gradient. Stochastic Gradient Descent is an improved version of gradient descent, where the weights are adjusted by picking a random subset of samples from the training set, rather than iterating through the complete dataset for adjusting the weights as done in gradient descent [55]. The more robust version of this algorithm is mini-batch gradient descent [56], in which the network is trained on fixed-sized random subsets of data, and the weights are updated for every mini-batch.

AdaGrad [57] and RMSProp [58] are two variants of stochastic gradient descent. Adagrad is usually used in case of sparse data where each parameter is assigned its own learning rate, while RMSProp is a method that normalizes current gradients based on recent gradients. This is achieved by decreasing the momentum (step size) for larger gradients to avoid exploding gradients and reducing momentum for small gradients to avoid vanishing. Adam is another variant of SGD [59]. It uses the feature of AdaGrad to cope with sparse gradients and feature of RMSProp to deal with noisy gradients. It is one of the most widely used and best performing optimizers for deep neural networks.

3.4 Regularization for neural networks

Like any other machine learning algorithms, neural networks are also prone to overfitting [60]. Overfitting is a phenomenon in which the network over-learns the training data and, as a result, loses its capability to generalize to new data. The application of regularization can reduce overfitting in neural networks. Some of the most commonly used regularization methods include dropout [61], early stopping [62], and controlling the momentum and decay rate of the learning rate [63]. Now let us discuss each of these techniques in detail. Dropout randomly deactivates

some of the neurons while training the model. By deactivating random neurons, dropout makes a model more robust by making it learn robust functions of its input features. In this way, dropout improves generalization of the model, and reduces over-fitting. Early stopping is the process of stopping the training process when the validation or test loss stops decreasing. This prevents the network from fitting the training data too well, leading to overfitting.

In this context we also mention the technique of batch-normalization. Batch-normalization helps overcome the problem of exploding and vanishing gradients, usually due to a large range of values in the input features. Batch-normalization normalizes all the activation values across a given batch of input data so that the data is scaled down to a smaller range. Batch Normalization is not a direct regularization technique, but its application produces a regularization effect [64]. In addition to these techniques, data augmentation is one other prominently used technique to improve generalization. The augmentation of data is one of the basic practices in the case of image recognition tasks. The augmentation process includes creating synthetic data by making minor modifications to the original data. [65] For example, in case of image data, data augmentation is performed by randomly rotating and translating the original images.

3.5 Convolutional Neural Networks

Convolution Neural Networks (CNNs), also known as ConvNets, are a class of neural networks that have proven to work well on spatial and image data. The connectivity patterns between neurons in the visual cortex is the motivation behind the concept of a CNN [66]. Different neurons in the visual cortex respond to other stimuli as received by the eye from the visual field. As perceived by the brain, the visual field's final image results from the overlap of the receptive field of different neurons. CNNs overcome overfitting by leveraging the data's structural pattern to learn much-sophisticated features and relations between the features. ConvNets operate by the application of convolution function [67] on the spatial input data.

Like most neural networks, a convolutional neural network architecture comprises an input and output layer with one or more hidden layers. The hidden layers are the convolution layers that

convolve over the input received from the previous layer. In mathematical terms, the convolution operation is a pair-wise sliding dot product. In the case of two-dimensional data, the sliding window has just length and width while the height gets included in the case of three-dimensional data. Each of the convolutional layers is associated with an input dimension, kernel size, and the output dimension.

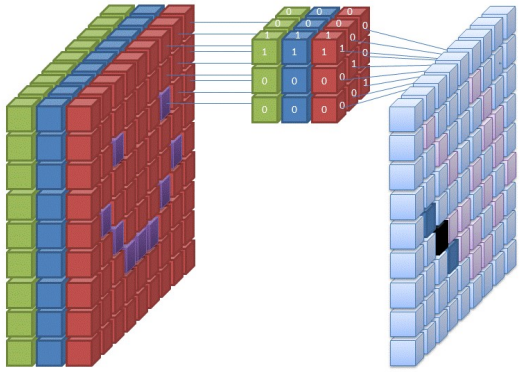


Figure 3.2: Features of a Convolution Neural Network consisting of multiple channels. The convolution operation is applied to a patch of the input data over all the channels for every iteration [13].

Besides these attributes, a convolutional layer has two additional features called the stride and the padding. The stride determines the amount by which the kernel/filter shifts on the input data while convolving. During convolution, if the spatial size of the data is not preserved, the size of the data would reduce rapidly eventually resulting in washing away of data at the borders too quickly. The solution to this problem is using padding at the borders. In padding, empty cells are added at the perimeter of the input data to increase its dimension and thus preserving the size of the spatial size of the data at different layers . Figure 3.3 shows a schematic representation of a convolution filter on two-dimensional data with padding.

For a given dimension d, if the input-size is n,the kernel-size is k, stride is s and padding p, the output-size can be calculated as:

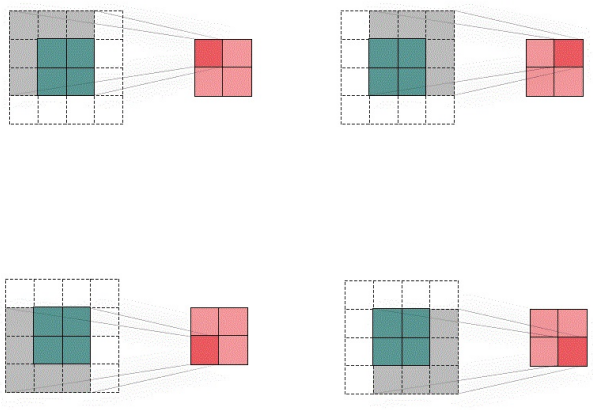


Figure 3.3: A 5×5, two-dimensional data, with a kernel of dimension 3×3, stride 1, and padding of size 1 [14].

$$O_d = \left\lceil \frac{n + 2p - k}{s} + 1 \right\rceil, \tag{3.2}$$

where the $\lceil x \rceil$ is the ceiling function that returns the smallest integer value that is bigger than or equal to a number x.

One of the common problems associated with the convolution output feature maps is that they are sensitive to the location of the significant features in the input [68]. One of the most common ways to deal with this situation is to down-sample the input using pooling layers [69]. Pooling layers help in accumulating features from the maps resulting from the convolution layers. Three pooling techniques commonly used with convolution networks are MaxPooling, AvgPooling, GlobalMaxPooling, and GlobalAveragePooling. Figure 3.5 shows a schematic representation of all the pooling techniques mentioned above. MaxPooling accounts for the maximum value in the pooling window, whereas AvgPooling accounts for the pooling window's average. GlobalMaxPooling and GlobalAveragePooling account for the Global Maximum and Average of all convolving windows, respectively.

Once the data is processed by the convolutional layers, it is prepared for classification using dense or fully connected layer(s).

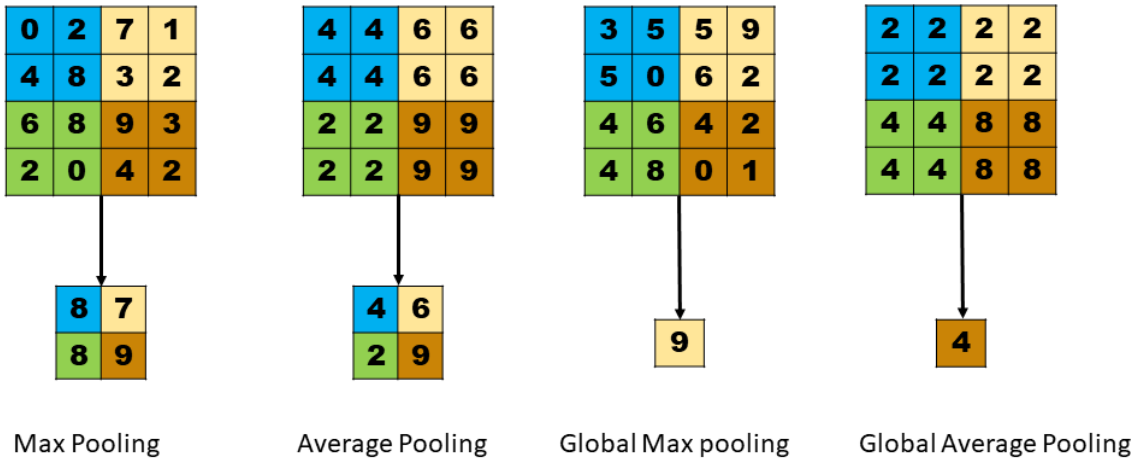


Figure 3.4: Various pooling techniques applied on a 4×4, two-dimensional data, with a pooling filter of dimension 2×2.

3.6 Applications of Convolutional Neural Networks

3.6.1 Image recognition

One of the traditional and notable applications of CNNs is image recognition. LeCun and Bengio first proposed the use of CNN for images [70] at AT&T Bell laboratories in the year 1995. Subsequently, several architectures were proposed aiming towards better and fast performance, such as VGG [71], GoogLeNet [72], ResNet [73]. AlexNet [74] is one of the most accredited architectures that won the ImageNet competition in 2012. An image classification task is associated with identifying the label of the object(s) contained in the image, where objects to be identified by a collection of labels. Deep learning on images needs a lot of data and ImageNet is one such database that contains 14 million images, each hand-annotated with labels corresponding to the objects contained in the image. The labels are then provided as one-hot encoded vectors.

3.6.2 Video processing

The second most widely used application of CNNs is Video analysis [75]. A video is a spatial representation of a series of images, all associated with a temporal dimension. One of the

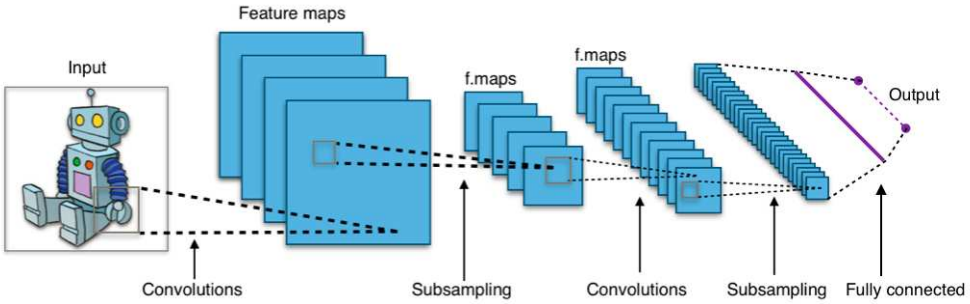


Figure 3.5: A depiction of 2D convolution on the images. The features extracted from the original image are down-sampled by applying pooling layers of different sizes. [15]

most prominent uses of CNNs is video classification and identifying objects in the videos just as from images. By learning from the raw pixel position and information from the videos, CNNs tend to identify objects from the videos. This task is useful for object identification and provides substantial information to consider a video of a specific category.

3.6.3 Natural Language Processing (NLP)

NLP is a sub-branch of linguistics and artificial intelligence that deals with the interpretability of human speech and text processing capabilities by a machine. By looking at the pattern of characters or words in a section of text or speech, a machine can generalize on these patterns and decipher deep contexts from the input. Just like an image, a part of text or speech could be represented in a matrix format. The sentence's words comprise the rows, and the individual characters of these words include the column. Such representation is well suited for application of 1D-convolutions [76]. CNN's have proven to be successful in NLP tasks such as sentiment analysis and dialogue translation from one language to another. In the last few years, a substantial amount of work has been done to enhance the NLP tasks. One such prominent technique is the Convolution augmented Attention model that promotes larger weights to the locations that tend to bear much important information required for performing the task.

Chapter 4

Convolutional Neural Networks for Protein Structures

In this section we discuss prior work on 3D CNNs for modeling protein structures. The driving motivation behind all these methods is to overcome the need to prepare hand engineered features as 3D-CNNs can learn patterns on their own from the raw data. 3D-CNN have been previously used for quality assessment of protein structures and assessment of protein-protein docked interfaces. Quality assessment of protein structures refers to the problem of predicting the quality of protein structural models to determine how close they are to the corresponding native structure. All the methods discussed in this section use a 3D-voxel based feature representation and differ in various ways described below.

Derevyanko, Grudinin, Bengio, and Lamoureux proposed an approach for quality assessment of tertiary structures of proteins [1]. The authors propose to represent 3D protein structures in terms of atomic densities that are used to construct 11 channels (details below). For any atom in the protein complex, its atomic density is stored in a voxel of dimension 1Å. Here the voxel position is determined by the atom's actual 3D coordinates in the 3D strcuture of the protein. Each grid has a dimension of $120 \times 120 \times 120$ cells. The model tries to rank the decoy structures based on the GDT_TS score [77], which is a measure of the similarity between two protein structures with same amino-acid sequences but a different tertiary structure. Equation 4.1 provides the mathematical definition of the atomic-density calculation. For a given coordinate location of an atom in 3D space, its atomic density $\rho(r)$ is computed over a spherical region of 2Å as a function of the distance r:

$$\rho(r) = \begin{cases} e^{-r^2/2}, & \text{if } r \leq 2\text{Å} \\ 0, & \text{otherwise.} \end{cases} \tag{4.1}$$

Sato and Ishida presented a protein structure assessment method based on local structure quality [16]. Their method first evaluates the local structure's quality for each residue and produces an equivalent local score. These local scores are later summed up to produce a single score corresponding to the predicted structure's overall score. The occupancy of the atoms comprising the protein are used to build the 3D-voxel input for the 3D-CNN.

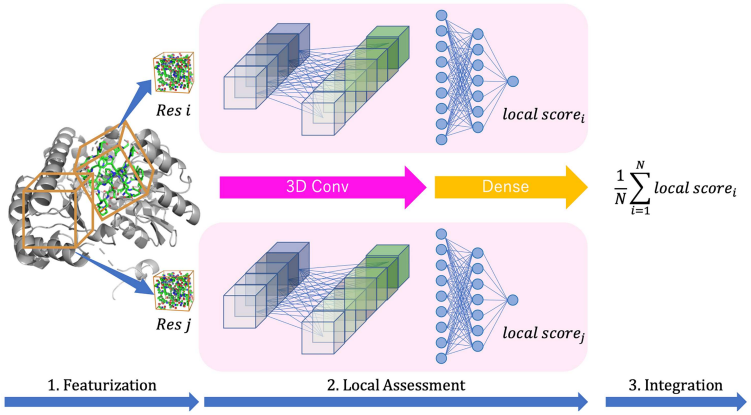


Figure 4.1: Residue level feature extraction and global score calculation for the model structure as proposed by Sato and Ishida [16].

A similar approach was adopted by Kihara, Wang, Christoffer, Terashi, and Zhu in developing a voxel-based deep neural network for protein-docking evaluation called DOVE [78]. DOVE is designed to differentiate between near-native decoys and incorrect decoys. The input data includes atomic positions and atom-wise statistical potentials in a cubic region of 20Å or 40Å aligned to the interface center. An interface is defined by atoms from the two interacting proteins that are at most 10Å from each other. The atomic level information is projected to four different channels corresponding to the atom type of the atom any given spatial location. The first three channels correspond to the Carbon, Nitrogen, Oxygen respectively while all the remaining atom types are included in the fourth channel. The number of atoms of the corresponding type is stored in a 1Å voxel and a 2Å voxel for a 20Å and a 40Å cube, respectively. In addition to these positional features, two contact potential features called ITScore [79] and GOAP [80] are used as input features. Figure 4.2 depicts the 3D-CNN architecture of the DOVE model.

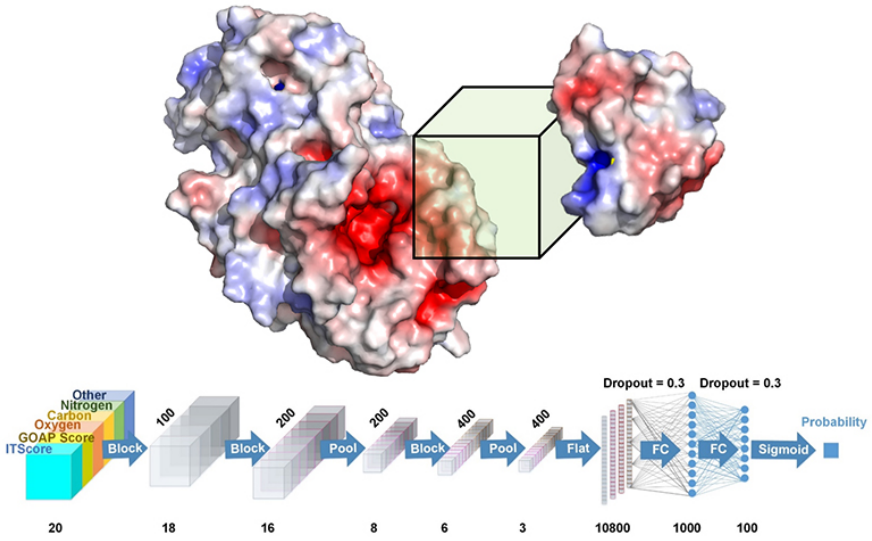


Figure 4.2: 3D CNN model architecture for DOVE. *Picture used by permission*

DeepInterface is another method which predicts how close an interface is to the true interface [17]. The interface is represented as 3D grids with 40 channels. Each of these channels accounts for the 19 residue types for each of the participating proteins and one channel for each of the protein's alpha alpha-carbon. A 3D Gaussian distribution of these amino acids averaged over C-alpha and C-beta coordinates are used as details for the 3D-voxel grids. Figure 4.3 gives a schematic representation of the 3D-CNN model for DeepInterface.

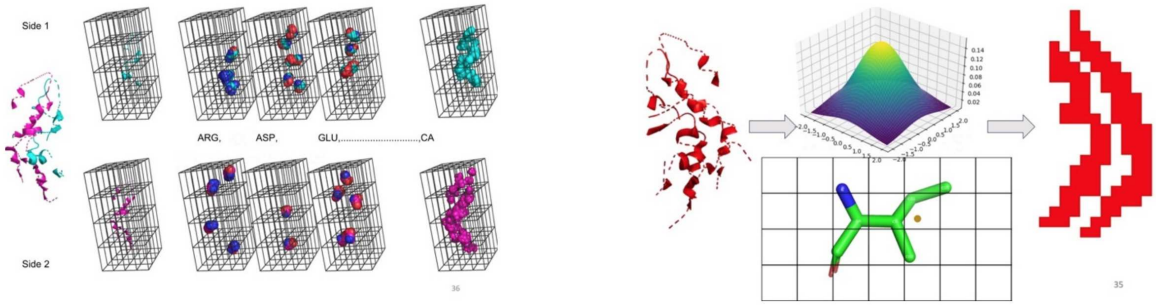


Figure 4.3: The 3D CNN model architecture for DeepInterface [17]. *Picture used by permission*

Altman and Torng proposed a similar way of representing the protein-protein interface for protein functional site detection using 3D convolution. The method consists of a 3D CNN to

predict the probability of a residue to be a part of the binding site. It uses four different channels representing Sulphur, Carbon, Oxygen, and Nitrogen atoms. The occupancy of these atoms is projected to the corresponding channel. Finally, a Gaussian filter is applied to normalize the counts across the four channels. For every residue in the protein, the model convolves over the residue's local environment and outputs a probability of its likelihood in binding-site formation.

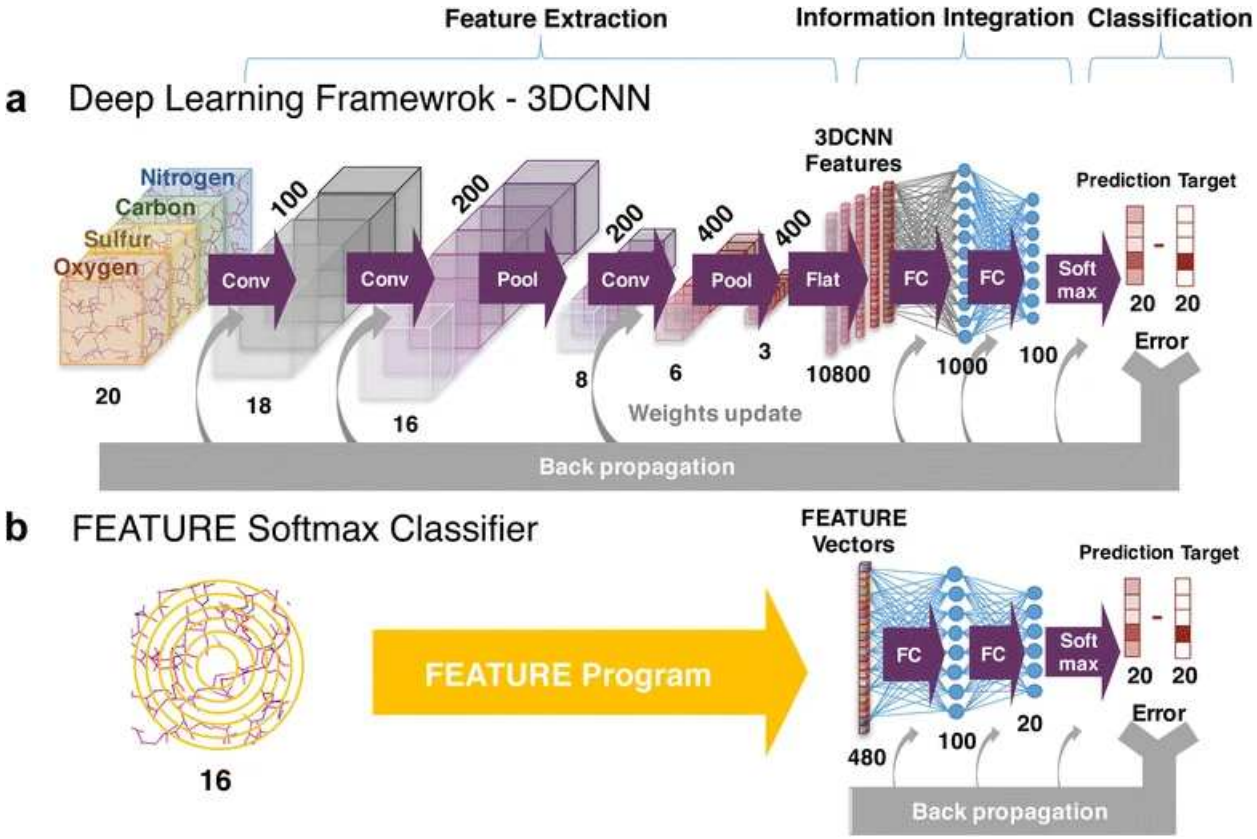


Figure 4.4: Deep 3D Convolutional Neural Network and FEATURE-Softmax Classifier models [18]. *Picture used by permission*

Table 4.1 summarizes the various aspects associated with each of the methods discussed in this section.

Table 4.1: Table summarizing different methods on protein structure analysis

Summary of the 3D-CNN based methods for assessment of proteins structures and protein-protein interfaces				
Method	Problem	Features	Loss function	Model architecture
Georgy Derevyanko, Sergei Grudinin, Yoshua Bengio, Guillaume Lamoureux	Quality assessment of protein folds	11 density maps for 11 different atom types for the entire protein structure	Pair-wise ranking loss	Input Dimension - 120 x 120 x 120 3D convolution layers - 12 Fully connected layers - 3
Rin Sato, Takashi Ishida	Protein model accuracy estimation based on local structure quality assessment	For each residue , 14 channels with - 11 for different atom types and 3 for C-alpha, backbone chain and occupancy	cross entropy at local level	Input Dimension - 28 x 28 x 28 3D convolution layers - 6 Fully connected layers - 3
DOVE	Protein docking model evaluation	6 channels, 4 for 4 atoms types and 2 for ITSCORE and GOAP score	cross entropy	Input Dimension - 20 x 20 x 20 3D convolution layers - 5 Fully connected layers - 3
DeepInterface	Protein-Protein interface validation	40 channels - For each protein 19 channels for 19 residue types 1 channel for C-alpha atom	cross entropy	Input Dimension - 50 x 50 x 50 3D convolution layers - 4 Fully connected layers - 3
Wen Torng, Russ B. Altman	Amino acid environment similarity analysis	4 channels for 4 atom types (C, N, O and S). Occupancy of each atom is added to the corresponding 3D grid.	cross entropy	Input Dimension - 20 x 20 x 20 3D convolution layers - 5 Fully connected layers - 3

Chapter 5

The proposed method for protein-protein docking assessment

In this thesis, we propose a 3D-CNN ranking method that is applied to the interface of two interacting proteins and learns to rank the quality of these docked configurations as measured by the DockQ score [81]. The proposed method applies to protein complexes consisting of two interacting proteins (dimers). The following three sections discuss the extraction of the atoms participating in the interface, representing the interface as a 3D grid and splitting them into multiple channels. This chapter also gives a detailed overview of the 3D-CNN architecture followed by the loss functions that have been used. Finally, this section discusses the stratified sampling method we used to deal with the imbalance in our dataset.

5.1 Docked decoy representation

We are interested in ranking protein-protein docking decoys. As a part of the solution, we focus on the interface region of the two interacting proteins. To represent the interface we choose all atoms belonging to residues which are within 10Å of any residue of the other protein (see Figure 5.1).

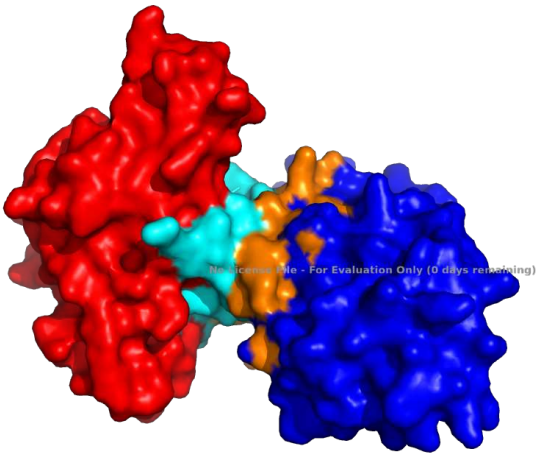


Figure 5.1: A cartoon example of a protein-protein interface for the complex 1E96 generated using the PyMol software [8]. The blue and red colored portion represents the two participating proteins. The orange and cyan region represents the residues chosen to represent the interface.

Commonly used sources of protein data such as the Protein Data Bank [82] and the Docking Benchmark Dataset [83], provide the 3D structure of protein complexes. This makes it straightforward to map these structures to a 3D grid based representation and feed them to 3D convolutional networks.

We chose to rotate all the protein-protein interfaces parallel to the XY axis for all the experiments conducted. To do this, we first compute the plane that separates the protein-protein interface and then rotate all the interface atoms along this plane while keeping the interface aligned to the XY plane. This would not only bring consistent interface representation but would also discard the need for data augmentation with random rotations and translation of the interface. To achieve this, we used a 3D-SVM from scikit-learn [84] to find the plane that separates the interface between two chains in 3D space Fig 5.2 depicts a 3D grid representation of atoms after rotating the interface.

As mentioned in the previous chapter, several approaches have been proposed to represent protein structures. As a part of this thesis, we use Derevyanko, Grudinin, Bengio, and Lamoureux's idea to store the atomic density in a voxel of each 3D grid corresponding to each atom type of interest (refer to chapter 4). However, we use four channels for four atom types like Kihara,

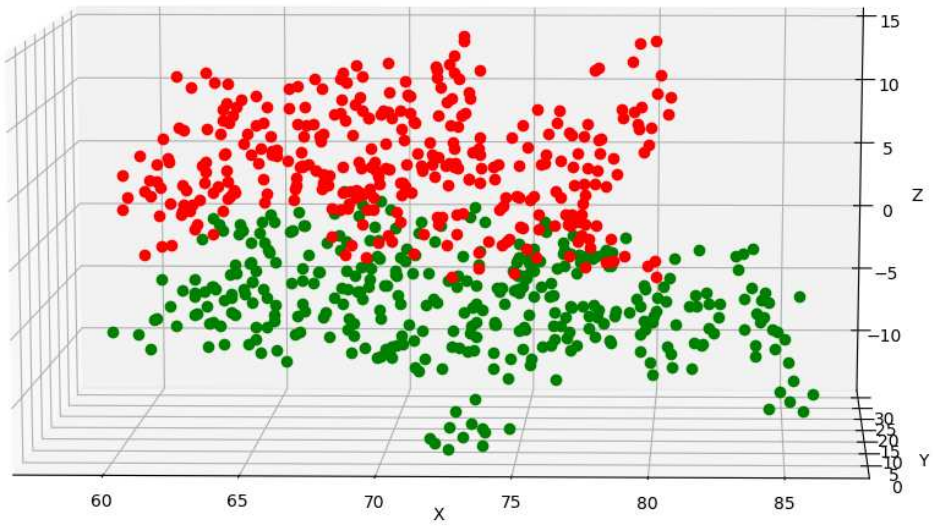


Figure 5.2: A 3D grid representation of atoms at the interface in the protein-protein complex 1S1Q from DBD4. The green dots represent atoms from the receptor and the red ones represent the atoms from the ligand. This image is rendered using python's matplotlib library [19].

Wang, Christoffer, Terashi, and Zhu [78] to represent the protein interface. We also experimented with using eleven channels for eleven different atom types. Here, the atom types differ from each other in terms of either element type or molecular characteristics. For example, we considered 3 different channels corresponding to three different hybridization states of carbon. A more detailed categorization of these atom types is provided in appendix A.

5.2 3D convolution on protein-protein docked interfaces

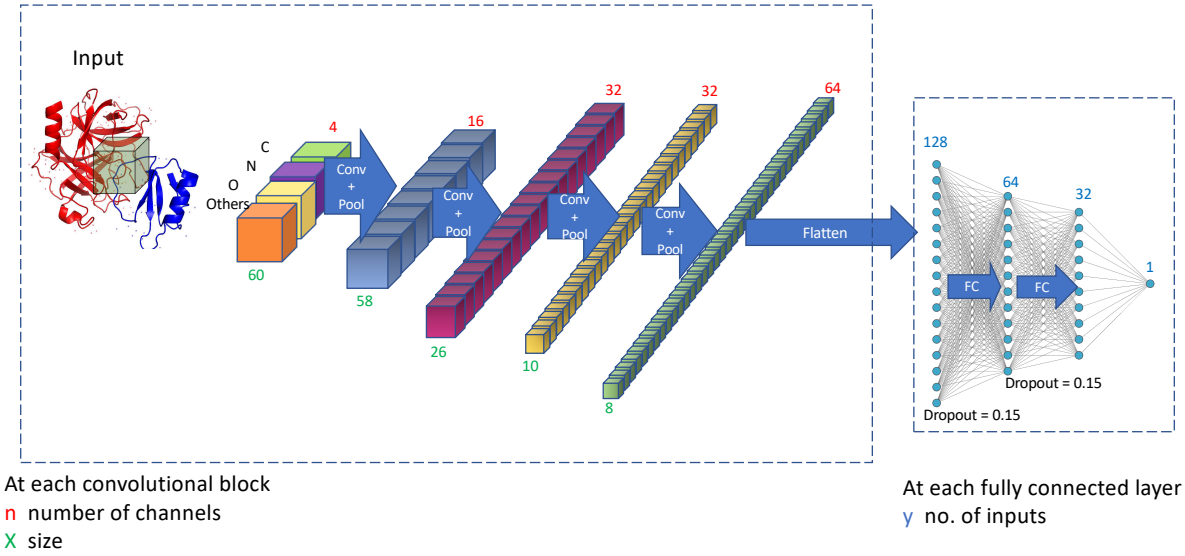


Figure 5.3: The architecture of the proposed 3D CNN. A part of this image is rendered using NN-SVG software [12]

Our model consists of five convolutional blocks and three dense layers (see Figure 5.3 for a schematic representation). As we progress from the first convolutional block to the last convolutional block, we increase the number of filters from 16 to 128. Increasing the number of filters not only improves the model's sensitivity to the local spatial information, but also allows the representation of abstractions that capture higher order features. The final convolutional block is connected to a block consisting of three fully connected layers. All layers have ReLU activation between them except for the last layer. The final outcome of the fully connected layer is a score predicted by the network. The total trainable parameters for this network are 330,321.

5.3 Loss functions

Docking decoy assessment can be formulated in two ways. One way is to treat it as a regression problem by predicting a score corresponding to the interface quality. Secondly, it can be treated as a ranking problem since we want to rank the decoy structures according to their quality. We

consider the DockQ [81] score as a label to measure the quality of the docked interface of the decoy model.

As a part of this thesis, we consider three different loss functions. The first is the Squared Error, which is typically applied for solving regression problems. Secondly, Absolute Error which is also a commonly used loss function for regression problems. Finally, we consider a ranking loss method called the pair-wise ranking loss. The subsequent sections discuss each of them in detail and their application to protein-protein interface quality assessment. For all the loss function, we used Adam optimizer [59].

5.3.1 Squared Error

Regression analysis in machine learning refers to a learning method to predict a dependent (target) variable as a function of independent variables (features). The traditionally used error or loss function is the squared error $(y_i - \hat{y}_i)^2$, where y_i is the target value and $\hat{y}_i$ is the value predicted by the network.

subsectionAbsolute Error

Absolute error is another commonly used loss function for regression problems. Absolute Error, given as $|y_i - \hat{y}_i|$ is the absolute difference between the actual and the predicted values. Here, in the equation, y_i is the target value and $\hat{y}_i$ is the value predicted by the network.

5.3.2 Pairwise Ranking Loss

Tie-Yan Liu first proposed several approaches for ranking loss in the paper "Learning to Rank" [85]. The training data consists of ordered data samples. The pairwise ranking loss considers a pair of items at a given time from a list of items that need to be ranked. The goal here is to organize the paired items so that the produced ranking is optimal and as close as to the ground-truth order keeping the number of rearrangements as small as possible [86].

We use the same pairwise-ranking loss used by Derevyanko, Grudinin, Bengio and Lamoureux [1] for quality assesment of protein-folds, based on the work on "Learning to Rank". The pairwise-ranking loss is defined in terms of the ranking margin. For each pair of decoys, let $DockQ_i$

denote the DockQ of decoy i, and $DockQ_j$ denote the DockQ of decoy j. Let y_{ij} be the ordering coefficient of decoys i and j, equal to 1 if $DockQ_i \leq DockQ_j$ and to –1 otherwise. Then, the pairwise loss function can be expressed as:

$$L_{ij} = w_{ij} * \max\left[0, 1 - y_{ij} \cdot (s_i - s_j)\right].$$

Here, s_i is the score predicted by the network for decoy i, and s_j is the score predicted for decoy j. The coefficient w_{ij} is expressed in such a way that decoys with similar DockQ score are eliminated from the training process. In other words, for a pair of decoys i and j, w_{ij} would be 0 if the absolute difference between the DockQ scores of these decoys is less than the defined threshold τ and equal to 1 otherwise. For our experiments, we tested with threshold values of 0.2, 0.3. Finally, the loss is averaged for all targets selected over a mini-batch.

5.4 Stratified sampling

Most of the decoys in our dataset had a low DockQ score (refer fig:5.4). Thus we needed a sampling method during the training process to preferentially pick a target with a greater number of good quality decoys. Imbalanced datasets are one of the common issue in classification problems. With such datasets, the model tends to focus on the categories that have many samples, which can lead to a model that would put every unseen sample into the category with largest number of samples. To overcome this issue, we use the concept of Weighted Random Sampling, where each sample in the training set is assigned a weight based on the number of samples its category has in the training set.

We categorized the interfaces into three main categories according to their DockQ score:

$$\begin{aligned} \textbf{Low} \quad & 0 \leq DockQ \leq 0.23 \\ \textbf{Acceptable} \quad & 0.23 \leq DockQ \leq 0.49 \\ \textbf{Medium-High} \quad & 0.49 \leq DockQ \leq 1. \end{aligned} \tag{5.1}$$

We have added two levels of stratification to our sampling process: At the first level we do a probability-based picking of targets, and at the second level we randomly select a docking software available for that chosen target. For performing the first level of selection, we assign probabilities to all the targets in the training set. This probability is assigned on the basis of the number of medium-high quality decoys a target has from all the docking softwares. For the second level of stratification, we choose a random software to avoid a bias in selecting the docking software with a larger number of good quality decoys.

5.5 DockQ category based binning

As mentioned in the previous sections, our dataset dominantly consisted of low quality decoys, so we wanted our sampler to include docking solutions from all three DockQ categories to improve generalization and avoid bias. Fig 5.4 shows decoy distribution in the dataset over three DockQ categories we chosed for binning.

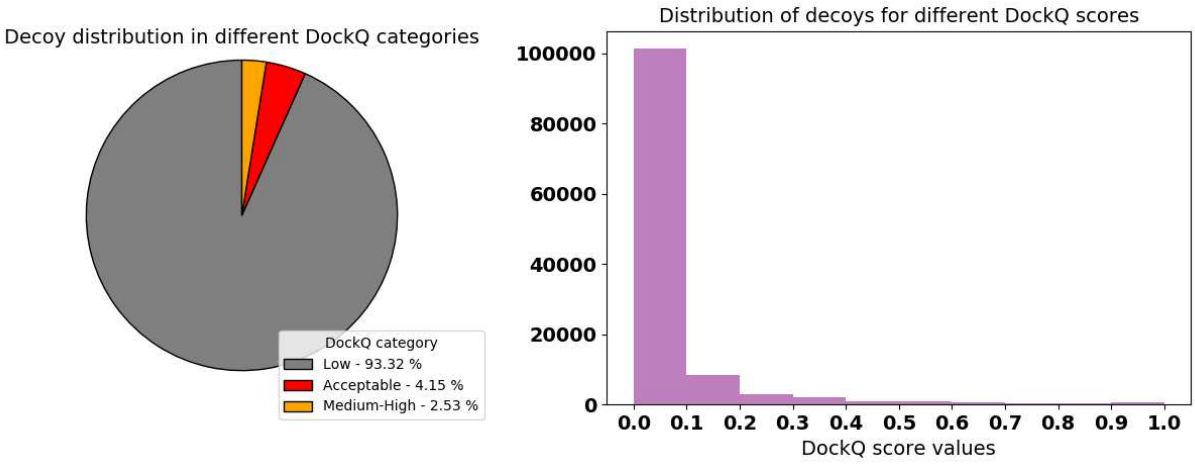


Figure 5.4: Distribution of decoys from the dataset over three different DockQ categories

Once a target and a random docking software are selected, we bin decoys from each category. The idea behind the binning strategy here is to pick equal amount of decoys from each of the DockQ categories for a given batch size. The highest preference is assigned to medium-high

category followed by acceptable and low categories. If there are insufficient decoys available for any selected target, we substitute them by binning from the next preferable category.

5.6 Best model selection

The model was trained on a CentOS linux machine with 12GB GPU memory and 16GB CPU memory. The machine was enabled with a NVIDIA Corporation GV100 TITAN V GPU. It took around 16 hours to train the model on a single GPU for 100 iterations. For every iteration, we used 200 mini-batches of size 6 and evaluated the model on the entire validation set for that iteration. Finally, we choose the best weights for the iteration at which the model identified highest number of natives and first near-natives for validation set targets.

As discussed in this chapter, we have experimented with various methods of interface representation and various loss functions. In order to select the best configuration amongst the experimented combinations, we evaluated each of these models on the validation set (discussed in chapter 6, section 6.1.2). The model that could identify the largest number of natives and first near-natives for the targets from the validation set was considered as the best model.

Chapter 6

Experimental setup

All the experiments described in chapter 5 were evaluated and validated against state-of-the-art docking software. All the data was prepared using four docking softwares namely ZDOCK, HADDOCK, FRODOCK, and ClusPro. Each of the docking software is described in chapter 2 section 2.3.4.

6.1 Data

The dataset comes from three sources: Docking Benchmark Data V5 (DBD5) [87] , ProPairs [88], and Dockground [89]. The sections below provide a brief overview of each of the datasets used and the techniques used to filter them.

6.1.1 Data sources

Docking Benchmark Data version 5.0

Docking Benchmark Data version 5.0 is a corpus of 3D structures of protein-protein complexes selected from the Protein Data Bank (PDB) [82] in both bound and unbound conformation. DBD5 is created so that none of the partnering proteins in a complex belong to the same SCOP (Structural Classification of Proteins) family [90]. DBD5 targets are divided into three categories: rigid body (easy), medium difficulty, and difficult, based on the extent of the conformational changes they undergo while forming the bound conformation.

ProPairs

ProPairs is a dataset of non-redundant protein-protein docking complexes obtained from PDB. ProPairs is based on an algorithm that extracts valid protein docking complex structures and the unbound structures of the binding partners from PDB. For any bound complex, if either of the partners' unbound structure is absent, it is replaced by the unbound structure of another protein which has a similar structure to the same partner in the bound configuration. This resulted in

5,642 protein complexes. Finally, after filtering to remove redundancy at a level of 40% sequence identity, 2070 out of the 5642 complexes remained.

Dockground

Dockground is another dataset that serves as a source for testing and validating different protein-protein docking techniques. For every protein-protein complex in the dataset, Dockground provides a bound configuration, the co-crystallized structure of the complex, and an unbound X-ray structure for each of the partnering proteins. It also provides simulated docking benchmark sets, model–model docking benchmark sets, scoring benchmarks (docking decoys), and templates for comparative docking.

6.1.2 Data cleaning

DBD5 includes 23 new targets along with 176 targets obtained from DBD4 [83]. We used the non-redundant dataset for ProPairs, which consisted of 2070 multimer protein complexes. The Dockground dataset consists of 99 non-redundant complexes. From all the datasets, we kept the dimers (dimers are protein complexes with just two interacting proteins). This resulted in 199 targets from DBD5, 1053 targets from PropPairs, and 99 from Dockground.

To remove redundancy and split the data into training, validation, and the test set, we used the CD-HIT [91] to distribute the dataset into multiple clusters based on the threshold specified. Since we are more interested in heteromer dimers, we discarded those complexes where both the participating proteins had similar sequence. In order to do this. We used BioPython's SeqIO module [92] to get the sequence of each of the protein and then applying a 90% similarity threshold on the sequences of both the proteins. At the same time, we also discarded the complexes where either of the ligand or receptor had less than 40 atoms. After imposing some constraints on the sequences within each of the targets, then performed a Protein Sequence Identity match with a 95% threshold between all the targets in the entire dataset to remove redundancy between multiple datasets. To split this data into training, validation, and test dataset, we again performed a Protein Sequence Identity match with a 25% against the 23 new targets from DBD5. Initially, we started with a test

set consisting of 23 targets from the DBD5. Later, if any target from the datasets mentioned above has S.I greater than 25% against the test set, they are added to the test set otherwise kept in the training set. Similarly, we moved some targets from the training set to the validation set using an S.I threshold of 40%. If any two targets in the training set have S.I greater than 40%, they will only be retained in the training set; otherwise, they will move to the validation set. If a complex is present in multiple datasets, preference has always been given to DBD5 followed by DockGround and ProPairs while choosing the bound and unbound structures of the target.

After all the redundancy check, the final dataset consisted of 390 targets in total with 131 from ProPairs, 60 from the Dockground and 199 from DBD5.

Table 6.1 summarizes the distribution of targets over multiple datasets before and after removal of redundant targets.

Table 6.1: Table summarizing the initial and final count of targets from each of the dataset before and after performing the redundancy check over the dataset

Dataset summary - from multiple data sources			
	DBD5	Propairs	Dockground
Initial	199	1053	99
Final	199	131	60

The targets identified previously were distributed into training, validation and test sets. Initially we had 50 targets in the validation set, but we found that only 20 of them had at least one near native decoy. Thus, this led us to move those remaining 30 targets into the training set. The final distribution of the training, validation and the test set are summarized in table 6.2.

Table 6.2: Table summarizing the distribution of targets over the training, validation and the test

Summary of the final dataset		
Training	Validation	Test
390	20	50

6.1.3 Independent dataset for model evaluation

To test our model performance, we evaluated our model on the combined CAPRI-ScoreSet and MOAL dataset provided by the publishers of ProqDock [93]. The provided dataset had 19013 models for 15 targets from CAPRI and 56015 models for 118 targets from MOAL. All the targets for CAPRI and MOAL were selected from Docking Benchmark Data version 4. To obtain a dataset with low similarity to our training set, we performed a homology sequence comparison of targets to avoid any similarity or redundancy with respect to the targets in the training and validation sets. We choe targets from CAPRI and MOAL that were less than 25% similar to our training and validation set. We further filtered these targets by selecting only the ones that had at least one good quality decoy. With this, we finally ended up with 6 targets from CAPRI and 29 from the MOAL.

Table 6.3 summarizes number of targets from the CAPRI and MOAL datasets.

Table 6.3: Table summarizing the number of targets from the CAPRI and MOAL datasets at each level of refinement.

The CAPRI and Moal dataset refinement summary		
	CAPRI	MOAL
Initial	8	123
Non-redundant	6	81
At least 1 good-quality decoy	6	29

6.2 Docked decoys dataset

Table 6.4: This table summarized number of targets for which each of the docking software produced docking solutions.

Dataset summary - targets for each docking softwares.				
	ZDOCK	HADDOCK	FRODOCK	ClusPro
targets/docking software	32	38	421	449

As a part of the experiment, we considered docking solutions from all of the docking softwares mentioned above. Table 6.5 summarized the average number of decoys produced per target from the corresponding docking software.For ZDOCK, we initially had 50,000 decoys per target, but the composition consisted of a large number of low-quality decoys, so we ended up selecting the top 1000 for each of the targets.

Table 6.5: Table summarizing the average number of decoys per target produced from each of the docking software.

Number of decoys per target for each software				
	ZDOCK	HADDOCK	FRODOCK	ClusPro
decoys/target	1000	500	100	100

The ZDOCK [83] and the HADDOCK decoys [94] were obtained from a pre-computed set of decoys for targets from DBD4 and DBD5 respectively. While for the FRODOCK and ClusPro decoys, we used the software on ubound receptors and ligands of targets from ProPairs, Dockground and DBD5.

6.3 Data augmentation

Data augmentation is a common practice in many deep learning applications [65]. Data augmentation is a method to expand a dataset by adding synthetically generated data or modifying the existing data. Since deep learning models tend to learn and generalize well on larger datasets, data augmentation proved to be one of the best techniques to improve model performance, especially for sparse or imbalanced datasets. As mentioned earlier, our dataset has a high percentage of low-quality decoys compared to the high-quality ones. As part of our experiments, we augmented our dataset by randomly rotating the protein-protein interface about the center of the interface. More information about calculating the angle of rotation is provided in the appendix A.

Chapter 7

Results and conclusion

This chapter explores the performance of the 3D-CNN model in ranking the quality of protein-protein interfaces. In this chapter, we frequently use the terms **native interface** and **near-native** interface. Here, a native interface refers to the target's true/crystal structure interface. In contrast, a near-native interface refers to a decoy interface whose structure closely resembles the target's true structure. This closeness is measured in terms of the decoy's DockQ score. Since our dataset consists of decoys from various docking softwares, a single target can have many near-native decoys generated from different docking softwares. To maintain consistency and achieve better comparison, for any target we choose a near-native decoy from a docking software that produces a good quality decoy in terms of the DockQ score. The goodness of a method is defined as its ability to identify a native or a near-native interface in top n ranks. We chose five different values of n as 1, 5, 10, 20, and 50.

In this section we compare the performance of models trained using various loss functions (discussed in section 5.3) on the test data. Then we evaluate the performance of the best model on the our test data and the independent test data (CAPRI and MOAL). Here we inspect number of native and near-native interfaces identified by the model in top n ranks. Following this, we compare the best model's performance to each of the docking software used to create the dataset except for ClusPro. Finally we compare our model's performance against the DOVE model (referred to in the chapter 4) on the test set. Each of the comparisons mentioned above is made at three different DockQ thresholds, as discussed in section 5.5.

7.1 Performance of different 3D-CNN models

This section compares the performance of several models which we have tried and tested as a part of this thesis. As a part of these experiments we have tested different loss functions. We also experimented with different regression models. For a single variate regression, we used just the DockQ as the independent label while for multi-regression, we used all the components of DockQ (F_{nat}, F_{nonnat}, $iRMS$, and $LRMS$) along with the DockQ score itself as multiple independent variables. In addition to this we experimented with using 11 different channels to represent the interface atom features instead of four, along with data augmentation. The model that uses 11 channels was not competitive with the model that used four channels; those results are in appendix A.

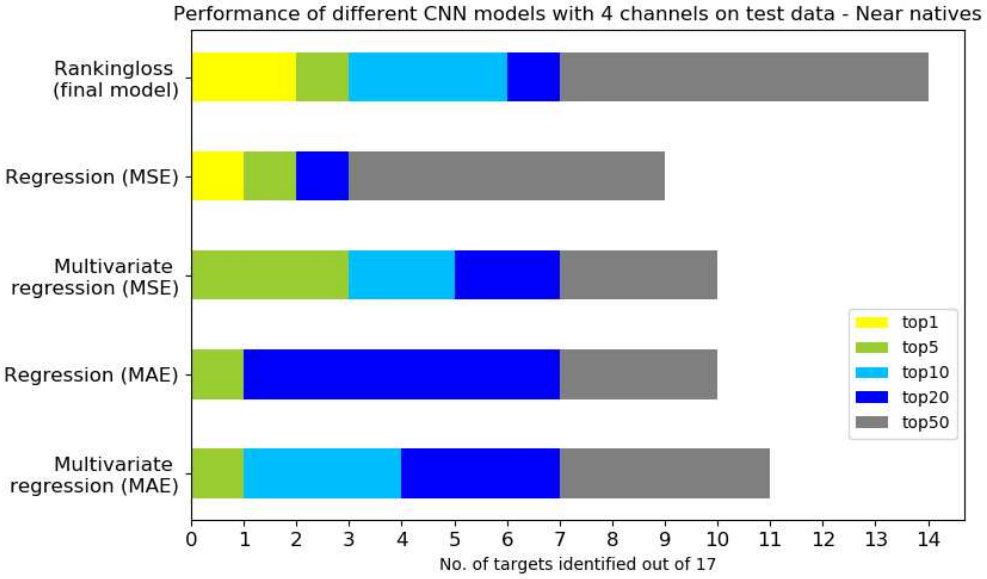


Figure 7.1: Performance of 3D-CNN models on the test data. Each section in the barplot gives the number of targets for which each of the models could identify its near-native in the top *n* ranks.

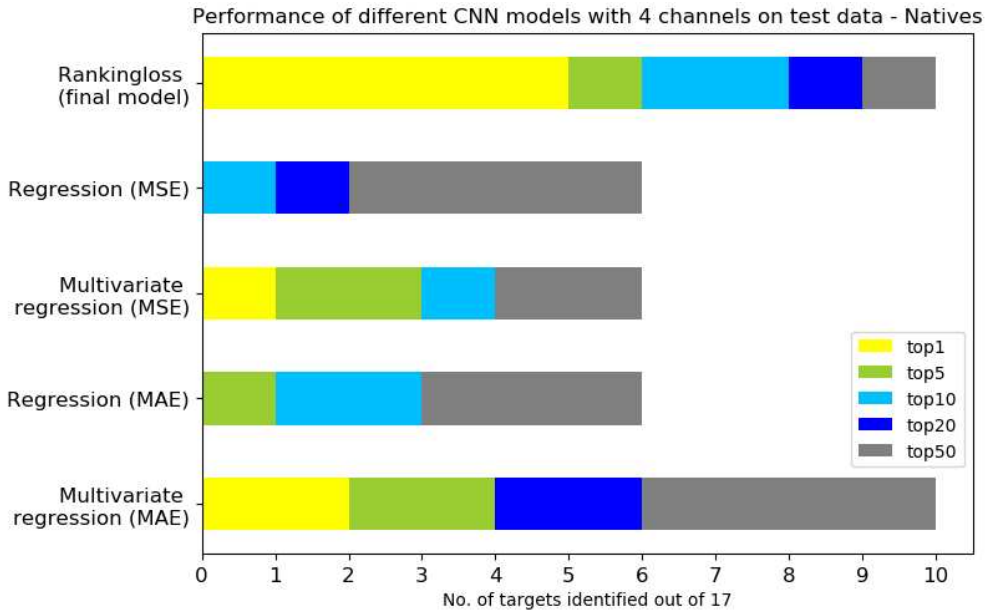


Figure 7.2: Performance of 3D-CNN models on the test data. Each section in the barplot gives the number of targets for which each of the models could identify its native in the top n ranks.

Fig 7.1 shows performance of several 3D CNN models. While looking at the bar plot for near-native structures, we can see that the Ranking loss gave the best performance overall, being the only model to identify some near-natives in top 1 rank. For regression loss, the model performs much better with a Squared Error loss than Absolute Error loss function. We can also see that there is a slight performance improvement using the multi-variate regression model for both the Absolute Error and Squared Error loss functions.

Fig 7.2 shows performance of several 3D CNN models for native structures. Again, we can see that the ranking loss gave the best performance when compared to the two forms of regression loss. For regression models, we can see that with Absolute Error loss, model is performing well than with Squared Error loss in case of single-variate regression as well as multi-variate regression. The performance improvement from single-variable regression to multi-variable regression is quite similar to the case of near-natives. This improvement is considerably more than to the case of near-natives.

7.2 Performance on the test, CAPRI and the MOAL data

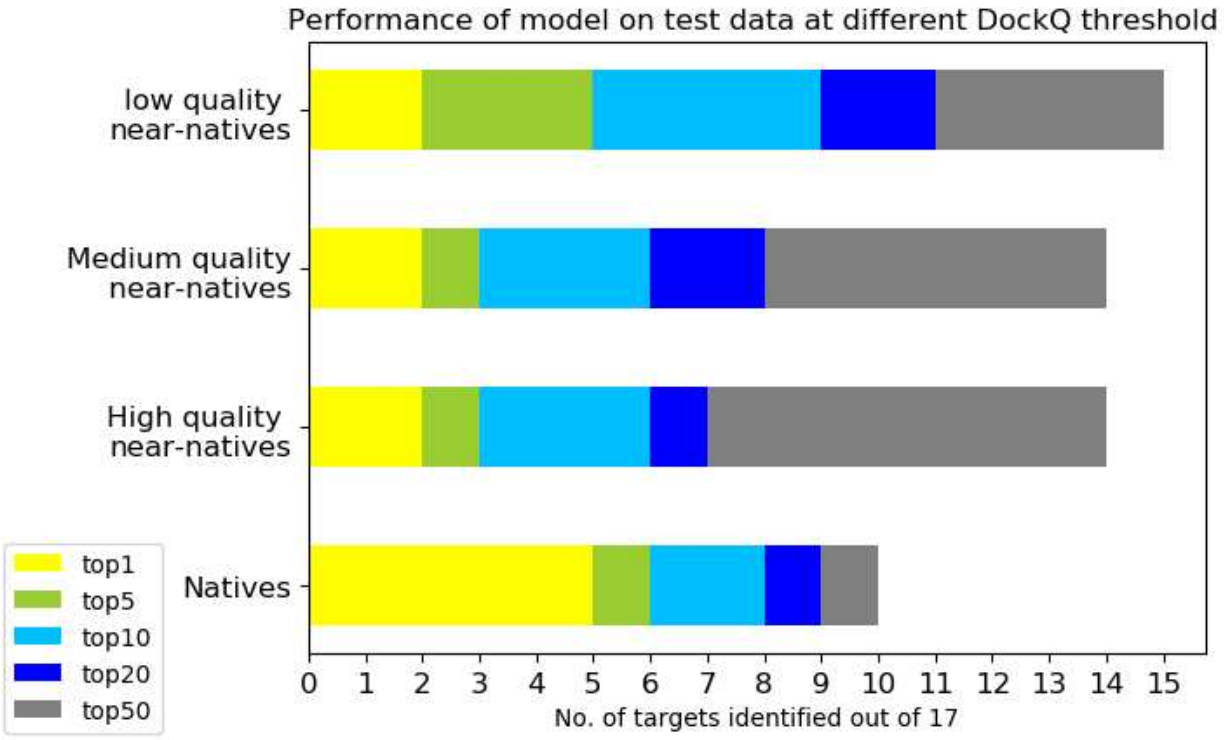


Figure 7.3: Model performance on the test data. Here we compare number of natives and different quality of near natives identified by the model. Each section in the barplot defines the number of targets for which the model could identify its native/near-native in top *n* ranks.

Fig 7.3 gives performance of the 3D-CNN model on the test data.The test data had 50 targets initially and out of these 50 targets, only 17 of them had at least one high quality decoy. So we used only these 17 targets to report model's performance on the test set. We can see that as the quality of the selected near-native decoy increases, the number of targets in top *n* ranks decreases. Also, we can see that the model is able to identify more native structures at higher ranks than at lower ranks. This is because, a native structure would have a higher rank than a near-native structure. However, there are more near-natives at higher ranks (at rank 20 and rank 50) and these numbers increase as we relax the DockQ threshold criteria from 0.65 to 0.23. This trend can be justified by the fact that a low quality decoy is less likely to be identified in top *n* ranks for a higher evaluation

criteria. With this, we can say that the model is able to distinguish well between low and high quality decoys.

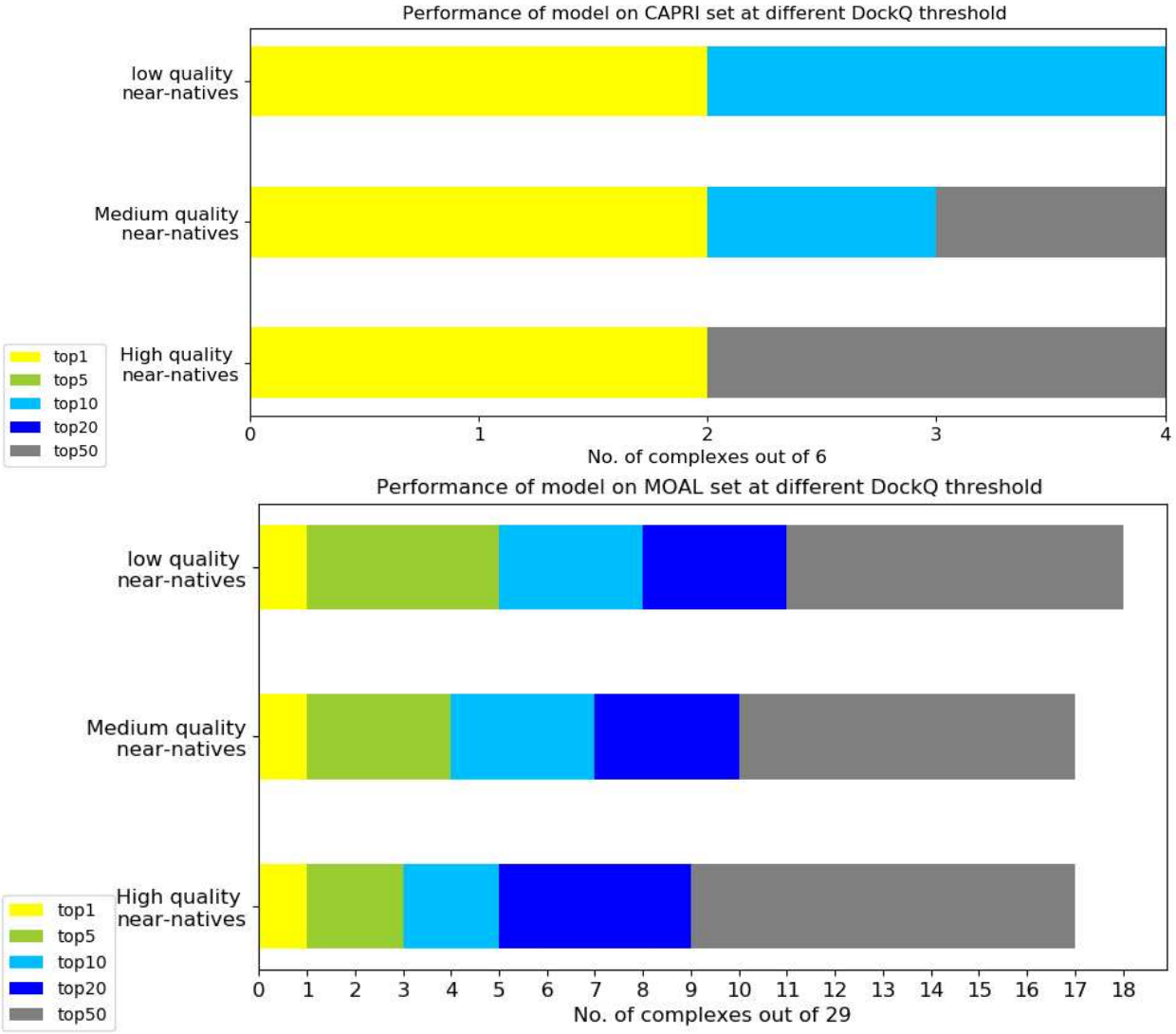


Figure 7.4: Model's performance on CAPRI and MOAL dataset. Each section in the barplot defines the number of targets for which each of the models could identify its near-native in top *n* ranks.

Here, we evaluate the model's performance on the independent dataset of CAPRI and MOAL discussed in chapter 6, section 6.1.3. Fig 7.4 shows the performance of the 3D-CNN model on the CAPRI and the MOAL dataset. Like every other evaluation, we evaluated our model on these datasets at for three different quality of near-natives. For both the datasets, we can see that the

number of near-natives identified decreases as we increase the DockQ score threshold for the near-native selection. This trend is justified since it is more likely to find a high number of low quality decoys at any top *n* rank as compared to high quality decoys. And vice versa, it is less likely for a low quality decoy to be present to top 1 ranks. This can be clearly seen from the bar plots as we get same number of decoys in top 1 ranks irrespective of the quality of the near-native we choose. This means that the model is able to rank the decoys correctly as it is able to identify same set of near-natives at top 1 ranks at any level of the criteria.

7.3 Performance of 3D-CNN vs Docking software

This section compares the model's performance to FRODOCK, and ZDOCK. We had only 3 targets for HADDOCK in the test set, so we discuss the HADDOCK results in appendix A. We could not acquire rankings for ClusPro software since ClusPro produces decoys in clusters based on decoys' quality. That means all similar quality decoys would be clustered together in a group, and hence there are no formal rankings that the ClusPro associates to the decoys it produces. Here we evaluate the model and the docking software's ability in identifying three different quality near-natives. In addition to this, we might find a different near-native decoy when we change the selection criteria. This is because, we always pick the first decoy for a target whose DockQ is closest to the selected threshold.

7.3.1 3D-CNN vs FRODOCK

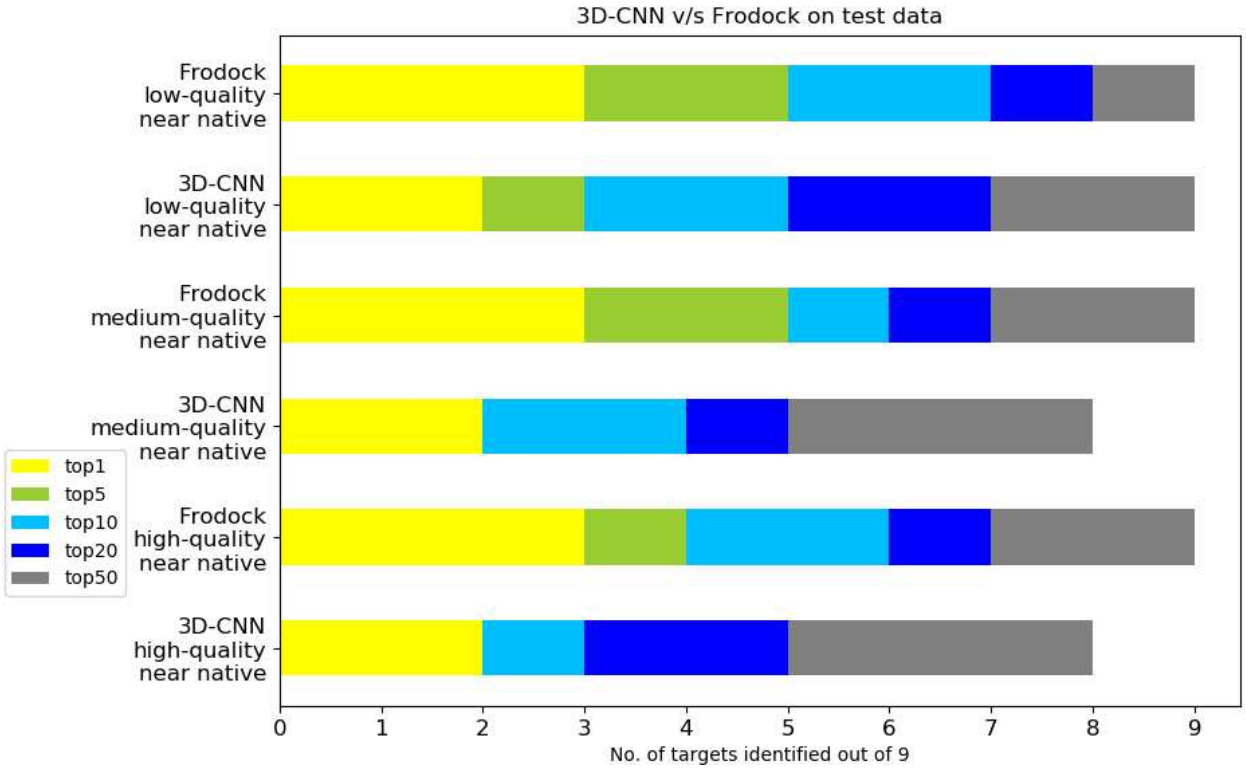


Figure 7.5: Model vs FRODOCK performance on test data. Each section in the barplot defined the number of targets for which the model and FRODOCK could identify its near-native in top *n* ranks. Here we assess the performance of model and the docking software for in identifying three different quality of decoys.

Figure 7.5 gives a comparison of model and the FRODOCK docking software's performance on 9 targets from the test set. Overall, FRODOCK has outperformed the 3D-CNN model in all the scenarios by identifying more first near-natives in all top *n* ranks at all evaluation criterias. For generating docked solutions for FRODOCK, we used FRODOCK2.0 standalone software. The underlying literature states that the FRODOCK2.0 was parameterized on a subset of DBD5 targets. Since our test is composed of DBD5 targets and given that FRODOCK has been fine tuned on a subset of this data, it is highly probable for FRODOCK to perform well on any similar dataset. This supports the higher performance of FRODOCK than the 3D-CNN model on our test set. Given that our model has no prior knowledge of DBD5 targets, it gave a near close performance to the FRODOCK software.

7.3.2 3D-CNN vs ZDOCK

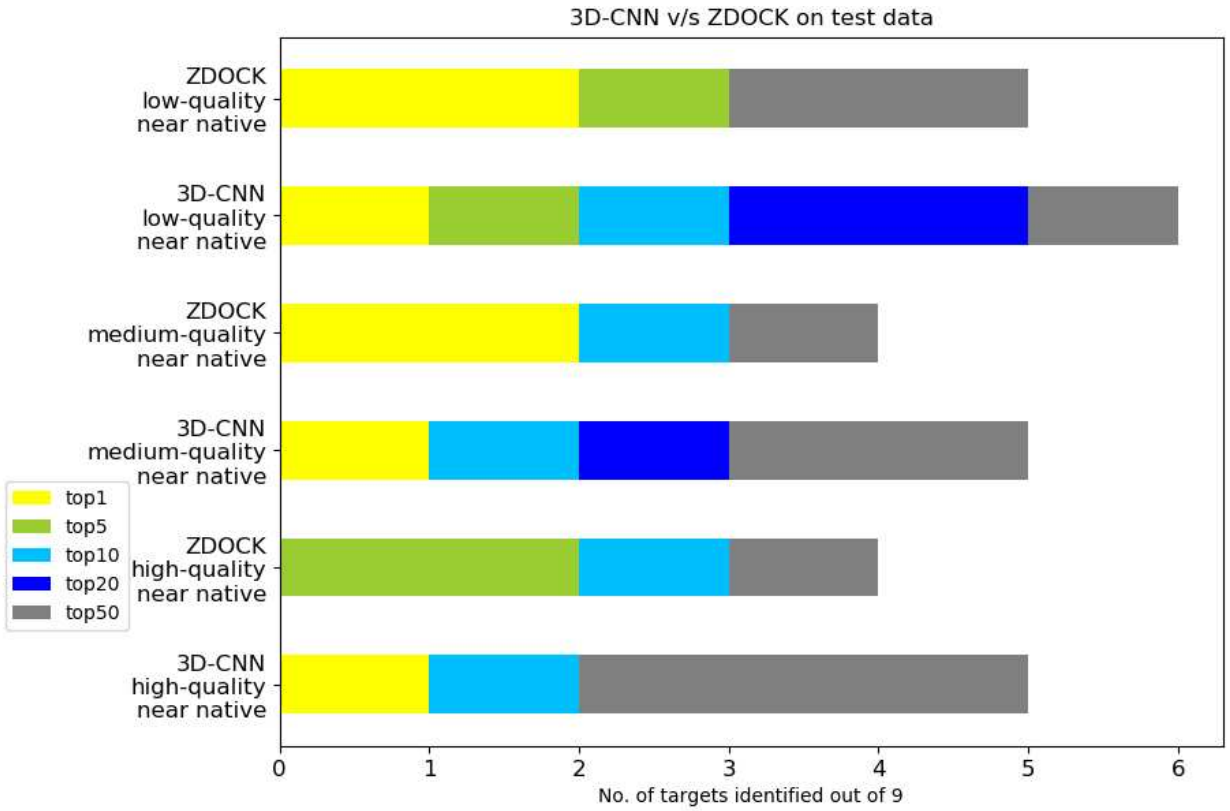


Figure 7.6: Model vs ZDOCK performance on test data. Each section in the barplot defined the number of targets for which the model and the corresponding docking software could identify its near-native in top n ranks.

Figure 7.6 gives a comparison of model and the ZDOCK docking software's performance on 9 targets from the test set. We can see that 3D-CNN is able to identify more near-natives in top 20 and top 50 ranks at every DockQ threshold criteria. However, ZDOCK could identify more near-natives at higher ranks than the 3D-CNN model. Only in the case of high-quality near-natives, 3D-CNN could identify one target in top 1 rank where as ZDOCK could not identify any. As mentioned in chapter 6, we added some targets from Docking Benchmark Dataset 4 [83] that were similar to the new targets from Docking Benchmark Dataset 5. The decoy dataset we used from ZDOCK consisted of decoys from Docking Benchmark Dataset 4, and were generated using ZDOCK:3.0.2 [95] which was optimized for all the targets from Docking Benchmark Dataset 4.

This justifies ZDOCK's better performance as all the decoys we had for ZDOCK in the test set were from Docking Benchmark Dataset 4. However, our 3D-CNN model is partially trained on Docking Benchmark Dataset 4 targets,but these targets were highly dissimilar to the test targets. Stated that, our 3D-CNN model performed nearly equivalent to ZDOCK.

7.4 3D-CNN vs DOVE on the test data

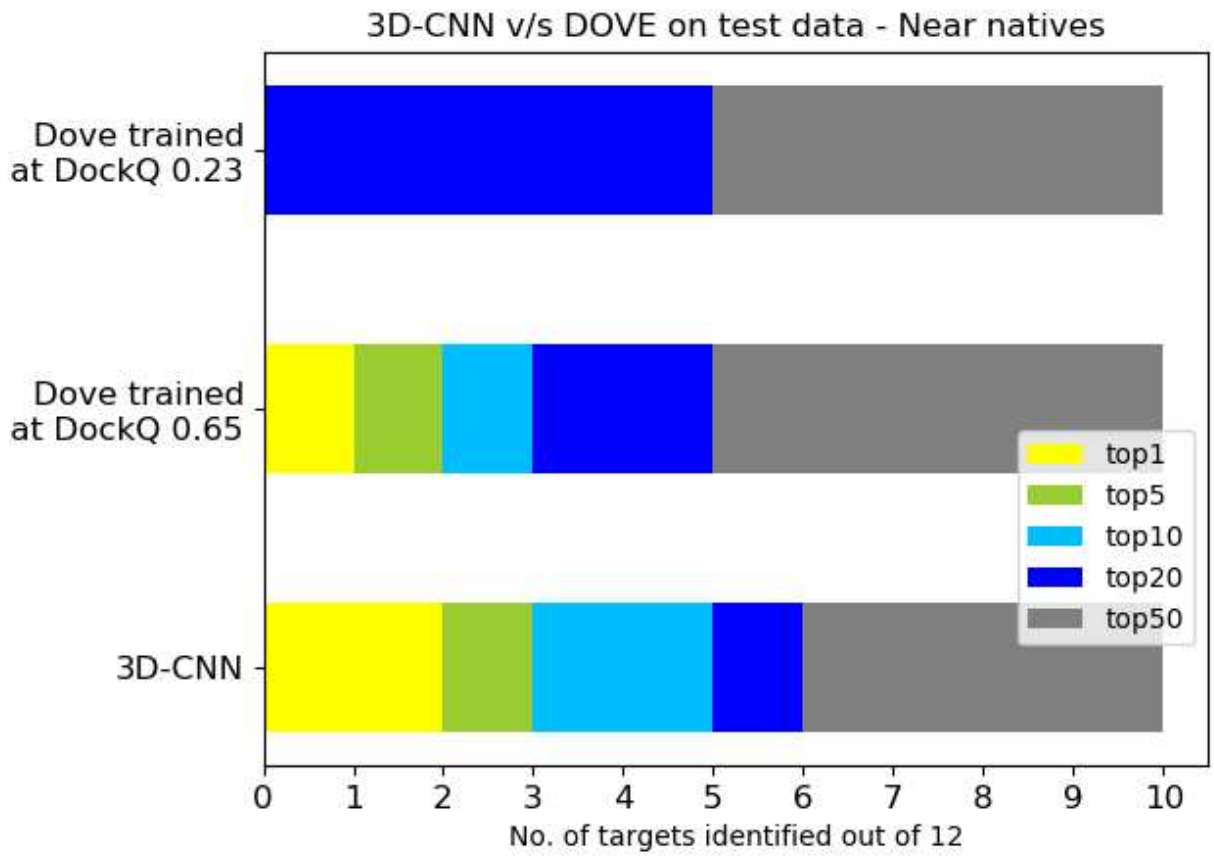


Figure 7.7: Model vs DOVE's performance on test data. Each section in the barplot defined the number of targets for which each of the models could identify its near-native in top *n* ranks.

This section compares our model's performance against an existing deep learning method for protein-protein interface assessment called DOVE. However, DOVE is designed to evaluate protein-protein docking decoys that use 3D-CNNs to identify the resulting interface is a true interface or not. On the other hand, our proposed model is designed to rank the decoy structures.

Ranking decoy structures is a much harder problem since it is not only associated with identifying a true interface but also with identifying its ranks with respect to other true interfaces. As a part of this thesis, our goal is not just to identify a true interface but also to find its order amongst hundreds and thousands of decoy structures corresponding to a target.

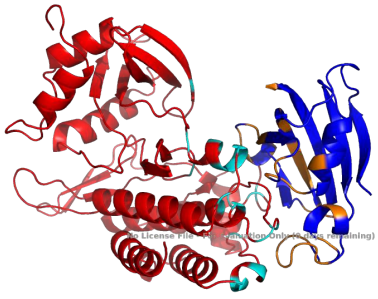
We retrained and tested the DOVE model from scratch on the same training and test set we used to train and evaluate our model. For the classification task, the decoys are labeled as positive or negative based on the DockQ cutoffs. The way the DOVE model is designed, it can differentiate between an acceptable and non-acceptable decoy interface. Referring to the DockQ criteria mentioned in table 6.4, an acceptable decoy has a DockQ score between 0.23 and 0.49. Initially, we used a DockQ cutoff of 0.23 to get the decoy interfaces' labels in the dataset. Later, to increase the DOVE model's robustness and sensitivity, we generated another set of labels for these decoys by using a DockQ cutoff of 0.65. the performance of these to independently trained models is compared against our 3D-CNN model in fig 7.7.

The DOVE model produces a probability value as an output that determines the quality of the input interface. A higher probability corresponds to a higher quality interface and vice versa. We used these output probability values to rank the near-native decoys for 12 targets from the test set which had at least one high quality decoy. Two things can be inferred from fig. Firstly, there is a slight improvement in DOVE's performance when the classification criteria were made more stringent (from DockQ score threshold of 0.23 to 0.65). This improvement can be observed from the increase in the number of targets identified by DOVE in the top 10 to top 1 ranks. Secondly, our 3D-CNN model's overall performance is better to the two DOVE models mentioned above. However, all the three models could identify same number of targets in top 50, but our model performed well at all other remaining ranks. Since the probability of a near-native being present in top n ranks increases as the value of n decreases, so we can say that a model that identifies more near-natives at higher ranks is a more robust model.

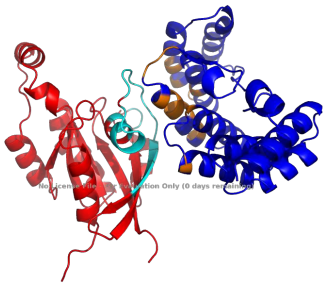
7.5 Visualization of correct and wrong predictions

In this section, we would be discussing some factors that caused the model to make a correct or wrong prediction. For each of the following subsections, we choose a decoy from each of the docking software so as to make a generalized inference.

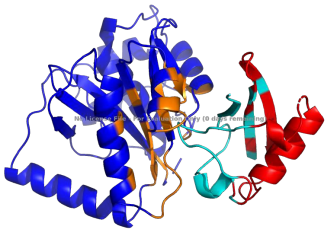
7.5.1 False positive



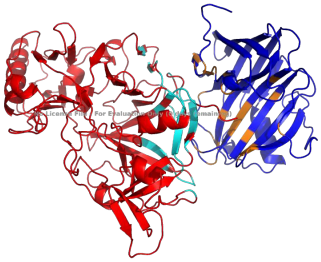
(a) Target 1B6C, decoy from ClusPro



(b) Target 1E96, decoy from Haddock



(c) Target 1XD3, decoy from ZDOCK



(d) Target 3VLB, decoy from FRODOCK

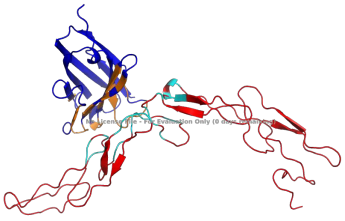
Figure 7.8: PyMol visualizations of decoys which have been ranked higher but have a low DockQ score. Here The two interacting proteins are colored in Red and Blue respectively and the portions of the corresponding proteins participating in the interface are colored in Cyan and Orange respectively

A false positive prediction refers to a mis-classifed decoy that has been ranked higher by the model but in reality, is a poor quality decoy. From the visualization in fig 7.8, what we can observe is, whenever the interface region for any one of the participating proteins is small (typically span-

ning over a very small region) as compared to the other partner, it has caused the model to consider it as a positive sample.

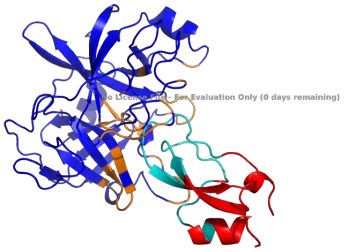
Typically, for a good quality model, both of the interacting proteins would have a small and nearly same size of the contributing interface region. So, for any decoy, if the interface region is small, due to the smaller size of the contributing chain, the model has considered it to be as a good quality decoy.

7.5.2 False negatives

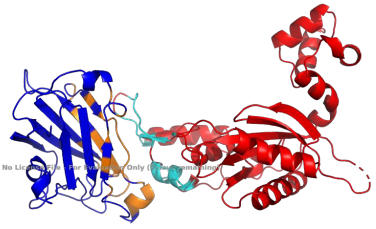


(a) Target 5DMJ, decoy from ClusPro

(b) Target 1E96, decoy from Haddock



(c) Target 1EAW, decoy from ZDOCK



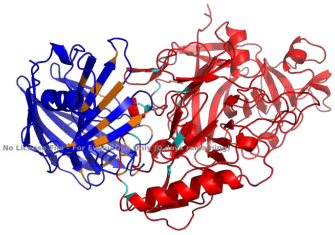
(d) Target 3K75, decoy from FRODOCK

Figure 7.9: PyMol visualizations of decoys which have been ranked lower but have a high DockQ score. Here The two interacting proteins are colored in Red and Blue respectively and the portions of the corresponding proteins participating in the interface are colored in Cyan and Orange respectively

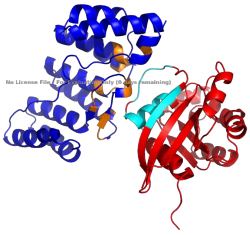
A false negative prediction refers to a mis-classifed decoy that has been ranked lower by the model but in reality, is a good quality decoy. From the visualization in fig 7.9, what we can observe is, if the size of the interface is large due to the greater width or length of the contributing chain,

we see a false negative. From the visualization, we can see that either of the participating proteins has a longer chain and a scattered interface region.

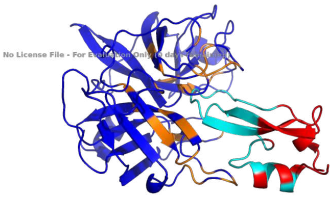
7.5.3 True positives



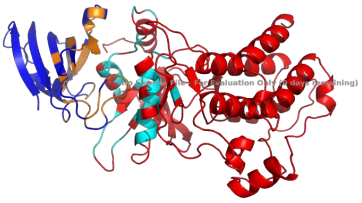
(a) Target 3VLB, decoy from ClusPro



(b) Target 1E96, decoy from Haddock



(c) Target 1EAW, decoy from ZDOCK



(d) Target 1B6C, decoy from FRODOCK

Figure 7.10: PyMol visualizations of decoys which have been ranked higher and have a high DockQ score as well. Here The two interacting proteins are colored in Red and Blue respectively and the portions of the corresponding proteins participating in the interface are colored in Cyan and Orange respectively

7.6 Conclusion

In this thesis we presented a 3D-CNN model for quality assessment of docked protein structures that uses raw atomic densities of atoms comprising the interface of the docked structure. When compared to the state-of-art docking softwares, we can see that our 3D-CNN model performed closely to FRODOCK and ZDOCK, noting that both the docking tools have been optimized and tuned partially/fully on our test targets. In the case of HADDOCK, our 3D-CNN model

outperformed the docking algorithm in finding near-natives of all qualities. Our model also outperformed the existing 3D-CNN method DOVE [78] even when it has been re-trained for a higher DockQ threshold. To conclude we can say that using simple features like raw atomic densities, our model has shown a good performance even with a limited good quality training data and has given on-par performance to some existing interface assessment methods.

Chapter 8

Future work

In this section, we will be discussing some possible directions for future work. Firstly, we can experiment with more possible input features. In our case, we consider atomic densities of the atoms at the docking interface. Likewise, we can consider other physical quantities such as electrostatic potential or solvent density. In addition to this, it is also possible to include other structure level features such as sequence conservation that could potentially boost performance.

Secondly, building a larger and better dataset. This can be done by bringing into consideration other protein database and other docking softwares. As mentioned in chapters 5 and 6, most of our dataset is composed of low quality decoys. Therefore we expect that adding more decoys with good quality to the dataset would definitely help the model to generalize well and differentiate more accurately between good and bad decoys.

Other areas of improvement include enhancing the loss function and its interpret-ability. If we look at the scores produced by the model, it is quite difficult to make a interpret them at a glance. For example, looking at the work published by Bau, David and Zhou, Bolei and Khosla [96], it has been shown that interpret-ability of models in computer vision can be improved by including fine-grained details about the labels such the bounding boxes and the high level features of the objects in it. In a similar way, we can use details such as amino acid environment and secondary structure elements for achieving a more precise judgement. Unlike standard deep learning models, these features would not be used as input for making predictions, rather they would be used for enhancing the interpret-ability.

In case of regression loss, we observed better performance when we used multiple labels rather than just the DockQ score. This definitely indicates that the model is learning better when more details are added for the label. Therefore we expect that a multi-variate ranking loss could potentially perform better as compared to a single variable ranking loss which we have used as a part of this thesis.

There is also a potential to use models that have been pre-trained on protein 3D structures. Transfer learning is one of most common methods used in deep learning applications [97]. Studies show that machine learning models that have been trained for a certain application if used as a starting point for another application, tend to learn and generalize better. Even if the existing and the new application have a different goal, the model for the existing application could still be useful for the new problem if they share common input data (e.g. protein 3D structures in our case). 3D-CNN Models like Proq3D [98] and SASNet [99] are some potential models that can be used as a starting point to solve our problem of docked protein interface assessment.

There are many modern and enhanced convolutional network architectures like AlexNet [74], VGGNet [71] etc. that have shown better performance on image data than the standard architecture we used in this thesis. Since we have used a 3D protein feature representation, it is highly probable that these architectures, if trained on protein data could potentially give better performance.

Finally, we will discuss the scope of improving the computation speed. The PyTorch 3D-CNN model used in this application is currently designed to run on a single GPU only. However, it is very feasible to extend it to run on multiple GPUs simultaneously. This would definitely reduce the total execution time of the model.

Bibliography

[1] Georgy Derevyanko, Sergei Grudinin, Yoshua Bengio, and Guillaume Lamoureux. Deep convolutional networks for quality assessment of protein folds. *Bioinformatics*, 34(23):4046–4053, 2018.

[2] Wikimedia Commons. File:genetic code.svg — wikimedia commons, the free media repository, 2019. [Online; accessed 24-May-2020].

[3] Wikimedia Commons. File:alpha beta structure (full).png — wikimedia commons, the free media repository, 2020. [Online; accessed 24-May-2020].

[4] Wikimedia Commons. File:beta sheets.svg — wikimedia commons, the free media repository, 2020. [Online; accessed 24-May-2020].

[5] Wikimedia Commons. File:osc microbio 07 04 proteinstr.jpg — wikimedia commons, the free media repository, 2019. [Online; accessed 24-May-2020].

[6] Wikimedia Commons. File:ligand-receptor interaction.png — wikimedia commons, the free media repository, 2020. [Online; accessed 5-October-2020].

[7] Wikimedia Commons. File:x ray diffraction.png — wikimedia commons, the free media repository, 2017. [Online; accessed 25-May-2020].

[8] Schrödinger, LLC. The PyMOL molecular graphics system, version 1.8. November 2015.

[9] Wikimedia Commons. File:docking representation 2.png — wikimedia commons, the free media repository, 2017. [Online; accessed 25-May-2020].

[10] Wikimedia Commons. File:overview docking.png — wikimedia commons, the free media repository, 2017. [Online; accessed 25-May-2020].

[11] Marc F Lensink, Sameer Velankar, Andriy Kryshtafovych, Shen-You Huang, Dina Schneidman-Duhovny, Andrej Sali, Joan Segura, Narcis Fernandez-Fuentes, Shruthi

Viswanath, Ron Elber, et al. Prediction of homoprotein and heteroprotein complexes by protein docking and template-based modeling: A CASP-CAPRI experiment. *Proteins: Structure, Function, and Bioinformatics*, 84:323–348, 2016.

[12] Alexander LeNail. NN-SVG: Publication-ready neural network architecture schematics. *Journal of Open Source Software*, 4(33):747, 2019.

[13] Wikimedia Commons. File:convolutional neural network with color image filter.gif — wikimedia commons, the free media repository, 2020. [Online; accessed 25-May-2020].

[14] Wikimedia Commons. File:same-padding-convolution.gif — wikimedia commons, the free media repository, 2019. [Online; accessed 25-May-2020].

[15] Wikimedia Commons. File:typical cnn.png — wikimedia commons, the free media repository, 2019. [Online; accessed 25-May-2020].

[16] Rin Sato and Takashi Ishida. Protein model accuracy estimation based on local structure quality assessment using 3D convolutional neural network. *PLOS ONE*, 14(9):e0221347, 2019.

[17] AT Balci, C Gumeli, A Hakouz, D Yuret, O Keskin, and Attila Gursoy. DeepInterface: Protein-protein interface validation using 3d convolutional neural networks. *bioRxiv*, page 617506, 2019.

[18] Wen Torng and Russ B Altman. 3D deep convolutional neural networks for amino acid environment similarity analysis. *BMC Bioinformatics*, 18(1):302, 2017.

[19] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[20] Suzanne Clancy and William Brown. Translation: DNA to mRNA to protein. *Nature Education*, 1(1):101, 2008.

[21] JW Pelley. 3-protein structure and function. *Elsevier's Integrated Biochemistry*, pages 19–28, 2007.

[22] Yael Avissar, Jung Choi, Jean DeSaix, Vladimir Jurukovski, Robert Wise, Connie Rye, et al. Biology: Openstax. 2018.

[23] Wikipedia contributors. Protein–protein interaction — Wikipedia, the free encyclopedia, 2020. [Online; accessed 12-March-2020].

[24] Wikibooks. Structural biochemistry/protein function/binding sites — wikibooks, the free textbook project, 2020. [Online; accessed 12-March-2020].

[25] Uros Kuzmanov and Andrew Emili. Protein-protein interaction networks: probing disease mechanisms using model systems. *Genome medicine*, 5(4):37, 2013.

[26] James A Wells and Christopher L McClendon. Reaching for high-hanging fruit in drug discovery at protein–protein interfaces. *Nature*, 450(7172):1001–1009, 2007.

[27] Wikibooks. Structural biochemistry/protein function/binding sites — wikibooks, the free textbook project, 2020. [Online; accessed 4-October-2020].

[28] Joël Janin, Ranjit P Bahadur, and Pinak Chakrabarti. Protein–protein interaction and quaternary structure. *Quarterly Reviews of Biophysics*, 41(2):133–180, 2008.

[29] X-ray protein crystallography - physics libretexts. https://phys.libretexts.org/Courses/University_of_California_Davis/UCD%3A_Biophysics_200A_-_Current_Techniques_in_Biophysics/X-ray_Protein_Crystallography. (Accessed on 05/27/2020).

[30] Jan Drenth. *Principles of protein X-ray crystallography*. Springer Science & Business Media, 2007.

[31] Wikipedia contributors. Nuclear magnetic resonance spectroscopy of proteins — Wikipedia, the free encyclopedia, 2020. [Online; accessed 28-May-2020].

[32] Nurcan Tuncbag, Attila Gursoy, and Ozlem Keskin. Prediction of protein–protein interactions: unifying evolution and structure at protein interfaces. *Physical Biology*, 8(3):035006, 2011.

[33] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[34] B. Shariat, D. Neumann, and A. Ben-Hur. BLRM: A basic linear ranking model for protein interface prediction. In *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 29–35, 2018.

[35] Protein docking. https://www.slideshare.net/TNAUgenomics/protein-docking. (Accessed on 04/06/2020).

[36] Pedro J Ballester and John BO Mitchell. A machine learning approach to predicting protein–ligand binding affinity with applications to molecular docking. *Bioinformatics*, 26(9):1169–1175, 2010.

[37] Joel Janin. Welcome to CAPRI: a critical assessment of predicted interactions. *Proteins: Structure, Function, and Bioinformatics*, 47(3):257–257, 2002.

[38] Aleksey Porollo and Jarosław Meller. Prediction-based fingerprints of protein–protein interactions. *Proteins: Structure, Function, and Bioinformatics*, 66(3):630–645, 2007.

[39] Shide Liang, Chi Zhang, Song Liu, and Yaoqi Zhou. Protein binding site prediction using an empirical scoring function. *Nucleic Acids Research*, 34(13):3698–3707, 2006.

[40] Huiling Chen and Huan-Xiang Zhou. Prediction of interface residues in protein–protein complexes by a consensus neural network method: test against nmr data. *Proteins: Structure, Function, and Bioinformatics*, 61(1):21–35, 2005.

[41] Sanbo Qin and Huan-Xiang Zhou. meta-PPISP: a meta web server for protein-protein interaction site prediction. *Bioinformatics*, 23(24):3386–3387, 2007.

[42] Sjoerd J de Vries and Alexandre MJJ Bonvin. CPORT: a consensus interface predictor and its performance in prediction-driven docking with HADDOCK. *PLOS ONE*, 6(3):e17695, 2011.

[43] Hani Neuvirth, Ran Raz, and Gideon Schreiber. ProMate: a structure based prediction program to identify the location of protein–protein binding sites. *Journal of Molecular Biology*, 338(1):181–199, 2004.

[44] Sjoerd J de Vries, Aalt DJ van Dijk, and Alexandre MJJ Bonvin. WHISCY: what information does surface conservation yield? application to data-driven docking. *Proteins: Structure, Function, and Bioinformatics*, 63(3):479–489, 2006.

[45] Irina Kufareva, Levon Budagyan, Eugene Raush, Maxim Totrov, and Ruben Abagyan. PIER: protein interface recognition for structural proteomics. *Proteins: Structure, Function, and Bioinformatics*, 67(2):400–417, 2007.

[46] Li C Xue, Drena Dobbs, Alexandre MJJ Bonvin, and Vasant Honavar. Computational prediction of protein interfaces: A review of data driven methods. *FEBS letters*, 589(23):3516–3526, 2015.

[47] Rong Chen, Li Li, and Zhiping Weng. ZDOCK: an initial-stage protein-docking algorithm. *Proteins: Structure, Function, and Bioinformatics*, 52(1):80–87, 2003.

[48] José Ignacio Garzon, José Ramón Lopéz-Blanco, Carles Pons, Julio Kovacs, Ruben Abagyan, Juan Fernandez-Recio, and Pablo Chacon. FRODOCK: a new approach for fast rotational protein–protein docking. *Bioinformatics*, 25(19):2544–2551, 2009.

[49] Dima Kozakov, David R Hall, Bing Xia, Kathryn A Porter, Dzmitry Padhorny, Christine Yueh, Dmitri Beglov, and Sandor Vajda. The ClusPro web server for protein–protein docking. *Nature Protocols*, 12(2):255, 2017.

[50] Sjoerd J De Vries, Marc Van Dijk, and Alexandre MJJ Bonvin. The HADDOCK web server for data-driven biomolecular docking. *Nature protocols*, 5(5):883, 2010.

[51] Susana Cristobal, Adam Zemla, Daniel Fischer, Leszek Rychlewski, and Arne Elofsson. A study of quality measures for protein threading models. *BMC bioinformatics*, 2(1):5, 2001.

[52] S Agatonovic-Kustrin and Rosemary Beresford. Basic concepts of artificial neural network (ann) modeling and its application in pharmaceutical research. *Journal of Pharmaceutical and Biomedical analysis*, 22(5):717–727, 2000.

[53] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[54] Augustin Cauchy et al. Méthode générale pour la résolution des systèmes d'équations simultanées. *Comp. Makes. Sci. Paris*, 25(1847):536–538, 1847.

[55] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[56] Mu Li, Tong Zhang, Yuqiang Chen, and Alexander J Smola. Efficient mini-batch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 661–670, 2014.

[57] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(7), 2011.

[58] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[59] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[60] Adam P Piotrowski and Jarosław J Napiorkowski. A comparison of methods to avoid overfitting in neural networks training in the case of catchment runoff modelling. *Journal of Hydrology*, 476:97–111, 2013.

[61] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[62] N. Morgan and H. Bourlard. Generalization and parameter estimation in feedforward nets: Some experiments. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 630–637. Morgan-Kaufmann, 1990.

[63] Anders Krogh and John A. Hertz. A simple weight decay can improve generalization. In J. E. Moody, S. J. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems 4*, pages 950–957. Morgan-Kaufmann, 1992.

[64] Ping Luo, Xinjiang Wang, Wenqi Shao, and Zhanglin Peng. Towards understanding regularization in batch normalization. *arXiv preprint arXiv:1809.00846*, 2018.

[65] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, 2019.

[66] Convolutional neural network - wikipedia. https://en.wikipedia.org/wiki/Convolutional_neural_network. (Accessed on 04/30/2020).

[67] Convolution - wikipedia. https://en.wikipedia.org/wiki/Convolution. (Accessed on 04/30/2020).

[68] A gentle introduction to pooling layers for convolutional neural networks. https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/. (Accessed on 05/04/2020).

[69] Y-Lan Boureau, Jean Ponce, and Yann LeCun. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 111–118, 2010.

[70] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The Handbook of Brain Theory and Neural Networks*, 3361(10):1995, 1995.

[71] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[72] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[73] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[74] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25:1097–1105, 2012.

[75] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[76] Yoav Goldberg. Neural network methods for natural language processing. *Synthesis Lectures on Human Language Technologies*, 10(1):1–309, 2017.

[77] Wenlin Li, R Dustin Schaeffer, Zbyszek Otwinowski, and Nick V Grishin. Estimation of uncertainties in the global distance test (GDT_TS) for CASP models. *PLOS ONE*, 11(5):e0154786, 2016.

[78] Xiao Wang, Genki Terashi, Charles W Christoffer, Mengmeng Zhu, and Daisuke Kihara. Protein docking model evaluation by 3D deep convolutional neural networks. *Bioinformatics*, 36(7):2113–2118, 11 2019.

[79] Sheng-You Huang and Xiaoqin Zou. An iterative knowledge-based scoring function for protein–protein recognition. *Proteins: Structure, Function, and Bioinformatics*, 72(2):557–579, 2008.

[80] Hongyi Zhou and Jeffrey Skolnick. Goap: a generalized orientation-dependent, all-atom statistical potential for protein structure prediction. *Biophysical Journal*, 101(8):2043–2052, 2011.

[81] Sankar Basu and Björn Wallner. DockQ: A quality measure for protein-protein docking models. *PLOS ONE*, 11(8):e0161879, 2016.

[82] Helen M. Berman, John Westbrook, Zukang Feng, Gary Gilliland, T. N. Bhat, Helge Weissig, Ilya N. Shindyalov, and Philip E. Bourne. The Protein Data Bank. *Nucleic Acids Research*, 28(1):235–242, 01 2000.

[83] Howook Hwang, Thom Vreven, Joël Janin, and Zhiping Weng. Protein–protein docking benchmark version 4.0. *Proteins: Structure, Function, and Bioinformatics*, 78(15):3111–3114, 2010.

[84] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[85] Tie-Yan Liu. *Learning to rank for information retrieval*. Springer Science & Business Media, 2011.

[86] Pointwise vs. pairwise vs. listwise learning to rank | by nikhil dandekar | medium. https://medium.com/@nikhilbd/pointwise-vs-pairwise-vs-listwise-learning-to-rank-80a8fe8fadfd. (Accessed on 11/29/2020).

[87] Thom Vreven, Iain H Moal, Anna Vangone, Brian G Pierce, Panagiotis L Kastritis, Mieczyslaw Torchala, Raphael Chaleil, Brian Jiménez-García, Paul A Bates, Juan Fernandez-Recio, et al. Updates to the integrated protein–protein interaction benchmarks: docking benchmark version 5 and affinity benchmark version 2. *Journal of Molecular Biology*, 427(19):3031–3041, 2015.

[88] Florian Krull, Gerrit Korff, Nadia Elghobashi-Meinhardt, and Ernst-Walter Knapp. Propairs: a data set for protein–protein docking. *Journal of Chemical Information and Modeling*, 55(7):1495–1507, 2015.

[89] Dominique Douguet, Huei-Chi Chen, Andrey Tovchigrechko, and Ilya A Vakser. Dockground resource for studying protein–protein interfaces. *Bioinformatics*, 22(21):2612–2618, 2006.

[90] Alexey G Murzin, Steven E Brenner, Tim Hubbard, and Cyrus Chothia. Scop: a structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology*, 247(4):536–540, 1995.

[91] Limin Fu, Beifang Niu, Zhengwei Zhu, Sitao Wu, and Weizhong Li. Cd-hit: accelerated for clustering the next-generation sequencing data. *Bioinformatics*, 28(23):3150–3152, 2012.

[92] Peter JA Cock, Tiago Antao, Jeffrey T Chang, Brad A Chapman, Cymon J Cox, Andrew Dalke, Iddo Friedberg, Thomas Hamelryck, Frank Kauff, Bartek Wilczynski, et al. Biopython: freely available python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25(11):1422–1423, 2009.

[93] Sankar Basu and Björn Wallner. Finding correct protein–protein docking models using proqdock. *Bioinformatics*, 32(12):i262–i270, 2016.

[94] C Geng, LC. Xue, and A Bonvin. Docking models for docking benchmark 4, 5 and CAPRI score_set., 2019.

[95] Brian G Pierce, Yuichiro Hourai, and Zhiping Weng. Accelerating protein docking in ZDOCK using an advanced 3d convolution library. *PLOS ONE*, 6(9):e24657, 2011.

[96] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6541–6549, 2017.

[97] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2009.

[98] Karolis Uziela, David Menendez Hurtado, Nanjiang Shu, Björn Wallner, and Arne Elofsson. Proq3d: improved model quality assessments using deep learning. *Bioinformatics*, 33(10):1578–1580, 2017.

[99] Raphael JL Townshend, Rishi Bedi, Patricia A Suriana, and Ron O Dror. End-to-end learning on 3d protein structure for interface prediction. *arXiv preprint arXiv:1807.01297*, 2018.

Appendix A

Supplementary Data

A.1 Atom types used for 11 channels

Table A.1: List of atoms used for 11 channels used as a part of the experiments in this thesis. This table is obtained from the publication *Deep convolutional networks for quality assessment of protein folds* by Derevyanko, Grudinin, Bengio, and Lamoureux [1]

Atoms description		
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

A.2 PyTorch model architecture

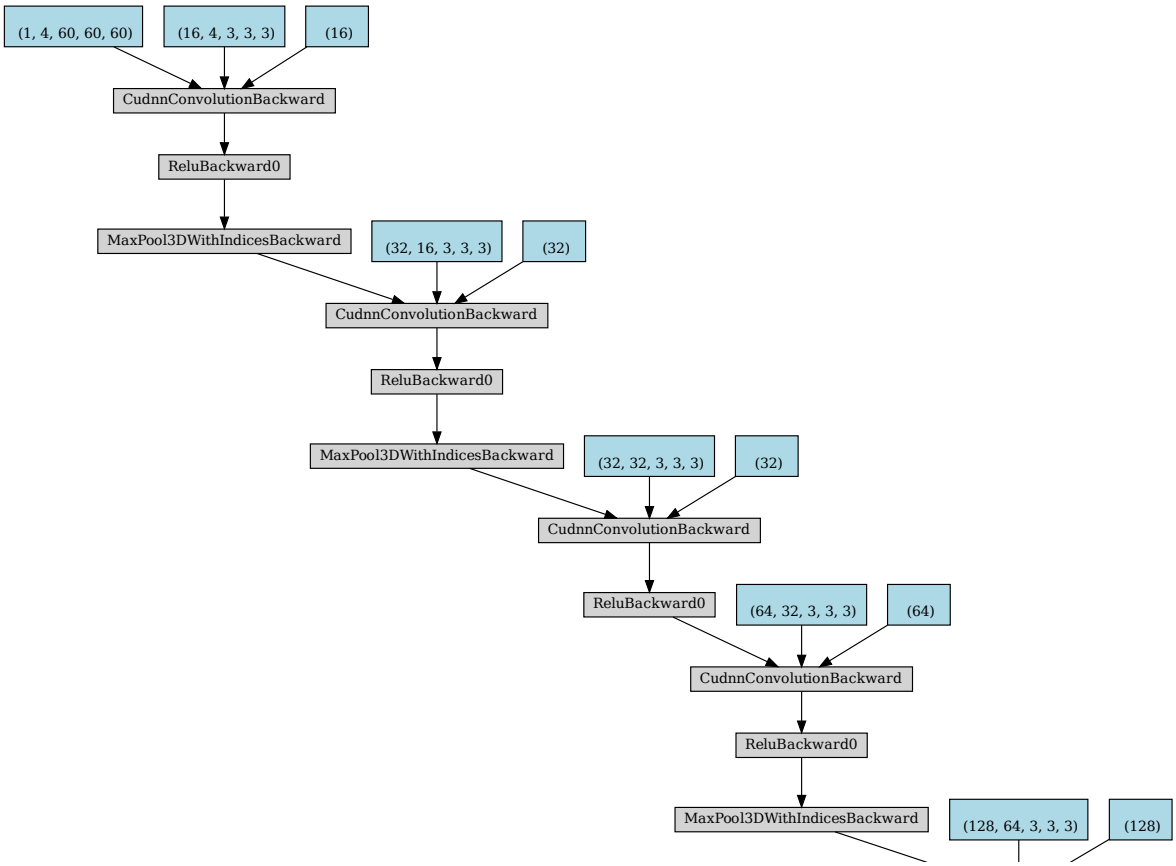


Figure A.1: Convolutional layers of the PyTorch model visualized using TorchViz tool. Each layer is associated with input dimension, filter size and bias for that layer

architecture contd..

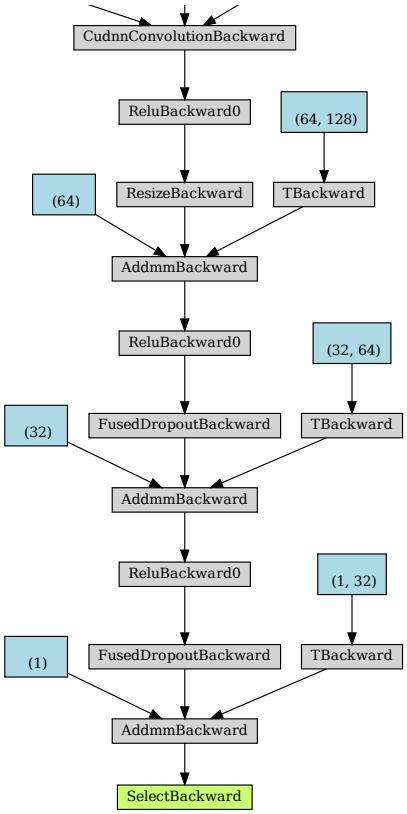


Figure A.2: The final convolutional block is connected to a block consisting of three fully connected layers

A.3 3DCNN vs HADDOCK

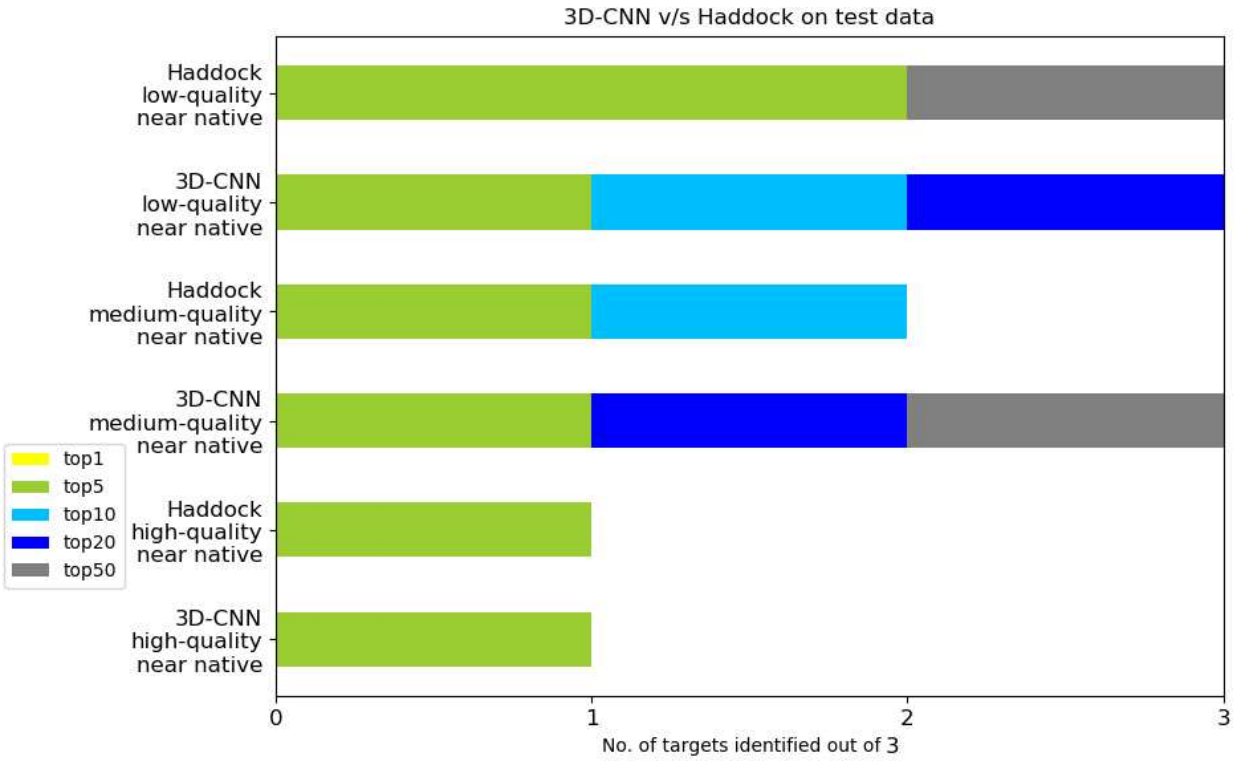
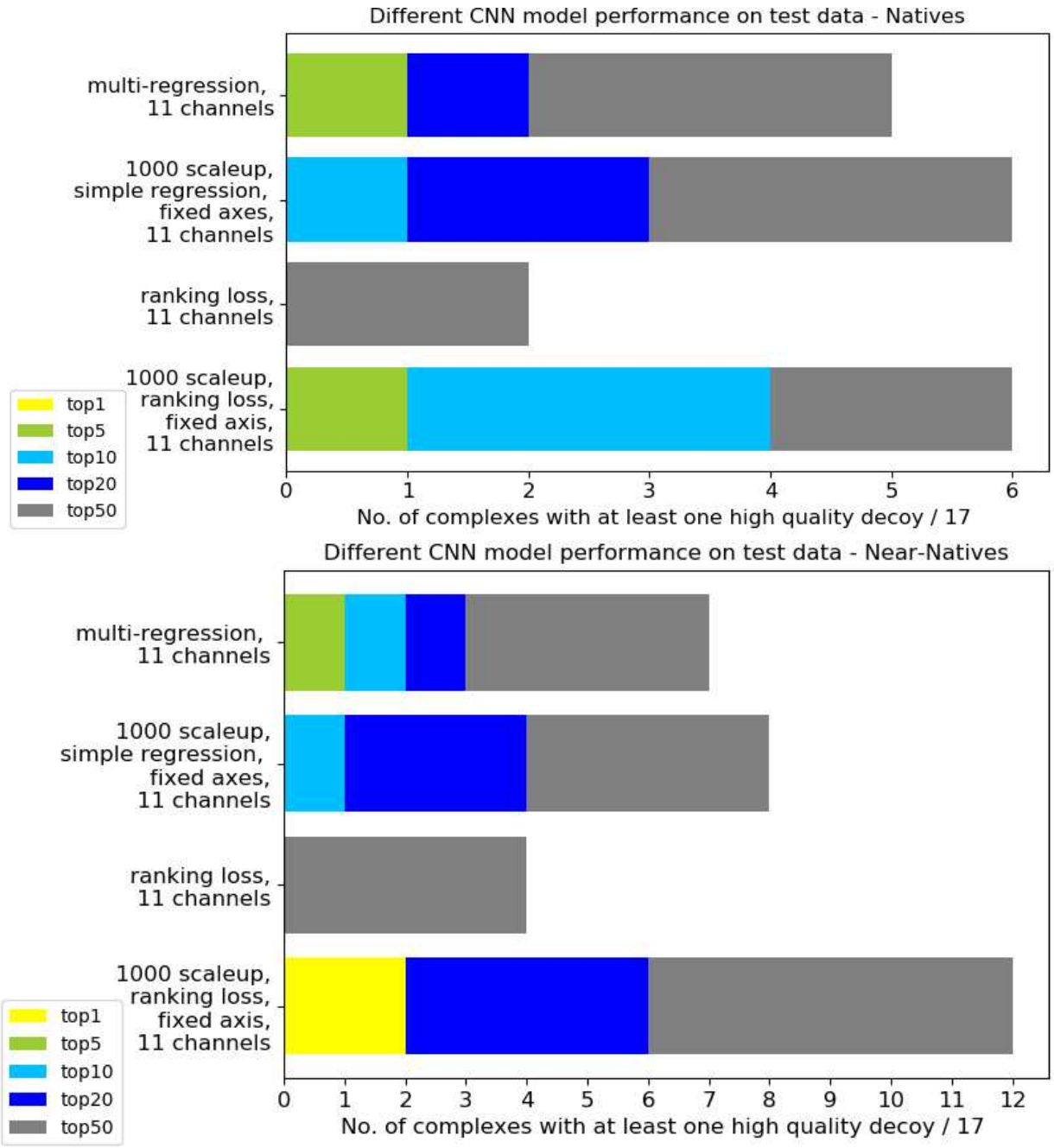


Figure A.3: Model vs HADDOCK performance on test data. Each section in the barplot defined the number of targets for which the model and HADDOCK could identify its near-native in top *n* ranks. Here we assess the performance of model and the docking software for in identifying three different quality of decoys.

Lookign at the overall performance of 3D-CNN on HADDOCK decoys, we can say that both 3D-CNN and HADDOCK docking software showed nearly same performance. A fewer number of targets for HADDOCK in the test set makes this conclusion difficult to justify.

A.4 Performance of 3D-CNN with 11 channels



We can see that scaling up the atomic density values showed some improvement in case of 11 channels. In addition to this, ranking loss showed better performance than regression loss.

A.5 Data augmentation experiments on 11 channels

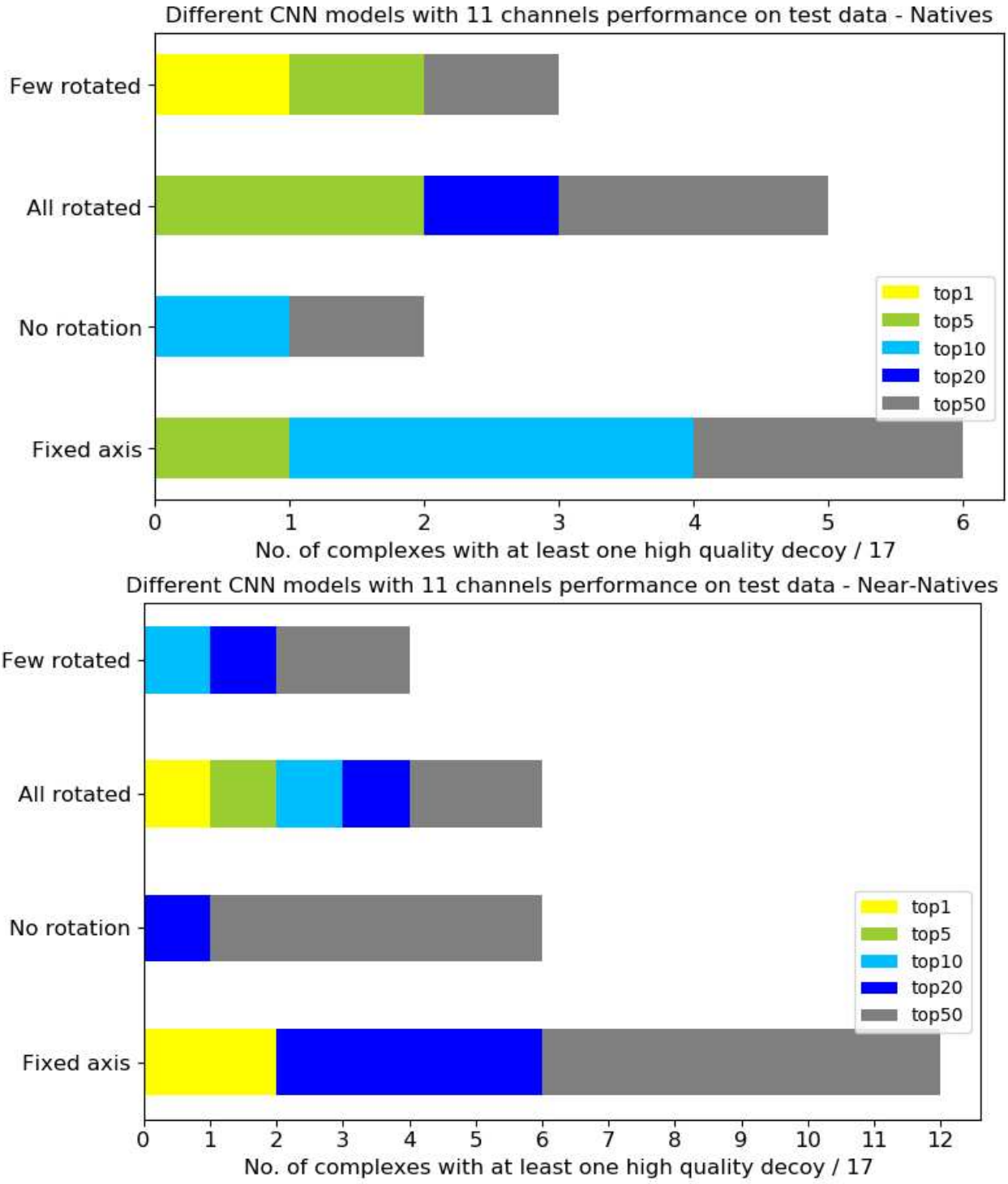


Figure A.4: Data augmentation experiments for 11 channeled 3D-CNN

We can clearly see that having a fixed axis for the interface helped the model to learn better. In contrast, having no rotation at all showed a very poor performance. For rotation experiments, wither we randomly rotated half of the samples from the batch or all of the samples for a batch. We can see that having all the samples rotated has shown a better performance than having only few rotated. All the rotations for these experiments we performed on fly, unlike the fixed axis experiment where the features for the interface we pre-computed before the training process.

A.6 Python dependencies

```
backports.functools-lru-cache==1.5
biopython==1.72
biopython-extensions==0.18.6
certifi==2019.6.16
chardet==3.0.4
cycler==0.10.0
enum34==1.1.6
fastrlock==0.4
funcsigs==1.0.2
future==0.17.1
graphviz==0.8.4
h5py==2.9.0
idna==2.8
kiwisolver==1.1.0
llvmlite==0.29.0
matplotlib==2.2.4
numba==0.44.1
```

Python libraries contd..

```
numpy==1.16.6
pandas==0.24.2
Pillow==6.0.0
pyparsing==2.4.0
pysam==0.15.2
python-dateutil==2.8.0
pytz==2019.1
PyYAML==5.1.1
requests==2.22.0
scikit-learn==0.20.4
scipy==1.2.2
semantic-version==2.6.0
singledispatch==3.4.0.3
six==1.12.0
subprocess32==3.5.4
tabulate==0.8.3
Theano==1.0.4
torch==1.4.0
torchsummary==1.5.1
torchvision==0.3.0
torchviz==0.0.1
urllib3==1.25.3
```