THESIS

QUALITY ASSESSMENT OF PROTEIN STRUCTURES USING GRAPH CONVOLUTIONAL NETWORKS

Submitted by

Soumyadip Roy

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Spring 2024

Master's Committee:

Advisor: Asa Ben-Hur

Nathaniel Blanchard
Wen Zhou

ABSTRACT

QUALITY ASSESSMENT OF PROTEIN STRUCTURES USING GRAPH CONVOLUTIONAL NETWORKS

The prediction of protein 3D structure is essential for understanding protein function, drug discovery, and disease mechanisms; with the advent of methods like AlphaFold that are capable of producing very high quality decoys, ensuring the quality of those decoys can provide further confidence in the accuracy of their predictions.

In this work we describe Q_ϵ, a graph convolutional network that utilizes a minimal set of atom and residue features as input to predict the global distance test total score (GDTTS) and local distance difference test score (lDDT) of a decoy. To improve the model's performance, we introduce a novel loss function based on the ϵ-insensitive loss function used for SVM-regression. This loss function is specifically designed for the characteristics of the quality assessment problem, and provides predictions with improved accuracy over standard loss functions used for this task. Despite using only a minimal set of features, it matches the performance of recent state-of-the-art methods like DeepUMQA.

The code for Q_ϵ is available at https://github.com/soumyadip1997/qepsilon.

ACKNOWLEDGEMENTS

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

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Proteins are found in all biological systems, from unicellular organisms to multicellular species like humans [5] and understanding their structure is essential for deciphering their biological functions [6]. The prediction of a protein's three-dimensional structure from its amino-acid sequence has been a longstanding challenge in computational biology [7].

In recent years, advances in protein structure prediction have been driven by the development of novel techniques and the growth of sequence and structure databases [8]. Many attempts are being undertaken in wet laboratories to identify physiologically native tertiary structures in order to decode protein functions [9]. Experimenting to identify protein structure takes time and money, but computational methods are far faster and less expensive. Technological improvements now allow for the production of hundreds of thousands of tertiary structures, known as decoys, in a matter of CPU hours [10]. Methods that predict structure produce a large number of decoys and are not great at finding the best ones, so it is necessary to identify high-quality, near-native decoys among hundreds of thousands of decoys in an ensemble [11]. This stage in identifying the near native decoys is also known as the quality assessment stage.

In this work we address the decoy quality assessment problem with the help of graph convolutional networks; we introduce a novel loss function inspired by the support vector regression ϵ-insensitive loss function that is designed to take into account our intuition about what makes for a good quality assessment predictor, namely that it focus on making correct predictions for those decoys that matter: those with high quality. We compare our method, called Q_ϵ, to other state-of-the-art methods and demonstrate that our method outperforms most of those methods while using only a very basic set of features computed from a decoy's sequence, without the need for engineered features.

The remainder of this thesis is organized as follows. The remainder of the introduction includes a basic explanation of proteins as well as a quick outline of quality assessment. Chapter 2 looks at

previous research in the topic of quality assessment. In Chapter 3, we'll learn how to characterize a protein as a graph and look at some of the ways to apply graph convolution to protein structures. In Chapter 4 and Chapter 5 we discuss about the graph architecture, pooling method, loss function and atom and residual features that are input into our network. In Chapter 6 we describe the experimental setup, which includes training, testing and validation data as well as the experimental approach and important metrics used to evaluate our various graph convolution algorithm. Finally in Chapter 7 we evaluate all of the findings from our investigations and outline future work that are based on the findings of this thesis.

1.1 Proteins

Cells are the basic building blocks of all living organisms on Earth. There are two main types of cells: prokaryotic and eukaryotic. Prokaryotic cells are smaller and simpler in structure, lacking a nucleus and other membrane-bound organelles. Eukaryotic cells, on the other hand, are more complex and larger in size, with a distinct nucleus and other organelles enclosed in membranes. Every eukaryotic cell consist of three primary components: the cell membrane, cytoplasm, and nucleus. The cell membrane acts as a selectively permeable barrier, regulating the entry and exit of substances into and out of the cell. It also plays a role in cellular recognition and communication. The cytoplasm is the gel-like substance that fills the cell providing a platform for cellular organelles to carry out their functions. It contains various molecules that support metabolic reactions. The nucleus is a crucial component of the eukaryotic cell. It houses the genetic material in the form of chromatin threads, which are consist of deoxyribonucleic acid or DNA. The nucleus has the ability to control the cell's shape and function, and it is also responsible for directing cellular activities, such as protein synthesis and cell division.

We now arrive at the central dogma of molecular biology [13], which is concerned with the transfer of genetic information from DNA to proteins via RNA. It implies that all of the information required to build a protein is contained inside a DNA sequence and is communicated to the ribosomes via a messenger RNA. Ribosomes aid in the translation of DNA data into the final

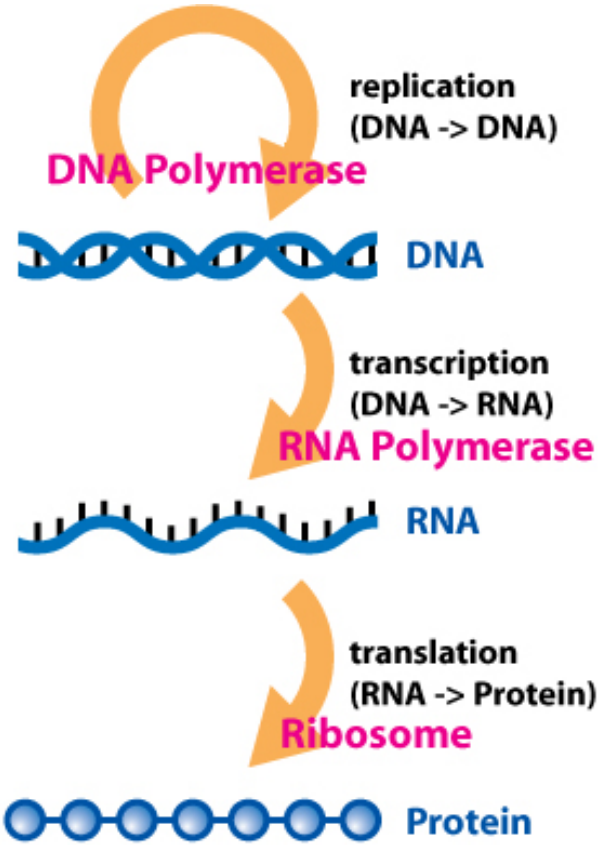


Figure 1.1: The three stages of the central dogma i.e. replication, transcription and translation [12].

product, which is a protein. This is referred to as gene expression. The Central Dogma can be described in three stages as illustrated in Figure 1.1 the first stage is DNA replication, in which a single parental double-stranded DNA molecule is duplicated into two double-stranded DNA molecules by the enzyme DNA polymerase. The second stage is transcription, during which messenger RNAs (mRNA) are synthesized from DNA by RNA polymerase, and the third stage is translation, during which the ribosomes read the messenger RNA sequence and use it to assemble a specific sequence of amino acids into a protein. Amino acids are a class of organic molecules that include at least an alpha carbon atom, an amine group, a single hydrogen atom and a carboxyl group (COOH). It contains a side chain that is connected to the α-carbon. The side chain differentiates amino acid from each other. Also, the side chain has an effect on the amino acid's chemical characteristics. Figure 1.2 shows the 21 amino acids based on the chemical properties exhibited by their side chains.

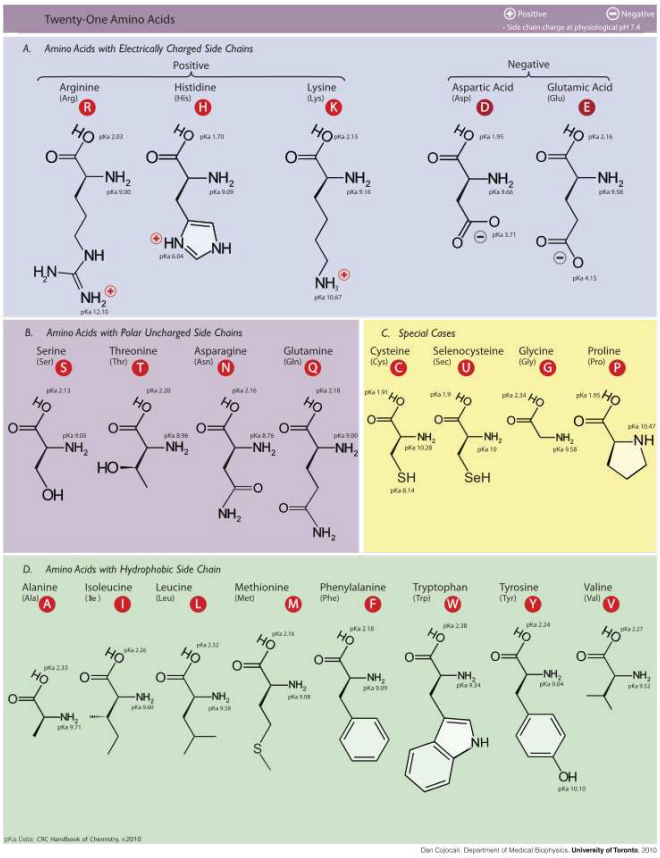


Figure 1.2: Amino acids and their chemical properties, determined by their side chains [14].

1.2 Protein Structure

Proteins can have primary, secondary, tertiary and quaternary structures. The primary structure of a protein is a linear sequence of amino acid residues contained within a single polypeptide. The amino acids in a polypeptide chain are held together by peptide bonds, which are double bonds that do not rotate. However, the other single bonds within the polypeptide chain do rotate, and as a result of this rotation, the linear polypeptide chain twists into regular patterns that form secondary structures. The α helix (Figure 1.3) and β sheets (Figure 1.4) are the two main examples of secondary structures.

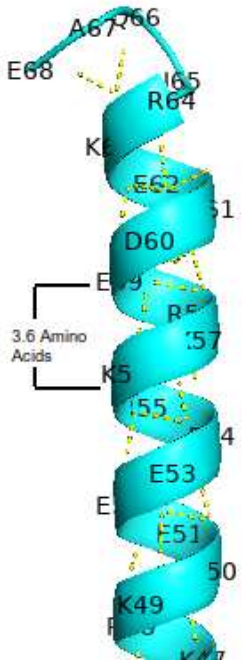
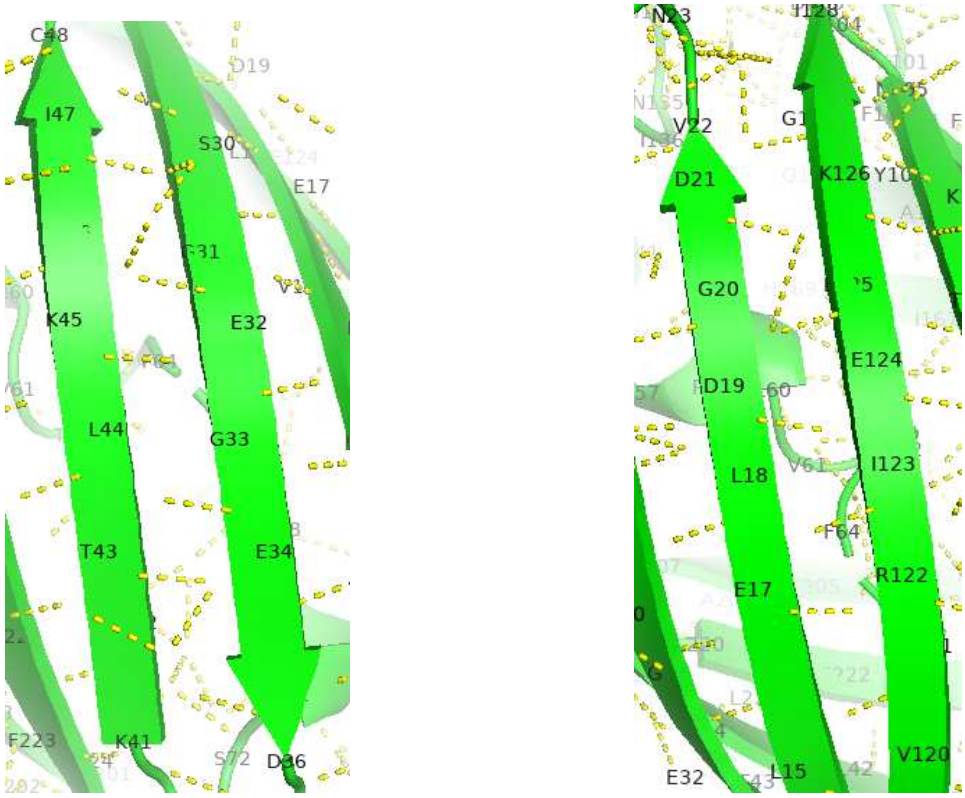


Figure 1.3: Part of the α helix from the 6OBI pdb file which is obtained from the pdb website. The residues have been represented by a single letter along with their residue number and the hydrogen bonding between the residues has been represented by yellow lines. Hydrogen bonding occurs between the i^{th} and the $(i+4)^{th}$ residue.Created with PyMol [15] and draw.io [16].



(a) Part of anti parallel beta sheets of the protein with pdb id 1EMA.

(b) Part of parallel beta sheets of the protein with pdb id 1EMA.

Figure 1.4: A part of parallel and anti parallel beta sheets of a protein with pdb id 1EMA. Residues are represented as letters along with their residue number and yellow lines represent the hydrogen bonding. Created with PyMol [15] and draw.io [16].

Tertiary structure refers to the folding of secondary structures into distinct arrangements (also known as domains) due to the properties of amino acid side chains [17]. The tertiary structure of a protein is critical for its biological function, as it plays a key role in determining the protein's binding specificity, enzymatic activity, and overall conformational dynamics [18]. The arrangement of multiple polypeptide chains within a protein complex is referred to as quaternary structure.

1.3 The Quality Assessment Problem

Prediction of protein 3D structures from sequence is one of the challenging problems in molecular biology [19]. To predict a protein structure many methods have been developed [20–22].

Computational methods for predicting a protein's 3D structure produce large numbers of decoy conformations for a given target protein. In quality assessment we seek to rank these decoys based on their similarity to the experimentally determined native structure. In most scenarios, the native structure is not available and a large number of decoys produced by the computational methods are not good. So we need a quality assessment (QA) stage to rank the decoys based on how well they represent the experimentally determined native structure. We collect a large number of decoys in connection to a target protein, and our goal is to determine which decoy has a structure similar to the target protein. This structural similarity is quantified using a score, which we shall go over in Chapter 5.

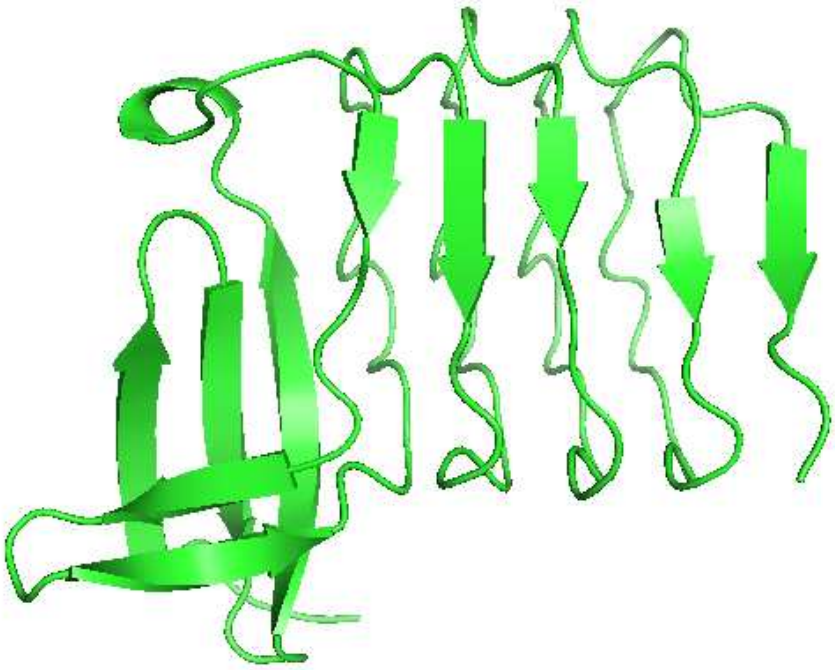
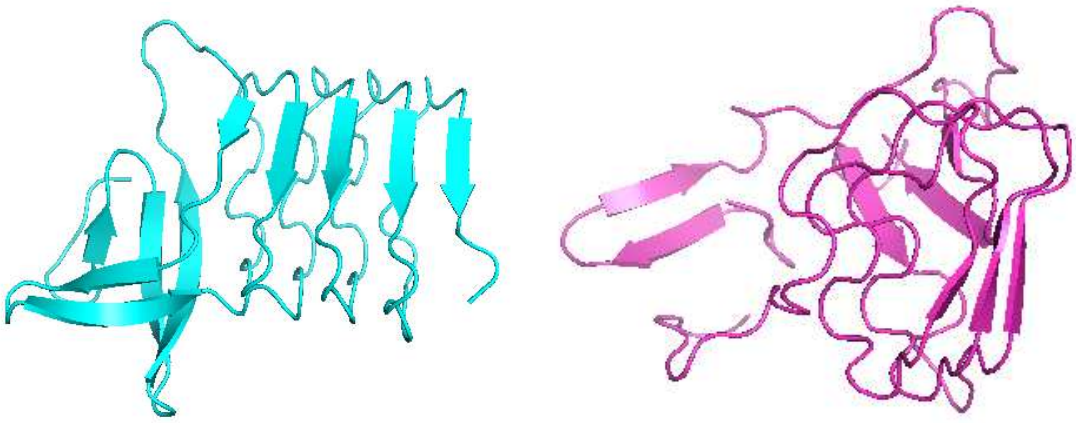


Figure 1.5: Target protein T0768-D1 from CASP11 dataset

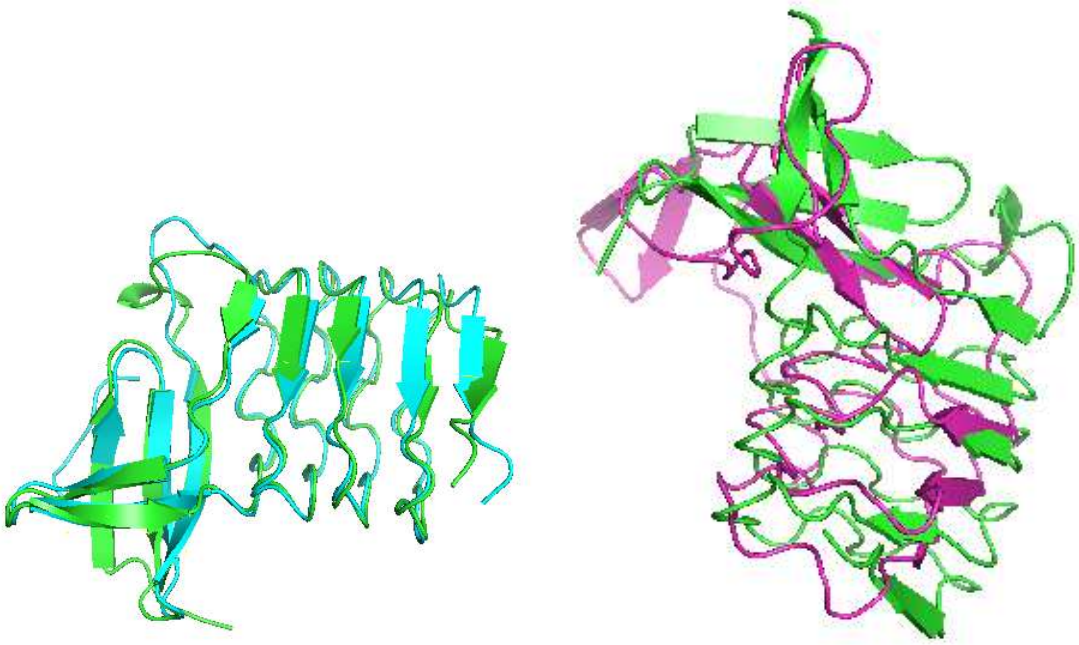
Figure 1.5 shows the image of the target protein, T0768-D1, from the CASP11 dataset and the images of two of its decoys, T0768TS216-D1 and T0768TS345-D1 are shown in Figure 1.6(a) and Figure 1.6(b) respectively. From the superposition figures (as shown in Figure 1.7) we can clearly conclude that T0768TS216-D1 is a near native decoy to the target than T0768TS345-D1.



(a) Decoy structure T0768TS216-D1 with respect to the target protein T0768-D1 of the CASP11 dataset

(b) Decoy structure T0768TS345-D1 with respect to the target protein T0768-D1 of the CASP11 dataset

Figure 1.6: High and low quality decoy structures of the target structure



(a) Image showing the superposition of the decoy structure of T0768TS216-D1 with the target structure T0768-D1

(b) Image showing the superposition of the decoy structure of T0768TS345-D1 with the target structure T0768-D1

Figure 1.7: Images showing the superpositions of the decoy structures with respect to their target structures

In the previous example, it was possible to identify the near-native decoy by comparing it with the available target structure. However, in quality assessment problems, where the target

structure is not available, decisions must be based solely on the decoy structures, which increases the difficulty of the quality assessment stage.

Chapter 2

Prior work in quality assessment

Protein quality assessment methods are evaluated in Critical Assessment of Structure Prediction (CASP) [23]. CASP is a community wide experiment to determine the state of the art techniques in modelling protein structure from amino acid sequence. CASP takes place every two years and the participants submit models for a set of proteins for which the experimental structures have not yet been made public. So CASP tests are aimed at determining the present state of the art in protein structure prediction, indicating where progress has been achieved, and showing where future effort should be spent most effectively. There have been fifteen CASP events and the quality assessment part began from CASP7.

Current techniques for quality assessment can be divided into two categories, one is the single model approach [24] that operates on single protein models to estimate their quality and the other one is the consensus approach [25] that uses consistency among several candidates to estimate quality. For the first time, in the recent CASP13, single model methods performed comparably or better than consensus methods [26]. In this thesis we will concentrate only on single model methods.

A variety of single model algorithms are available. One option is to employ various knowledge-based statistical methodologies [27], such as inter-residue contact energy or atom-atom distances, to find attributes from protein structures to evaluate whether it is a good quality decoy or a bad quality decoy. Another approach is to use machine learning algorithms [28], like SVM or decision trees, where the type of feature from each protein is the most important component, or to use neural networks, such as Convolutional Neural Networks and Graph Neural Networks, which automatically update features based on a defined measure. Here we will focus on the methods that use machine learning algorithms.

2.1 Methods involving machine learning

While knowledge-based potentials have been widely used to model molecular properties, they rely on assumptions and empirical data, and may not be transferable to new systems. Machine learning based approaches have emerged as a promising alternative, as they can be trained on large sets of data without relying on assumptions [29]. Until a few years ago methods that use standard machine learning techniques with a large collection of engineered features computed from sequence and structure were the prevalent approach for quality assessment. The ProQ series of methods (ProQ, ProQ2, ProQ3, ProQ3D) [30] used features such as the distribution of atom-atom contacts, residue-residue contacts, solvent accessibility, secondary structure, surface area, and evolutionary information. ProQ3 [31] also incorporated features based on Rosetta energies. ProQ3D [30] used the descriptors of ProQ3 as input in conjunction with a multilayer perceptron and was one of the top performers of CASP13. The current state-of-the-art for quality assessment uses deep learning, including various flavors of 3D convolutional networks and graph neural networks, which have been demonstrated to be effective modeling tools for protein 3D structures [32, 33]. Deep convolutional networks as a tool for the representation of decoy structures were introduced by [32]. Their method, 3DCNN, used 3D convolutional networks applied to a volumetric representation of a decoy structure. The Ornate method [34] improved upon 3DCNN by defining a canonical orientation for each residue. The GraphQA method [1] employed a graph convolutional network with an extensive number of engineered features and achieved state-of-the-art performance on CASP13 decoys. Chen et al [35] used a graph neural network to estimate the accuracy of AlphaFold models and is one of the current state-of-the-art methods, improving on the results obtained with DeepAccNet [36], while borrowing many ideas from its architecture. They used a combination of categorical loss and L2-loss on the lDDT scores to distinguish between decoys of varying quality levels. The DeepUMQA [2] method uses 3D convolution over a collection of residue-level engineered features, and its successor, DeepUMQA2 [3] is also a state-of-the-art performer.

Most existing methods for quality assessment rely on engineered features. In contrast, our approach uses sequence embeddings computed using protein language models; convolutional layers applied to both atomic and residue level graphs are then used to put them in the context of the decoy structure. In combination with a novel loss function specifically designed for the quality assessment problem, our method is able to outperform the recent DeepUMQA method [2].

Chapter 3

Graph neural networks

In the data domain, while a significant portion of data is vector data readily transformed into grid-like formats, like images, text, or audio, this representation is not always ideal. For irregular domains such as biological networks, knowledge graphs, or social networks, a grid layout falls short [37]. Instead, graph data structures become vital. Graph data structures capture both the information in objects and their relationships, using nodes and edges. Graph neural networks (GNN) use neural networks to handle graph data.

Protein structure remains same regardless of rotations or translations. So computational solutions should have rotational and translational invariance. While 3D Convolutional Neural Networks (3D CNNs) can employ 3D density maps to understand protein shapes, they lack rotational invariance [38]. To achieve the desired invariance, representing proteins as graphs becomes a promising approach. In such scenarios, we can represent the protein as a graph and use Graph Convolutional Networks (GCNs) which is a form of GNN.

3.1 Proteins as graphs

We used graphs to represent decoy 3D structures because they provide an abstract representation of the structure. The atoms and residues of the structure are represented as a set of nodes $V = \{\mathrm{v}_1, \mathrm{v}_2, ..., \mathrm{v}_N, \mathrm{r}_1, \mathrm{r}_2, ..., \mathrm{r}_M\}$, and a set of edges $\mathcal{E} = \mathrm{e}_1, \mathrm{e}_2, ..., \mathrm{e}_P$, where N and M represent the total number of atoms and residues in a decoy structure respectively and P represents the total number of edges present in the graph structure. Each node in the protein's atom-level representation (v_i, $N \geq i \geq 1$) stands in for one of the atoms and if the distance between two atoms is smaller than a certain threshold (in this case, 6Å), an edge will exist between the two nodes. Similarly, an edge occurs between two nodes in the representation of the residue level (r_i, $M \geq i \geq 1$) if the distance between any two of its atoms is less than a predetermined threshold (here, it is taken to be 6Å). For each of the atoms and residues we have considered a maximum of 20 neighbours.

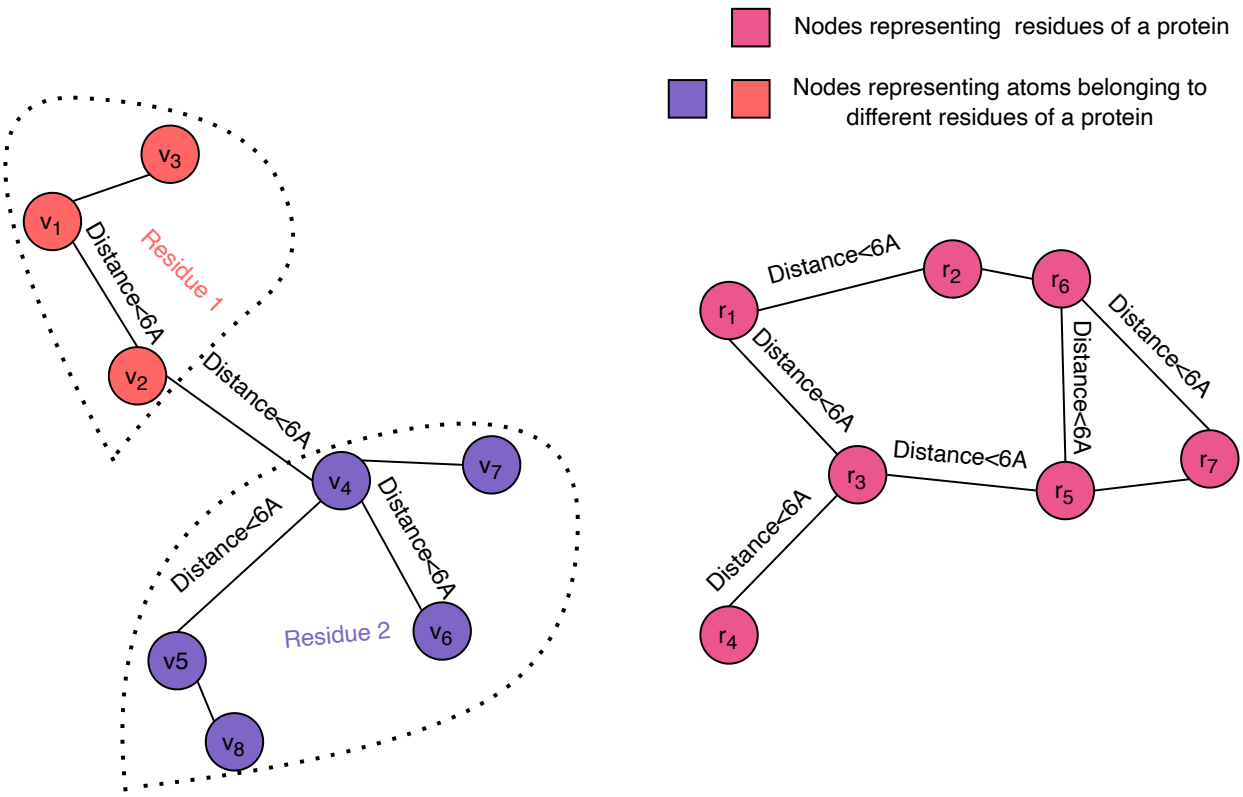


Figure 3.1: Graph representation of a decoy structure. We represent the structure of a decoy using two graphs: one at the atomic level (left), and one at the residue level (right). Our graph convolution operation at the atom level differentiates between edges within a residue and edges across neighboring residues as shown in the figure.

Additionally, the properties of the graph's nodes are coordinate independent. As a result, the graph representation of a protein is rotation and translation invariant.

3.2 Graph neural networks for protein structures

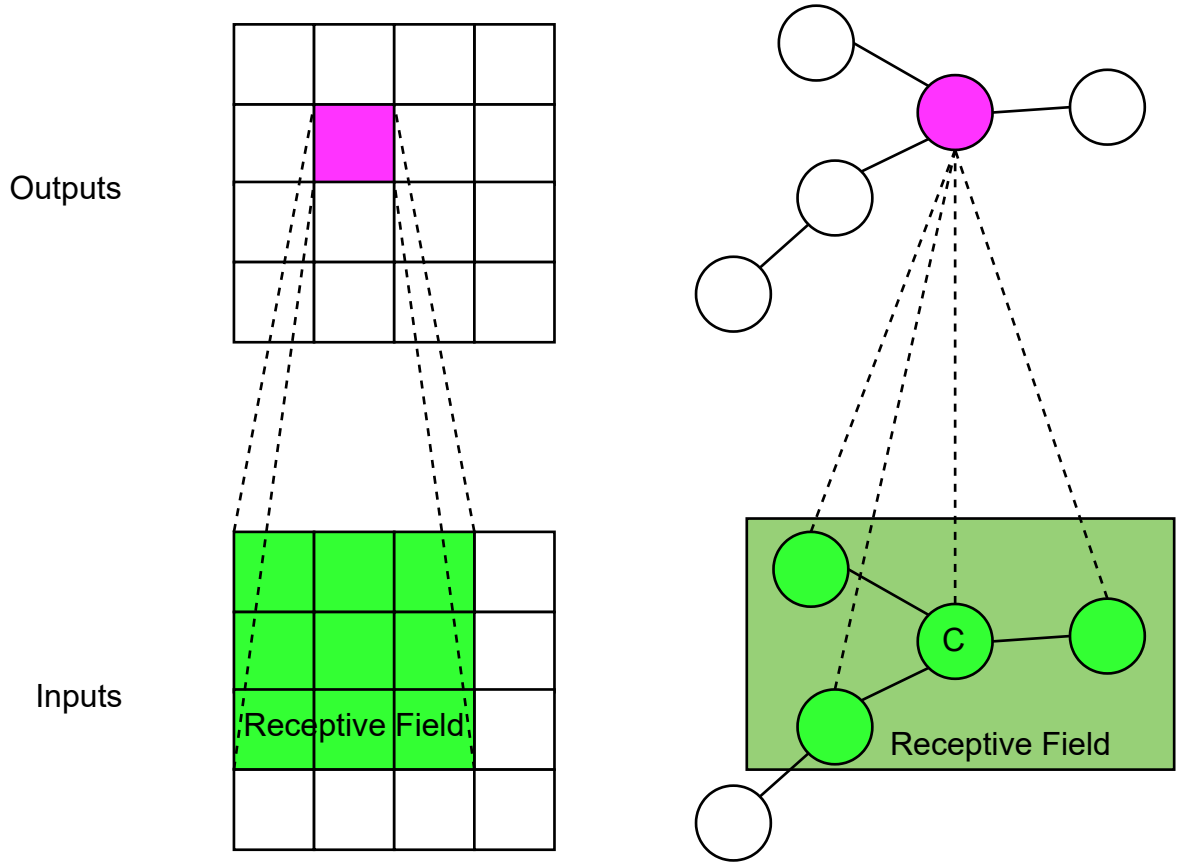


Figure 3.2: Convolution operation comparison between the Convolution Neural Network (CNN) on the left and the GCN on the right. The CNN collects all the hidden representations in a green receptive field to produce the final representation in pink color. Similarly, the GCN aggregates the hidden representations of neighbouring nodes surrounding a central node, C, to produce the center node's final representation, which is highlighted in pink.

In this thesis, we tackle a learning challenge on a graph, focusing on predictions at the protein level. In traditional Convolutional Neural Networks (CNNs), the convolution operation assumes a regular, structured grid, such as an image where pixels have a natural order based on their spatial positions. This allows CNNs to consistently recognize patterns such as edges or textures. However,

when dealing with graphs, there is no natural order among neighboring nodes. This is because nodes in a graph can be connected in various ways without any fixed pattern and there is no spatial structure like in an image. Given the absence of a natural order among neighboring nodes, we need a convolution process that is insensitive to order. That's where Graph Convolutional Networks (GCNs) come in. Unlike CNNs, GCNs are designed to handle the irregular structure of graphs as shown in Figure 3.2.

Fout et al. [39] were the first to use GCNs in this field of study, where they combined learned features across protein pairs and used them to classify whether pairs of amino acid residues are part of an interface or not. They represented the protein as a residue level graph. The convolution is performed according to the following equation:

$$v_i^{(l)} = \sigma\left(W_c^{(l)} v_i^{(l-1)} + \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} W_n^{(l)} {v_j}^{(l-1)} + b^{(l)}\right), \tag{3.1}$$

where $W_c^{(l)}$ and $W_n^{(l)}$ denote the weight matrix for the center and neighbouring nodes respectively, $b^{(l)}$ denotes the bias vector, and $v_i^{(l)}$ denotes the i^{th} node embedding in the l^{th} layer. The set of neighbours of the i^{th} node is denoted by $\mathcal{N}(i)$, while the activation function is denoted by σ.
They employed a graph convolutional network with two convolutional layers to capture spatial interactions between residues, and then concatenated the activations from each layer before passing them through a dense layer for final prediction. Their method outperformed the state-of-the-art SVM algorithm at the time.

Baldassarre et al [38] introduced GraphQA, a GCN-based model adept at assessing protein structure quality and demonstrated comparable performance to state-of-the-art methods on the CASP13 dataset. This was the first instance of GCNs being employed in the field of quality assessment for protein structures. In GraphQA, a protein structure is also represented as a residue graph and they performed convolution as stated in Equation 3.1. The model integrates both node and edge features, as well as a global bias term that accounts for missing information on specific

nodes or edges of the graph, which gets updated during training, while also using a loss function that combines both local and global scores.

Chen et al [35] used structural features generated from the state of the art AlphaFold method along with feature engineering techniques to beat AlphaFold on their datasets. They represented the protein as a residue level graph and employed a novel Graph Neural Network (GNN) consisting of two modules: one at the node level and the other at the edge level. They also used a combination of two loss functions- categortical cross entropy for their initial distance error as well as mean square error with respect to the lDDT scores.

Our approach is almost similar to that of GraphQA. However, we've incorporated an additional graph convolution at the atomic level to provide a more fine grained information about the decoy structure and have adopted a modified ϵ-insensitive loss function. Notably, our method forgoes extensive feature engineering, relying solely on features derived from the protein structure. Its performance is also comparable with state-of-the-art methods while outperforming GraphQA.

Chapter 4

Methods

4.1 Our proposed graph architecture

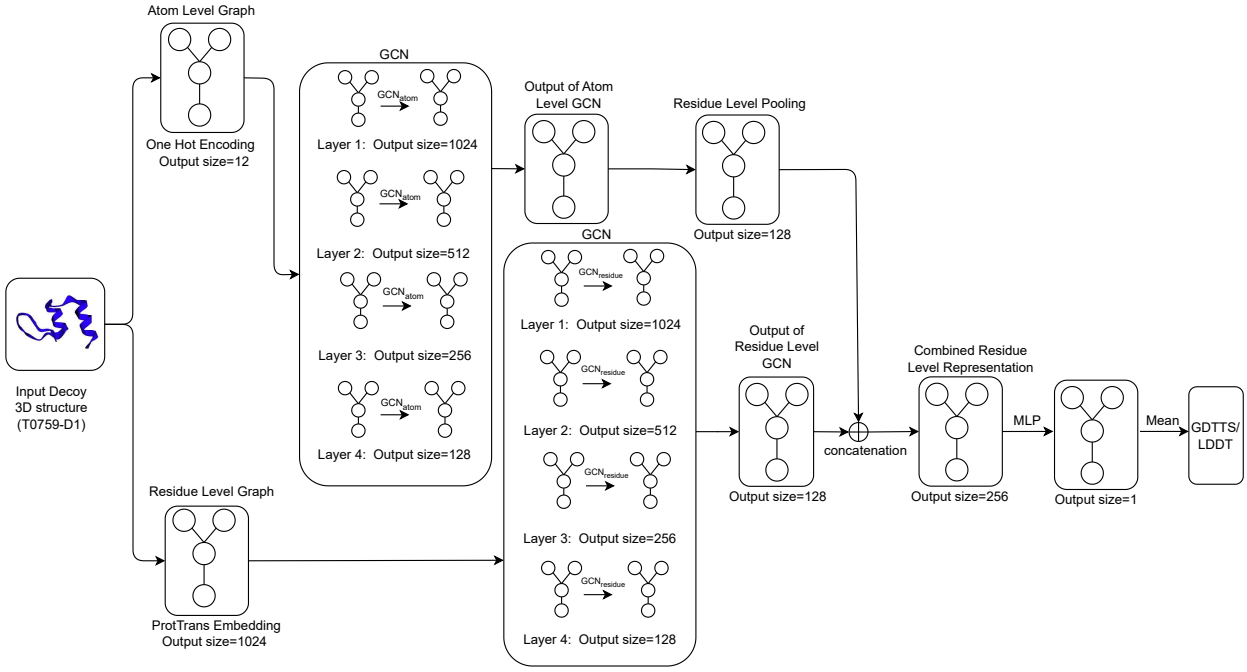


Figure 4.1: The Q_ϵ model architecture illustrating how an input decoy structure is propagated through multiple graph convolutional layers (GCN_{atom} for the atom level representation and $GCN_{residue}$ for the residue level representation of a protein); the outputs of the two sets of convolutional layers are concatenated and fed through a multi-layer perceptron (MLP) to generate local scores that are then averaged to compute the predicted GDTTS or predicted LDDT.

The architecture for Q_ϵ includes four graph convolutional layers that aggregate information at the atomic level (GCN_{atom}) and four graph convolutional layers that pass information at the residue-level ($GCN_{residue}$). To ensure model stability and generalization, we apply batch normalization [40] after each application of an activation function to normalize the activations across the nodes in the graph. To create a coherent representation at the residue level, we apply a maximum-pooling operation to the output of the final layer of GCN_{atom}. The final residue-level representation is

obtained by concatenating the output of the pooled atomic level convolution and the output from the residue-level GCN. This concatenated output is passed through a multi-layer perceptron (MLP) which outputs a single output per residue of the decoy structure. The final output of the network, which is our predicted value of the GDTTS or lDDT is then produced by averaging over the node-level scores. This process is shown in Figure 4.1.

4.2 Atom- and residue-level graph convolution

We perform graph convolution separately at the atom and residue levels. First we describe the atom level graph convolution($\mathrm{GCN}_{\mathrm{atom}}$). Each atom i is assigned a feature vector $\mathrm{v}_i^{(l)}$ that contains the features for layer l of graph convolution. The representation of a source atom $\mathrm{v}_i^{(l)}$ is updated based on its neighbors within the same residue ($\mathcal{N}^{(s)}(i)$) and the neighbors across residues ($\mathcal{N}^{(o)}(i)$) according to:

$$\mathrm{v}_i^{(l+1)} = \mathrm{ReLU}\left(W_l^{(c)}\mathrm{v}_i^{(l)} + \frac{1}{|\mathcal{N}^{(s)}(i)|}W_l^{(s)}\sum_{j\in\mathcal{N}^{(s)}(i)}\mathrm{v}_j^{(l)} + \frac{1}{|\mathcal{N}^{(o)}(i)|}W_l^{(o)}\sum_{j\in\mathcal{N}^{(o)}(i)}\mathrm{v}_j^{(l)} + b_v^{(l)}\right), \tag{4.1}$$

where $W_l^{(c)}$ is the weight matrix with respect to the source atom in layer l, $W_l^{(s)}$ is the weight matrix with respect to the neighbouring atoms in layer l within same residue as that of the source atom, and $W_l^{(o)}$ is the weight matrix with respect to the neighbouring atoms in layer l that belong to a different residue than the source atom; finally, $b_v^{(l)}$ is the bias in layer l for the atom level GCN. The inputs to the atom-level convolution are derived from one-hot-encoding of the atom type as described below.

In parallel to the atom-level convolution, we perform convolution over the residues that make up a decoy structure. This operation, denoted as $\mathrm{GCN}_{\mathrm{residue}}$ is used to update the residue level representation $\mathrm{r}_i^{(l)}$, which is the feature vector for residue i in layer l of the network. This operation is defined as follows:

$$\mathrm{r}_i^{(l+1)} = \mathrm{ReLU}\left(W_l^{(cr)}\mathrm{r}_i^{(l)} + \frac{1}{|\mathcal{R}(i)|}W_l^{(r)}\sum_{j\in\mathcal{R}(i)}\mathrm{r}_j^{(l)} + b_r^{(l)}\right), \tag{4.2}$$

where $\mathcal{R}(i)$ is the set of the neighbouring residues of residue i, $W_l^{(cr)}$ is the weight matrix with respect to the source residue in layer l, $W_l^{(r)}$ is the weight matrix with respect to the neighbouring residues in layer l, and $b_r^{(l)}$ is the bias in layer l. The inputs to the residue-level convolution are embeddings computed using ProtTrans [41] as described below.

4.3 Residue pooling method

Following graph convolution at the atom level, we extract residue-level features from the atom-level graph using a technique called residue pooling. This involves taking the maximum value of each feature from all the atoms within a given residue, resulting in a representation of each residue within a decoy. We then concatenate this residue-level representation with the output from the residue-level graph convolution, and pass the concatenated representation through a multi-layer perceptron (MLP). Finally, we take the mean of all the residues to obtain a global representation of the decoy. This approach allows us to capture both local and global scores of the protein.

4.4 A modified ϵ-insensitive loss

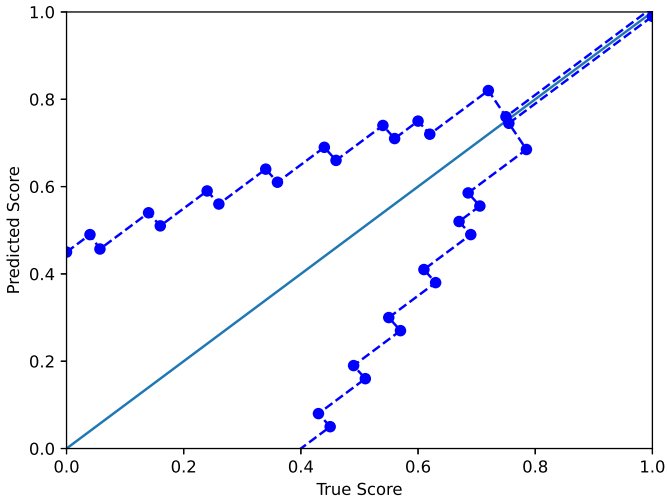


Figure 4.2: The modified ϵ-insensitive loss uses a variable sized band around the diagonal in which a predicted score is not penalized. The band becomes smaller as the GDTTS or lDDT increases, reflecting our wish for precise predictions for decoys that are closer to the native structure.

In this work we address quality assessment as a regression problem with the objective of predicting the GDTTS or lDDT score of a decoy. We propose a novel loss function that captures our desiderata for a quality assessment model: when it comes to poor decoys we don't care too much about the accuracy of the prediction, as long as we can differentiate it from a good decoy. On the other hand, the more accurate the decoy, the more accurate we want our prediction to be. This is especially important given the recent improvement in the quality of protein structure prediction methods. To achieve this goal, we modify the ϵ-insensitive loss, which is the loss function employed in SVM-regression [42] as follows:

$$L(y, y') = max(0, |y - y'| - \epsilon(y)), \tag{4.3}$$

where y and y' are the true and predicted score, respectively. As in the standard ϵ insensitive loss, this defines a tube of size ϵ within which there is no penalty; outside the tube, the loss grows linearly as in the L1 loss which is defined as $L(y, y') = |y - y'|$. In our application, the size of the tube is a function $\epsilon(y)$. In this work we used a tube defined as shown in Figure 4.2. The motivation for the modified ϵ-insensitive loss function is that the model shouldn't try too hard to fit accurately poor quality decoys where we don't need good accuracy anyhow. As decoy quality increases, we are asking the model to learn a fit that is more and more accurate.

Chapter 5

Input features and scores

This section explains about all the input features and metrics that are being used in our graph neural network.

5.1 Atom features

Table 5.1: Table showing the different types of atoms.

Type	Atom Type	Atom Name
1	Sulfur	CYS:SG, MET:SD
2	Aromatic carbon	HIS:CD2, HIS:CE1, HIS:CG, PHE:CD1, PHE:CD2, PHE:CE1, PHE:CE2, PHE:CG, PHE:CZ, TRP:CD1, TRP:CD2, TRP:CE2, TRP:CE3, TRP:CG, TRP:CH2, TRP:CZ2, TRP:CZ3, TYR:CD1, TYR:CD2, TYR:CE1, TYR:CE2, TYR:CG, TYR:CZ
3	Sp2 carbon	ARG:CZ, ASN:CG, ASP:CG, GLN:CD, GLU:CD, backbone C
4	Sp3 carbon	ALA:CB, ARG:CB, ARG:CG, ASN:CB, ASP:CB, GLN:CB, GLN:CG, GLU:CB, GLU:CG, HIS:CB, ILE:CB,ILE:CD1,ILE:CG1,ILE:CG2,LEU:CB, LEU:CD1, LEU:CD2, LEU:CG, LYS:CB, LYS:CD, LYS:CG, MET:CB, PHE:CB, PRO:CB, PRO:CG, SER:CB, THR:CG2, TRP:CB, TYR:CB, VAL:CB, VAL:CG1, VAL:CG2
5	Alpha Carbon	CA
6	Nitrogen of amide group	N (on terminal residues), ASN:ND2, GLN:NE2
7	Nitrogen of guanidinium group	ARG:NH1, ARG:NH2 ,ARG:NE
8	Nitrogen of ammonium group	LYS:NZ
9	Oxygen of hydroxyl group	SER:OG, THR:OG1, TYR:OH
10	Oxygen of carbonyl group	ASN:OD1, GLN:OE1
11	Oxygen of carboxyl group	ASP:OD1, ASP:OD2, GLU:OE1, GLU:OE2

We represent the atoms using one-hot encoding, by grouping atoms into 11 different types [32]. This grouping reflects both the type of atom (carbon, oxygen, nitrogen) and its context within the residue (e.g. alpha carbon, or the different group an atom belongs to). In doing so, we are able to incorporate important information about the atoms while also capturing the relationships between the atoms and their corresponding residues.

5.2 Residue features

We compute residue features by feeding the amino acid sequence of a decoy to the ProtTrans protein language model [41]. ProtTrans embeddings provide a very useful representation of the amino acid sequence, capturing relationships between residues as well as their structural context [41]. We take the embeddings from the last hidden state of the transformer attention stack of the ProtTrans model, which has an output embedding of 1024 dimensions, which serves as input to the residue level GCN.

5.3 Metrics

To evaluate the quality of a decoy, we use scoring metrics that measure how well it aligns with the target structure. In this thesis, we focus on two such metrics: the Global Distance Test - Total Score (GDT-TS) and the Local Distance-Dependent Information-based (LDDT) score.

The GDT-TS score was introduced in CASP4 as a measure of the similarity between the decoy model and the target structure. It is computed by superposing the Cα atoms of the decoy model onto those of the target and calculating the percentage of residues that fall within certain distance cutoffs (1, 2, 4, and 8 Å) between the two structures. The GDT-TS score is the average of the percentage of residues falling within these four distance cutoffs:

$\mathrm{GDT-TS} = \frac{1}{4}(\mathrm{GDT\_P_1} + \mathrm{GDT\_P_2} + \mathrm{GDT\_P_4} + \mathrm{GDT\_P_8})$

where $\mathrm{GDT\_P_d}$ represents percent of residues under distance cutoff $\leq$ d Å

The LDDT score measures the accuracy of the decoy at a local level by computing the difference between the distances of corresponding atom pairs in the decoy and target structures. It

is calculated for four different distance cutoffs (0.5, 1, 2, and 4 Å) and measures the fraction of accurate contacts preserved in the decoy. An accurate contact is defined as a pair of atoms that are in contact in the target structure and whose corresponding atoms in the decoy are within a certain distance cutoff of the target atoms.

In summary, we use the GDT-TS and LDDT scores to evaluate the quality of decoys based on their overall similarity to the target structure and their accuracy at a local level, respectively. These metrics have been widely used in the field of protein structure prediction and have been shown to be effective at discriminating between high-quality and low-quality decoy models.

Chapter 6

Data and training

6.1 Data

Table 6.1: The number of targets from CASP competitions in the training, validation, and testing data. We have two different CASP13 and CASP14 datasets, one for GDTTS evaluation and the other for lDDT evaluation, to match decoys used in other publications.

Mode	CASP	Targets	Decoys
Training Data	CASP9	117	31,863
	CASP10	100	23,755
	CASP11	84	15,573
	CASP12	30	5351
Validation Data	CASP12	10	1338
Testing Data (GDTTS)	CASP13	72	34,654
	CASP14	65	38,293
Testing Data (lDDT)	CASP13	76	10,739
	CASP14	70	10,380
	AlphaFold2 CASP15	17	85

We collected decoys from CASP9 to CASP14 along with their labels from the CASP website [43]. We have used CASP9-CASP12 as our training and validation set and CASP13 and CASP14 as our test sets (see Table 6.1). In order to match the decoys used in experiments performed by others we created two separate datasets for GDDTS evaluation (CASP13 and CASP14) and two datasets for evaluation of lDDT prediction (CASP13 and CASP14).

In CASP15, the focus shifted from predicting the accuracy of single-chain decoys to that of multi- chain complexes [44]. However, some of the targets were composed of single chains, and we chose to focus on those targets in our evaluation, leading to a dataset with 17 targets.

6.1.1 Network training

We have trained our network to predict GDTTS as well as lDDT scores. For GDTTS prediction, we first pre-train Q_ϵ with the L1-loss for 50 epochs followed by training with the modified ϵ-insensitive loss for the next 10 epochs. To train the network with lDDT scores, we take the best model from GDTTS ("best" with respect to the validation set) train it with the modified ϵ-insensitive loss for another 50 epochs keeping the same network architecture and hyperparameters.

The network was implemented in PyTorch [45], and optimized using the Adam method [46] with default parameters except for a learning rate of 0.001 and a dropout of 0.1 applied to the graph convolution layers; training used a batch size of 70. Since our training set is highly imbalanced, i.e. contains very few high quality decoys, we used the imbalanced sampler from the torchsampler package. During training we kept track of the loss over the validation set and used the model that gave the minimum loss. Our implementation uses the PyTorch Lightning framework for training and testing and PyTorch Geometric [47] for performing graph convolution. Model selection was performed over the hyperparameters and values described in Table 6.2. We iterated over all parameters and for each one chose the value that gave the highest Pearson correlation coefficient on the validation set. Following model selection, training took around 42 hours on an NVIDIA RTX 3090 GPU.

Table 6.2: The hyperparameter space. Model selection was performed based on performance on the validation set. The "Best" column provides the chosen value for each hyperparameter.

Hyperparameter	Values	Best
Number of graph convolution layers	2,3,4,5,6	4
Neighbour distance threshold	4,5,6,7,8,9	6
Maximum number of same residue atom neighbours	10,15,20,25	20
Maximum number of different residue atom neighbours	10,15,20,25	20
Maximum number of neighbours of a residue	10,15,20,25	20
Dropout rate for the graph convolution layers	0,0.1,0.2,0.3	0.1
Learning rate	0.0001,0.001,0.01,0.1	0.001

Chapter 7

Results

We compare Q_ϵ with other methods that have either state-of-the-art or very good performance in CASP13 and CASP14. In our first set of experiments we sought to compare our method with GraphQA, which uses a similar graph convolution architecture and was trained to predict GDTTS [1]. The results in Table 7.1 clearly indicate that Q_ϵ outperforms GraphQA and several other recent methods trained to predict GDTTS. This is despite not using engineered features; a detailed analysis of the contribution of the various components of the Q_ϵ architecture are described in an ablation study below.

The quality assessment community is transitioning to the use of the lDDT score, so we also compare Q_ϵ with more recent methods evaluated with lDDT. In this evaluation, Q_ϵ performed similarly to DeepUMQA, but was outperformed by its successor, DeepUMQA2 (see Table 7.2). Results from EnQA [35], which performed similarly to DeepUMQA2 are also better than Q_ϵ. Both methods use more complex architectures and extensive engineered features; DeepUMQA2 also used evolutionary information, including structural features from homologous templates.

Table 7.1: Performance of Q_ϵ and other methods in CASP13 and CASP14 GDTTS prediction. We provide the Pearson Correlation Coefficient globally (R) and per target (R_{target}) as well as Spearman Rank Correlation between predicted and known GDTTS. Best performance is highlighted in bold. Performance numbers for the other methods is quoted from [1].

Dataset	Method	R	R_{target}	ρ	RMSE
CASP13	Q_ϵ	**0.90**	**0.80**	**0.89**	**0.10**
	GraphQA [1]	0.86	0.78	0.86	0.13
	ModFOLD7_rank [48]	0.87	0.74	-	0.16
	ProQ4 [49]	0.70	0.66	-	0.18
	VoroMQA-A [50]	0.66	0.56	-	0.21
CASP14	Q_ϵ	**0.81**	**0.72**	**0.82**	**0.13**

Table 7.2: Performance of Q_ϵ and other methods in CASP13 and CASP14 with respect to lDDT scores. We provide the Pearson Correlation Coefficient globally (R) as well as Spearman Rank Correlation between predicted and known lDDT scores. Best performance is highlighted in bold. Performance figures for methods other than Q_ϵ are quoted from [2] and [3].

Dataset	Method	R	ρ
CASP13	Q_ϵ	0.857	**0.862**
	DeepUMQA2 [3]	**0.919**	-
	DeepUMQA [51]	0.837	0.804
	ModFOLD7_rank [52]	0.826	-
	ProQ3D [30]	0.801	-
	ProQ4 [30]	0.777	-
	ProQ2 [53]	0.715	-
	VoroMQA-A [50]	0.672	-
CASP14	Q_ϵ	0.826	**0.826**
	DeepUMQA2 [3]	**0.899**	-
	DeepUMQA [51]	0.799	0.736
	DeepAccNet [36]	0.829	-
	ModFold8 [54]	0.629	-
	GraphQA [1]	0.706	-
	ProQ3D [30]	0.717	-
	ProQ2 [53]	0.531	-
	ProQ4 [30]	0.547	-

To understand the contribution of the proposed modified ϵ-insensitive loss to the performance of Q_ϵ we show a scatter plot of true versus predicted GDTTS scores for the decoys in CASP13 and CASP14 (see Figure 7.1). We observe that the modified ϵ-insensitive loss leads to better learning of decoys of all quality levels compared to the L1-loss, and leads a pattern where the predictions are limited to a band around the true scores, which is a highly desirable property for a quality

assessment method. It was interesting that the width of the band is similar across all quality levels, despite the loss having a variable width band compared to the original ϵ-insensitive loss function.

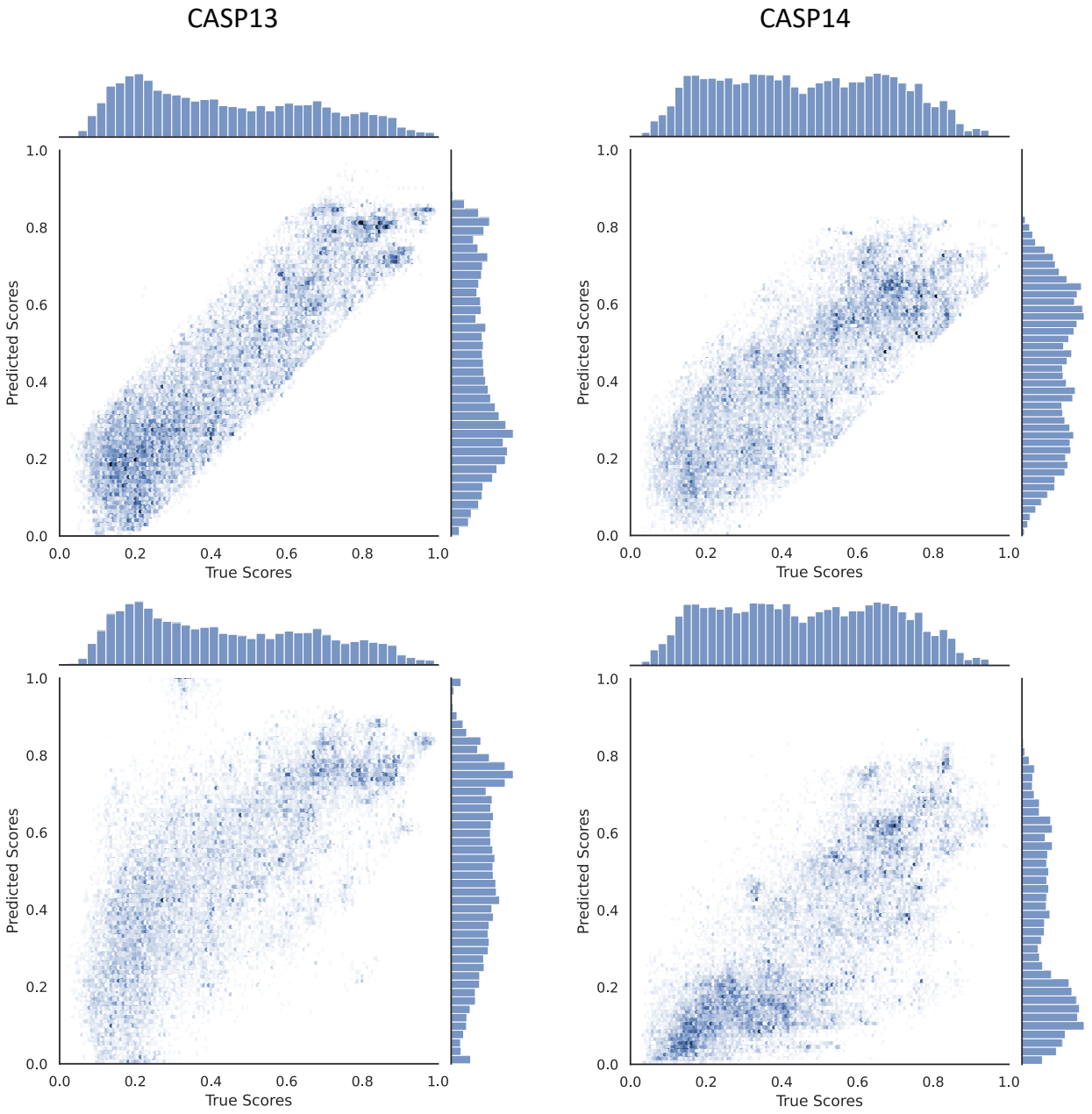


Figure 7.1: Scatter plots comparing the true and predicted GDTTS for both CASP13 and CASP14 using L1-Loss and modified ϵ-insensitive loss.

Table 7.3: Q_ϵ ablation study. We remove each of the major elements of Q_ϵ, demonstrating that each of them provides a major contribution to the performance of the method.

Method	R	R_{target}	ρ	RMSE
Q_ϵ (with atom and residue features, pretrained with L1-loss, and modified ϵ-insensitive loss)	**0.90**	**0.80**	**0.89**	**0.11**
Q_ϵ without modified ϵ-insensitive loss	0.75	0.66	0.69	0.17
Q_ϵ without L1-Loss	0.70	0.59	0.62	0.20
Q_ϵ with a constant ϵ (0.2)	0.63	0.55	0.66	0.24
Q_ϵ with only L2-Loss	0.65	0.52	0.56	0.23
Q_ϵ without residue features	0.70	0.65	0.69	0.19
Q_ϵ without atom features	0.79	0.77	0.76	0.18

7.0.1 Ablation Study

To demonstrate the contribution of each of the major components of our method we performed an ablation study with respect to GDTTS prediction. Its results are provided in Table 7.3. The first component we varied was the loss function. We observe that the pre-training with the L1 loss is key for the method's performance, serving to bootstrap the learning process. We also observe that performance dropped when using the original ϵ-insensitive loss function, L1 loss, or the L2 loss. This clearly shows the contribution of the proposed modification to the ϵ-insensitive loss. Our next observation is that both the residue-level and the atom-level convolutional blocks are crucial for the performance of the method. This is due to each of them providing different and complementary information. The residue-level blocks use Prot-Trans embeddings which have been documented to provide a variety of information regarding a residue's evolutionary history and structural context within the protein [41]. The atom-level convolutional blocks provide a more fine-grained view of a decoy structure, complementing the information at the residue level.

7.0.2 ϵ threshold selection

The modified ϵ-insensitive loss has nine threshold parameters associated with it, one for each bin of the prediction score. In our experiments we have used the values shown in Figure 4.2. In order to determine that our initial choice was good, we ran an experiment where we varied all the values in a coordinated manner: we chose values nine values lower or higher than the initial values (the columns low and high in Table 7.4). As shown in Table 7.4, lowering or increasing the values in a coordinated fashion of all the thresholds led to reduced accuracy on the validation set. As a sanity check, we verified that a similar decrease is observed on the test set as well.

Table 7.4: Model selection over the ϵ hyperparameter values. The top half of shows the values of ϵ for each score range. The lower half shows the performance for each combination of values (low/mid/high); R stands for the Pearson Correlation Coefficient. Results are shown for the validation set (first two rows) and the test set for both GDTTS and lDDT

Score Ranges and Results	Low values	Mid values	High values
ϵ for 0-0.1	0.40	0.45	0.50
ϵ for 0.1-0.2	0.35	0.40	0.45
ϵ for 0.2-0.3	0.30	0.35	0.40
ϵ for 0.3-0.4	0.25	0.30	0.35
ϵ for 0.4-0.5	0.20	0.25	0.30
ϵ for 0.5-0.6	0.15	0.2	0.25
ϵ for 0.6-0.7	0.10	0.15	0.20
ϵ for 0.7-0.8	0.05	0.1	0.15
ϵ for > 0.8	0.005	0.01	0.015
R on CASP12 (validation set) (GDTTS)	0.84	0.89	0.82
R on CASP12 (validation set) (lDDT)	0.81	0.84	0.77
R on CASP13 (test set) (GDTTS)	0.86	0.90	0.85
R on CASP14 (test set) (GDTTS)	0.80	0.81	0.79
R on CASP13 (test set) (lDDT)	0.84	0.86	0.83
R on CASP14 (test set) (lDDT)	0.82	0.83	0.80

7.0.3 Local quality assessment with Q_ϵ

In this section we demonstrate the ability of Q_ϵ to make accurate predictions at the residue level despite being trained only on global quality scores. This ability is a byproduct of the architecture of the network, where the global predicted score is an average of residue-level node summary scores (see Figure 4.1). This forces the network to learn accurate local scores, as demonstrated in the results shown in Table 7.5. Similar to the global prediction problem, the performance of Q_ϵ is between that of DeepUMQA and DeepUMQA2.

Table 7.5: Performance of Q_ϵ and other methods in CASP13 and CASP14 with respect to local lDDT scores. We provide the local Pearson Correlation Coefficient (R_{local}) between predicted and known local lDDT scores. Best performance is highlighted in bold. Performance figures for methods other than Q_ϵ are quoted from [4].

Dataset	Method	R_{local}
CASP13	Q_ϵ	0.80
	DeepUMQA2 [4]	**0.868**
	DeepUMQA [51]	0.766
	DeepAccNet [36]	0.740
CASP14	Q_ϵ	0.76
	DeepUMQA2 [4]	**0.822**
	DeepUMQA [51]	0.680
	DeepAccNet [36]	0.672

7.0.4 Results on CAMEO decoys

For further validation of the performance of Q_ϵ, we evaluated its performance on decoys from the CAMEO evaluation project [55]. We downloaded decoys used from 2022-05-13 to 2023-05-06 and followed the same evaluation protocol used by CAMEO: we calculated the Area Under the ROC curve (AUROC) and Area Under the Precision Recall curve (AUPR) using a local lDDT score threshold of 0.6, and obtained the results shown in Table 7.6. Again, we note that Q_ϵ was

not trained on local scores (unlike the other methods), and yet is able to perform almost on par with DeepUMQA2. As mentioned above, this can be traced to the fact that the global prediction score computed by Q_ϵ is computed by directly averaging local node summary scores, forcing those scores to reflect a local measure of quality.

Table 7.6: Performance of Q_ϵ and other methods on the CAMEO dataset. Best performance is highlighted in bold. All the other results have been taken from the CAMEO website.

Dataset	Method	Models	AUROC	AUPR
CAMEO-QA	Q_ϵ	6350	0.93	0.88
	DeepUMQA2 [3]	6225	**0.94**	**0.89**
	Proq3D_LDDT [30]	6498	0.90	0.81
	DeepUMQA [51]	6247	0.93	0.86
	Modfold9 [56]	6498	0.92	0.87

7.0.5 Performance on AlphaFold2 decoys

In CASP14 AlphaFold2 provided for the first time decoys with near experimental resolution [57], producing decoys with a median GDTTS of 92.4 making it the first team to achieve this level of accuracy in CASP. We gathered the decoys submitted by the AlphaFold team (team no 427) from the CASP14 website and evaluated Q_ϵ on their decoys. We also ran AlphaFold2 version 2.3.1 on CASP15 single chain targets. The results of this experiment are provided in Table 7.7. While AlphaFold2 provided better accuracy than our method, its value is in providing independent validation for the quality of AlphaFold2 predictions. While EnQA [35] slightly improves on the quality AlphaFold2 lDDT estimates, it does so by using the AlphaFold2 scores as one of its features. Therefore, we expect this method's results to be highly correlated to the results of AlphaFold2, and are less useful for independent verification of its predictions.

Table 7.7: Performance of Q_ϵ and AlphaFold2 on AlphaFold2-generated decoys in CASP14 and CASP15. We provide the global Pearson correlation (R), local Pearson correlation (R_{local}) and Spearman rank correlation (ρ) between predicted and known local and global lDDT scores. Best performance is highlighted in bold.

Dataset	Method	R	R_{local}	ρ
AlphaFold2-CASP14	Q_ϵ	0.772	0.730	0.832
	AlphaFold2	**0.85**	**0.792**	**0.882**
AlphaFold2-CASP15	Q_ϵ	0.64	0.60	0.60
	AlphaFold2	**0.75**	**0.72**	**0.67**

7.1 Conclusions and Future Work

In this study, we proposed a novel loss function to enhance the performance of deep learning for quality assessment of decoy structures. Our approach has performance that is close to the state-of-the-art while at the same time removing the need for engineered features computed from the protein structure, relying solely on features computed from the decoy sequence, demonstrating what is possible with a pure deep learning approach. These features are integrated using graph convolutional layers that operate at both the atom and residue levels, thereby improving the network's performance. The comparison of our approach with AlphaFold2 indicates there is a need for further research to provide accuracy estimates that improve on the local scores computed by AlphaFold2 in order to provide independent validation of the quality of its predicted structures.

Our approach can be extended in multiple ways. First, although it performs well on predicting local scores, the method is trained using only global quality scores. Joint learning of both global and local scores can potentially improve performance for both tasks. Second, we treated prediction of GDDTS and lDDT as independent tasks; there is a potential gain in addressing multiple quality scores at the same time [1]. Finally, in this work we chose to focus on the contribution of the loss function to method performance, so we used a relatively simple graph convolutional network similar to the one used in GraphQA [1]. Finally, we expect that the proposed loss function can be applied to regression problems whose objective is to detect high quality objects, and has the potential to be a useful addition to any deep learning toolbox.

Bibliography

[1] Federico Baldassarre, David Menéndez Hurtado, Arne Elofsson, and Hossein Azizpour. GraphQA: protein model quality assessment using graph convolutional networks. *Bioinformatics*, 37(3):360–366, 08 2020.

[2] Sai-Sai Guo, Jun Liu, Xiao-Gen Zhou, and Gui-Jun Zhang. DeepUMQA: ultrafast shape recognition-based protein model quality assessment using deep learning. *Bioinformatics*, 38(7):1895–1903, 02 2022.

[3] Jun Liu, Kailong Zhao, and Guijun Zhang. Improved model quality assessment using sequence and structural information by enhanced deep neural networks. *Briefings in bioinformatics*, 24(1):bbac507, 2023.

[4] Jun Liu, Kailong Zhao, and Guijun Zhang. Improved model quality assessment using sequence and structural information by enhanced deep neural networks. *Briefings in Bioinformatics*, 24(1), 12 2022. bbac507.

[5] Nizar N Batada, Laurence D Hurst, and Mike Tyers. Evolutionary and physiological importance of hub proteins. *PLoS computational biology*, 2(7):e88, 2006.

[6] Sali A. Baker D. Protein structure prediction and structural genomics. *Science*, 2001.

[7] B Al-Lazikani, J Jung, Z Xiang, and B Honig. Protein structure prediction. *Current opinion in chemical biology*, 5(1):51—56, February 2001.

[8] Daisuke Kihara, Hao Chen, and Yifeng David Yang. Quality assessment of protein structure models. *Current Protein & Peptide Science*, 10(3):216–228, 2009.

[9] Tatiana Maximova, Ryan Moffatt, Buyong Ma, Ruth Nussinov, and Amarda Shehu. Principles and overview of sampling methods for modeling macromolecular structure and dynamics. *PLoS computational biology*, 12(4):e1004619, 2016.

[10] Amarda Shehu. A review of evolutionary algorithms for computing functional conformations of protein molecules. *Computer-Aided Drug Discovery*, pages 31–64, 2015.

[11] Nasrin Akhter, Gopinath Chennupati, Hristo Djidjev, and Amarda Shehu. Decoy selection for protein structure prediction via extreme gradient boosting and ranking. *BMC bioinformatics*, 21(1):1–21, 2020.

[12] https://en.wikipedia.org/wiki/Central_dogma_of_molecular_biology#/media/File:Central_Dogma_of_Molecular_Biochemistry_with_Enzymes.jpg.

[13] FRANCIS CRICK. Central dogma of molecular biology. *Nature*, 227(3):561–563, 1970.

[14] https://en.wikipedia.org/wiki/Proteinogenic_amino_acid.

[15] https://pymol.org/2/.

[16] https://app.diagrams.net/.

[17] L. Skipper. Proteins | overview. In Paul Worsfold, Alan Townshend, and Colin Poole, editors, *Encyclopedia of Analytical Science (Second Edition)*, pages 344–352. Elsevier, Oxford, second edition edition, 2005.

[18] Brian Kuhlman and David Baker. Native protein sequences are close to optimal for their structures. *Proceedings of the National Academy of Sciences*, 97(19):10383–10388, 2000.

[19] Marcin J. Skwark and Arne Elofsson. PconsD: ultra rapid, accurate model quality assessment for protein structure prediction. *Bioinformatics*, 29(14):1817–1818, 05 2013.

[20] Konstantin Arnold, Lorenza Bordoli, Jürgen Kopp, and Torsten Schwede. The swiss-model workspace: a web-based environment for protein structure homology modelling. *Bioinformatics*, 22(2):195–201, 2006.

[21] Sheng Wang, Siqi Sun, Zhen Li, Renyu Zhang, and Jinbo Xu. Accurate de novo prediction of protein contact map by ultra-deep learning model. *PLoS computational biology*, 13(1):e1005324, 2017.

[22] Jinbo Xu. Distance-based protein folding powered by deep learning. *Proceedings of the National Academy of Sciences*, 116(34):16856–16865, 2019.

[23] John Moult, Jan T Pedersen, Richard Judson, and Krzysztof Fidelis. A large-scale experiment to assess protein structure prediction methods, 1995.

[24] Björn Wallner and Arne Elofsson. Can correct protein models be identified? *Protein science*, 12(5):1073–1086, 2003.

[25] Jesper Lundström, Leszek Rychlewski, Janusz Bujnicki, and Arne Elofsson. Pcons: A neural-network–based consensus predictor that improves fold recognition. *Protein science*, 10(11):2354–2362, 2001.

[26] Jianlin Cheng, Myong-Ho Choe, Arne Elofsson, Kun-Sop Han, Jie Hou, Ali HA Maghrabi, Liam J McGuffin, David Menéndez-Hurtado, Kliment Olechnovič, Torsten Schwede, et al. Estimation of model accuracy in casp13. *Proteins: Structure, Function, and Bioinformatics*, 87(12):1361–1377, 2019.

[27] Manfred J Sippl. Knowledge-based potentials for proteins. *Current Opinion in Structural Biology*, 5(2):229–235, 1995.

[28] Mohammed AlQuraishi. Machine learning in protein structure prediction. *Current opinion in chemical biology*, 65:1–8, 2021.

[29] Nongnuch Artrith, Alexander Urban, and Gerbrand Ceder. Efficient and accurate machine-learning interpolation of atomic energies in compositions with many species. *Phys. Rev. B*, 96:014112, Jul 2017.

[30] 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.

[31] Karolis Uziela, Nanjiang Shu, Björn Wallner, and Arne Elofsson. Proq3: Improved model quality assessments using rosetta energy terms. *Scientific reports*, 6(1):1–10, 2016.

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

[33] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. Protein interface prediction using graph convolutional networks. *Advances in neural information processing systems*, 30, 2017.

[34] Guillaume Pagès, Benoit Charmettant, and Sergei Grudinin. Protein model quality assessment using 3D oriented convolutional neural networks. *Bioinformatics*, 35(18):3313–3319, 02 2019.

[35] Chen Chen, Xiao Chen, Alex Morehead, Tianqi Wu, and Jianlin Cheng. 3D-equivariant graph neural networks for protein model quality assessment. *Bioinformatics*, 39(1), 01 2023.

[36] Naozumi Hiranuma, Hahnbeom Park, Minkyung Baek, Ivan Anishchenko, Justas Dauparas, and David Baker. Improved protein structure refinement guided by deep learning based accuracy estimation. *Nature communications*, 12(1):1340, 2021.

[37] Takuo Hamaguchi, Hidekazu Oiwa, Masashi Shimbo, and Yuji Matsumoto. Knowledge transfer for out-of-knowledge-base entities: A graph neural network approach. *arXiv preprint arXiv:1706.05674*, 2017.

[38] Federico Baldassarre, David Menéndez Hurtado, Arne Elofsson, and Hossein Azizpour. GraphQA: protein model quality assessment using graph convolutional networks. *Bioinformatics*, 37(3):360–366, 08 2020.

[39] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. Protein interface prediction using graph convolutional networks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[40] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[41] Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rehawi, Yu Wang, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Martin Steinegger, et al. Prottrans: Toward understanding the language of life through self-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 44(10):7112–7127, 2021.

[42] Harris Drucker, Christopher J. C. Burges, Linda Kaufman, Alex Smola, and Vladimir Vapnik. Support vector regression machines. In M.C. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9. MIT Press, 1996.

[43] CASP. CASP. https://predictioncenter.org/download_area/, 2021.

[44] Andriy Kryshtafovych, Maciej Antczak, Marta Szachniuk, Tomasz Zok, Rachael C Kretsch, Ramya Rangan, Phillip Pham, Rhiju Das, Xavier Robin, Gabriel Studer, et al. New prediction categories in casp15. *Proteins: Structure, Function, and Bioinformatics*, 2023.

[45] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

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

[47] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with PyTorch Geometric. *arXiv preprint arXiv:1903.02428*, 2019.

[48] Liam J McGuffin, Recep Adiyaman, Ali HA Maghrabi, Ahmad N Shuid, Danielle A Brackenridge, John O Nealon, and Limcy S Philomina. IntFOLD: an integrated web resource

for high performance protein structure and function prediction. *Nucleic acids research*, 47(W1):W408–W413, 2019.

[49] David Menéndez Hurtado, Karolis Uziela, and Arne Elofsson. Deep transfer learning in the assessment of the quality of protein models. *arXiv preprint arXiv:1804.06281*, 2018.

[50] Kliment Olechnovič and Česlovas Venclovas. Voromqa: Assessment of protein structure quality using interatomic contact areas. *Proteins: Structure, Function, and Bioinformatics*, 85(6):1131–1145, 2017.

[51] Sai-Sai Guo, Jun Liu, Xiao-Gen Zhou, and Gui-Jun Zhang. DeepUMQA: ultrafast shape recognition-based protein model quality assessment using deep learning. *Bioinformatics*, 38(7):1895–1903, 2022.

[52] Ali HA Maghrabi and Liam J McGuffin. Estimating the quality of 3d protein models using the modfold7 server. *Protein Structure Prediction*, pages 69–81, 2020.

[53] Arjun Ray, Erik Lindahl, and Björn Wallner. Improved model quality assessment using proq2. *BMC bioinformatics*, 13(1):1–12, 2012.

[54] Liam J McGuffin, Fahd MF Aldowsari, Shuaa MA Alharbi, and Recep Adiyaman. Modfold8: accurate global and local quality estimates for 3d protein models. *Nucleic acids research*, 49(W1):W425–W430, 2021.

[55] Jürgen Haas, Alessandro Barbato, Dario Behringer, Gabriel Studer, Steven Roth, Martino Bertoni, Khaled Mostaguir, Rafal Gumienny, and Torsten Schwede. Continuous Automated Model EvaluatiOn (CAMEO) complementing the critical assessment of structure prediction in CASP12. *Proteins: Structure, Function, and Bioinformatics*, 86:387–398, 2018.

[56] Liam J McGuffin, Nicholas S Edmunds, Ahmet G Genc, Shuaa MA Alharbi, Bajuna R Salehe, and Recep Adiyaman. Prediction of protein structures, functions and interactions using the IntFOLD7, MultiFOLD and ModFOLDdock servers. *Nucleic Acids Research*, page gkad297, 2023.

[57] Jeffrey Skolnick, Mu Gao, Hongyi Zhou, and Suresh Singh. Alphafold 2: why it works and its implications for understanding the relationships of protein sequence, structure, and function. *Journal of chemical information and modeling*, 61(10):4827–4831, 2021.

Appendix A

License

Colorado State University LaTeX Thesis Template

by Elliott Forney – 2017

This is free and unencumbered software released into the public domain.

Anyone is free to copy, modify, publish, use, compile, sell, or distribute this software, either in source code form or as a compiled binary, for any purpose, commercial or non-commercial, and by any means.

In jurisdictions that recognize copyright laws, the author or authors of this software dedicate any and all copyright interest in the software to the public domain. We make this dedication for the benefit of the public at large and to the detriment of our heirs and successors. We intend this dedication to be an overt act of relinquishment in perpetuity of all present and future rights to this software under copyright law.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.