DEEP LEARNING METHODS FOR PROTEIN PREDICTION PROBLEM

A Dissertation Presented to

The Faculty of the Graduate School

At the University of Missouri

In Partial Fulfillment

of the Requirements for the Degree

Ph.D. in Computer Science

By

SON PHONG NGUYEN

Dr. Yi Shang, Advisor

JULY 2017

The undersigned, appointed by the Dean of the Graduate School, have examined the dissertation entitled

DEEP LEARNING METHODS FOR PROTEIN PREDICTION PROBLEM

presented by Son Phong Nguyen

a candidate for the degree of Doctor of Philosophy

and hereby certify that, in their opinion, it is worthy of acceptance.

Dr. Yi Shang

Dr. Jianlin Cheng

Dr. Dong Xu

Dr. Ioan Kosztin

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor Dr. Yi Shang, for his continuous support, guidance, and never-ending patience for my PhD study during the years. His broad knowledge and deep insights help me overcomes many obstacles in the process of doing research. He helps me improve my research methodology, the way of thinking about problems and how to solve it. He is a great advisor not only for my study and research but also on my life and career path.

I would like to give my great thanks to Dr. Dong Xu, Dr. Ioan Kosztin and Dr. Jianlin Cheng for their helpful discussions and collaborative efforts over the years. Their support is essential for me to solve tough problems in research and achieve better results.

I also want to thank all of my colleagues Zhaoyu Li, Junlin Wang, Sean Lander, Jianjiong Gao, Jingfen Zhang, and Qingguo Wang, for their generous and selfless hep. It has been a great pleasure for me to work with them.

Finally, I would like to thank my family for their continuous support at any time that I need. Without their help, I would not be able to completed my graduate study and achieve my goals in life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Computational protein structure prediction is very important for many applications in bioinformatics. Many prediction methods have been developed, including Modeller, HHpred, I-TASSER, Robetta, and MUFOLD. In the process of predicting protein structures, it is essential to accurately assess the quality of generated models. Consensus quality assessment (QA) methods, such as Pcons-net and MULTICOM-refine, which are based on structure similarity, performed well on QA tasks. The drawback of consensus QA methods is that they require a pool of diverse models to work well, which is not always available. More importantly, they cannot evaluate the quality of a single protein model, which is a very common task in protein predictions and other applications. Although many single-model quality assessment methods, such as ProQ2, MQAPmulti, OPUS-CA, DOPE, DFIRE, and RW, etc. have been developed to address that problem, their accuracy is not good enough for most real applications.

In this dissertation, based on the idea of using C-α atoms distance matrix and deep learning methods, two methods have been proposed for assessing quality of protein structures. First, a novel algorithm based on deep learning techniques, called DL-Pro, is proposed. From training examples of distance matrices corresponding to good and bad models, DL-Pro learns a stacked autoencoder network as a classifier. In experiments on selected targets from the Critical Assessment of Structure Prediction (CASP) competition, DL-Pro obtained promising results, outperforming state-of-the-art energy/scoring functions, including OPUS-CA, DOPE, DFIRE, and RW. Second, a new method

DeepCon-QA is developed to predict quality of single protein model. Based on the idea of using protein vector representation and distance matrix, DeepCon-QA was able to achieve comparable performance with the best state-of-the-art QA method in our experiments. It also takes advantage the strength of deep convolutional neural networks to “learn” and “understand” the input data to be able to predict output data precisely.

On the other hand, this dissertation also proposes several new methods for solving loop modeling problem. Five new loop modeling methods based on machine learning techniques, called NearLooper, ConLooper, ResLooper, HyLooper1 and HyLooper2 are proposed. NearLooper is based on the nearest neighbor technique; ConLooper applies deep convolutional neural networks to predict Cα atoms distance matrix as an orientation-independent representation of protein structure; ResLooper uses residual neural networks instead of deep convolutional neural networks; HyLooper1 combines the results of NearLooper and ConLooper while HyLooper2 combines NearLooper and ResLooper. Three commonly used benchmarks for loop modeling are used to compare the performance between these methods and existing state-of-the-art methods. The experiment results show promising performance in which our best method improves existing state-of-the-art methods by 28% and 54% of average RMSD on two datasets while being comparable on the other one.

CHAPTER 1. INTRODUCTION

1.1 Motivations

Knowledge of three-dimensional (3D) structure of a protein is critical for understanding its function, mutagenesis experiments and drug developments. Several experimental methods such as the X-ray crystallography or Nuclear Magnetic Resonance (NMR) can help determine a good 3D structure but they are very time-consuming and expensive [1]. To address those limitations, computational protein structure prediction methods have been developed, including Modeller [2], HHpred [3], I-TASSER [4], Robetta [5], and MUFOLD [6]. The process of predicting protein structure commonly involves generating a large number of models, from which good models are selected using some quality assessment method.

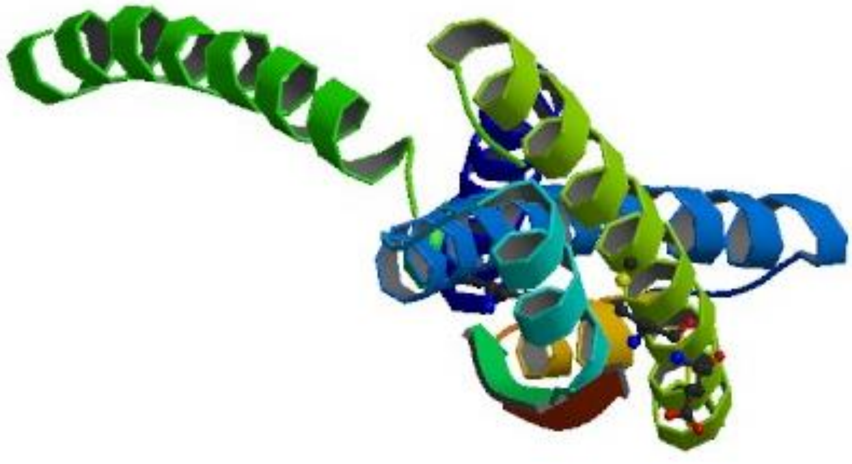


Figure 1. 3D structure of ORF52 from Murid herpesvirus 4 [33]

Although many protein model quality assessment (QA) methods have been developed, such as MUFOLD-WQA [7], QMEANClust [8] MULTICOM [9], OPUS_CA [10], RW [11], etc., they all have various limitations and are not applicable to real applications. The Critical Assessment of Structure Prediction (CASP) is a biennial world-wide event in the structure prediction community to assess the current protein modeling techniques, including QA methods. In CASPs, different prediction software programs from various research groups were given unknown proteins to predict their structures. State-of-the-art single-model quality assessment methods include various energy functions or scoring functions, such as MULTICOM [9], ProQ2 [81], MQAPmulti [82], OPUS_CA [10], DFIRE [12], RW [11], DOPE [13], etc. In CASP competitions [14,15], the accuracy of single-model QA methods has been improving consistently, but still not very high in most cases. In contrast, consensus QA methods, such as MUFOLD-WQA and Pcons-net [83], which are based on structure similarity, performed well on QA tasks, much better than single-model QA methods [14,15]. The drawback of consensus QA methods is that they require a pool of diverse models to work well, which is not always available. More importantly, they cannot evaluate the quality of a single protein model, which is a very common task in protein predictions and other applications.

1.2 Contributions

This study has three major contributions to the area of Protein Structure Prediction and Protein Quality Assessment that are summarized as follow.

First, I proposed new approach, called DL-Pro, based on C-α atoms distance matrix and deep learning methods for classifying the quality of any protein structure prediction.

To the best of our knowledge, this is the first attempt to use purely geometric information of a model and deep learning for this task of classifying the quality of single-model. Three new QA algorithms, DL-Pro, FFNN, and SVM using different learning methods have been proposed within the common framework. Experiments using selected CASP models and targets show very promising results. Compared to traditional feedforward neural networks, deep learning is better, as demonstrated by the performance difference between DL-Pro and FFNN. Deep learning was able to learn useful features representing good models and DL-Pro achieved the best results, outperforming state-of-the-art energy/scoring functions, including DFIRE, OPUS-CA, DOPE, and RW. Yet, the information used by DL-Pro is far less than other single-model QA methods. With additional model information, DL-Pro is expected to improve further.

Second, I developed a new protein quality assessment method DeepCon-QA based on deep learning (DCNN) and distance matrix representation of 3-D structures. From the rotation and translation independent representation of distance matrix, DCNN is able to learn a mapping function to predict an expected distance matrix that can be used later for evaluating quality of protein structure prediction. To the best of our knowledge, this is the first attempt to combine deep learning, geometric information and protein vector representation for protein quality assessment problem. Moreover, this research also shows that protein vector representation is a very useful set of features that can be used for QA tasks compare with profile features which are a very commonly used in many QA methods. It opens new promising research directions to solve other problems in protein prediction area. Experiments on common benchmark dataset CASP11 of hard targets have shown promising results. Compared to the best QA method MQAPmulti, our DeepCon-

QA method achieves equal result when comparing in percentage of number of cases while being comparable when comparing in the average value.

Third, I proposed five new loop modeling methods, in particular a new approach based on deep learning (DCNN, ResNet) and distance matrix representation of 3-D structures. From the rotation and translation independent representation of distance matrix, DCNN and ResNet are able to learn a mapping function to predict a loop model. Experiments using selected CASP models and com-mon benchmark datasets have shown promising results. Compared to state-of-the-art loop modeling methods, our HyLooper2 achieves the best result on two test datasets and being comparable on the other test dataset. The ResNet in ResLooper is able to generate good loop candidates in many cases. The HyLooper2 method, which is the combination of ResLooper and NearLooper achieves good performance because both methods are complementary and have their own strengths.

1.3 Outline of the Dissertation

This report is organized as follows.

Chapter 1 presents the motivations and contributions.

Chapter 2 introduces the basics of major techniques used in the proposed methods and some related works.

Chapter 3 presents DL-Pro: a novel deep learning method for protein model quality assessment.

Chapter 4 presents a new method for single-model quality assessment using deep networks and continuous distributed representation of protein Sequence.

Chapter 5 presents new deep learning methods for protein loop modeling.

Chapter 6 summarizes this dissertation.

.

CHAPTER 2. BACKGROUND & RELATED WORK

2.1 Deep Learning

2.1.1 Background of Multilayer Neural Networks

A Multi-Layer Perceptron (MLP) or multi-layer neural network defines a family of functions. For a typical example, consider the case of a single hidden layer neural network with the task to map a d-vector to an m-vector (e.g. for regression):

$$g(x) = b + W \tanh(c + Vx) \qquad (1)$$

where x is a d-vector (the input), V is an k times d matrix (called input-to-hidden weights), c is a k-vector (called hidden units' offsets or hidden unit biases), b is an m-vector (called output units offset or output units' biases), and W is an m times h matrix (called hidden-to-output weights) [36].

The output of the hidden layer is the vector-valued function $h(x)=tanh(c + V x)$. This also means the output is an affine transformation of the hidden layer. A non-linearity may be stacked on top of it in some other network architectures. The elements of the hidden layer are called hidden units [36].

The kind of operation computed by the above h(x) can be applied on h(x) itself, but with different parameters (different biases and weights). This would give rise to a feedforward multi-layer network with two hidden layers. More generally, one can build a deep neural network by stacking more of such layers. Each of these layers may have a different dimension (k above). A common variant is to have skip connections, i.e., a layer can take as input not only the layer at the previous level but also some of the lower layers [36]. Figure 2 shows the network structure of a simple artificial neural network with a single hidden layer.

Many algorithms have been proposed to train multi-layer neural networks but the most commonly used ones are stochastic gradient descent. Recently, Adam [38] has become a very popular optimization method thanks to its performance.

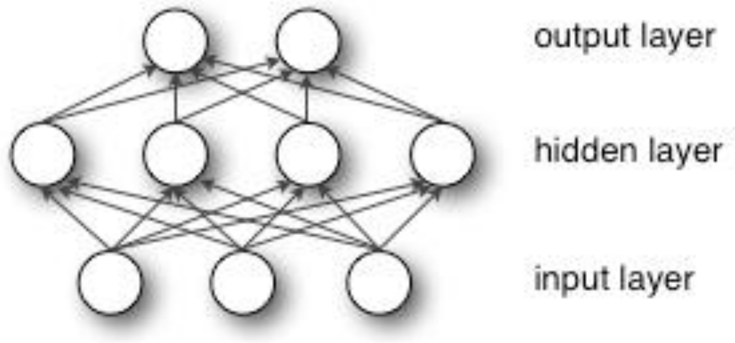


Figure 2. An artificial neural network with a single hidden layer [37]

2.1.2 Deep Learning with Convolutional Neural Networks

Basically, Convolutional Neural Networks (CNN) [40, 42] are very similar to a normal neural network that made up of neurons which have learnable weights and biases. CNN are usually comprised of a number of convolutional and subsampling layers followed by fully connected layers. The benefit of CNN is the ability to work directly on the 2D structures of an input image or any kinds of 2D input data. Another big advantage of CNN over ordinary neural networks is that they are easy to train and have a lot less parameters. Deep Convolutional Neural Networks have been shown the effectiveness in many applications such as classification [43], face recognition [44], semantic-segmentation [41], etc. This research use DCNN as a deep learning method for solving protein quality assessment problems.

Figure 3 shows architecture of LeNet-5, a typical CNN for digits' recognition [42]. It has multiple Convolutional layers with Subsampling layers following and Fully Connected layer at the end.

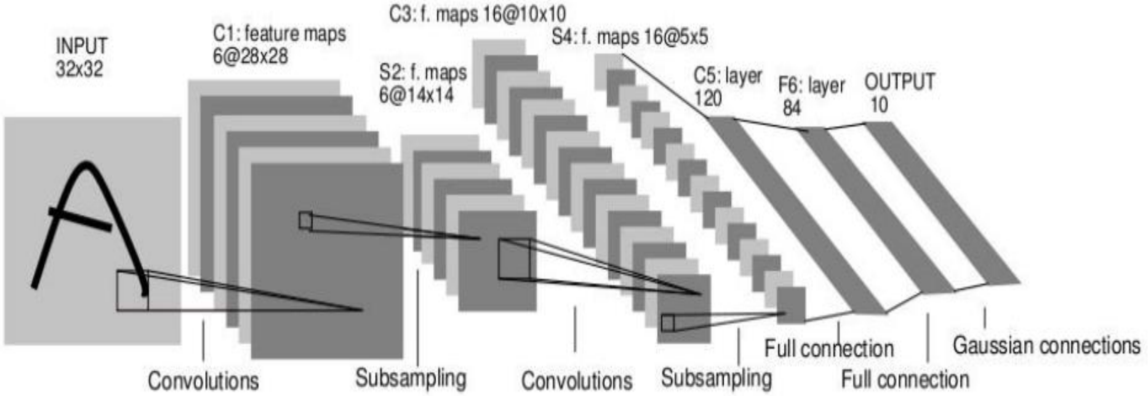


Figure 3. Architecture of LeNet-5, a typical Convolutional Neural Networks for digits' recognition [42].

2.1.3 Deep Learning with Sparse Autoencoder

An autoencoder [26-29] is a Feedforward Neural Network (FFNN) that tries to implement an identity function by setting the outputs equal to the inputs in training. Figure 4 shows an example. A compressed representation of the input data, as represented by the hidden nodes, can be learned by placing some restrictions on the network. One way is to force the network to use fewer nodes to represent the input by limiting the number of nodes in the hidden layer. Each hidden node represents a certain feature of the input data.

Autoencoders can be viewed as nonlinear low-dimensional representations as compared to linear low-dimensional representations generated by PCA. In autoencoders, the mapping of the input layer to the hidden layer is called encoding and the mapping of the hidden layer to the output layer is called decoding. In general, an autoencoder of a given structure tries to find the weights to minimize the following objective function:

$$\underset{W,b}{argmin} \quad J(W,b) = ||\, h_{W,b}(x) - x \,|| \qquad (2)$$

where x is the input, W the weights, b the biases, and h the function mapping input to output.

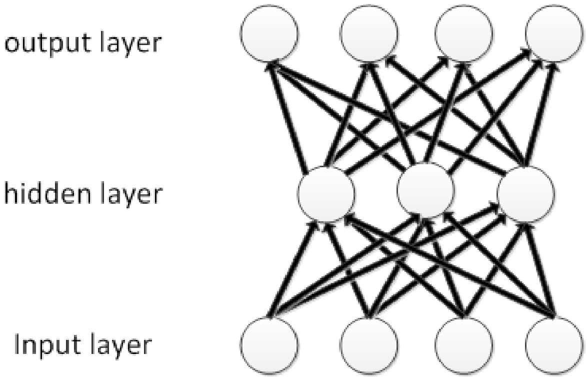


Figure 4. An example of autoencoder

Another technique of forcing an autoencoder to learn compressed representation is sparsity regularization on the hidden nodes, i.e., only a small fraction of hidden nodes are active for an input. With sparsity regularization, the number of hidden nodes can be more than that of the input nodes. Specifically, let

$$\hat{p}_j = \frac{1}{m}\sum_{i=1}^{m}\left[a_j^{(2)}(x^{(i)})\right] \qquad (3)$$

be the average activation of hidden unit j over a training set of size m. The goal here is to make $\hat{p}$ approximate a given sparsity parameter p. To measure the difference between p and $\hat{p}$, an extra penalty term can be added to Eq. (2):

$$R = \sum_{j=1}^{s_2} plog\frac{p}{\hat{p}_j} + (1-p)log\frac{1-p}{1-\hat{p}_j} \qquad (4)$$

where s_2 is the number of nodes in the hidden layer and j a hidden node. The value reaches minimum of 0 when $\hat{p}_j = p$ and goes to infinity as $\hat{p}_j$approaches 0 or 1. Now, the overall cost function becomes

$$J_{sparse}(W,b) = J(W,b) + \beta R \qquad (5)$$

where parameter β defines the tradeoff between the mapping quality and the sparsity of a network.

Given the objective function in Eq. (5), its derivatives w.r.t. W and b can be derived analytically. Variants of backpropagation algorithms can find optimal W and b values iteratively on training examples.

Stacked autoencoders are deep learning networks constructed using autoencoders layer-by-layer. Another autoencoder can be constructed on top of a trained autoencoder by treating the learned feature detectors in the hidden layer of the trained autoencoder as visible input layer. Autoencoder training is unsupervised learning since only unlabeled data are used. The learned weights and biases will be used as the starting point for the fine-tuning supervised learning stage of deep learning. Figure 5 and 6 show an example of a stacked autoencoder with 2 hidden layer. First, we would train the first hidden layer with input and get the activations of those hidden units. Then, these activations are used as input for the next hidden layer to do training. Finally, combining the 2 hidden layers will give us a stacked autoencoder.

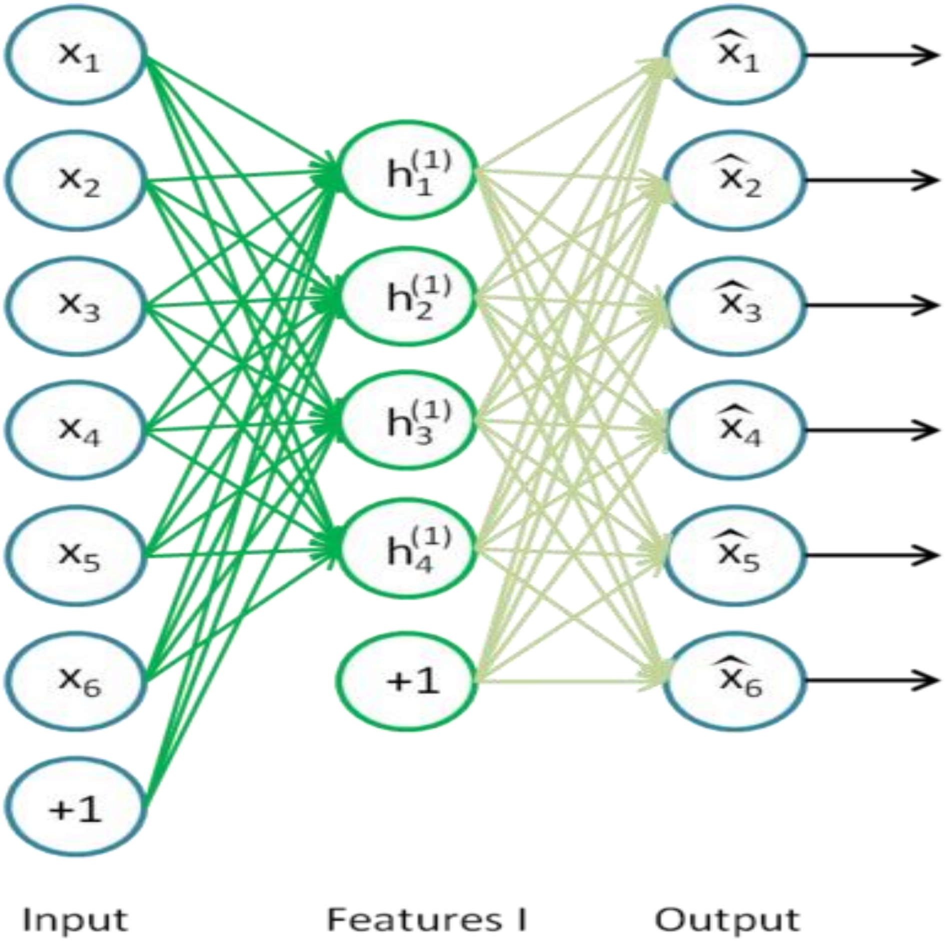


Figure 5. Train first layer of a stacked autoencoder

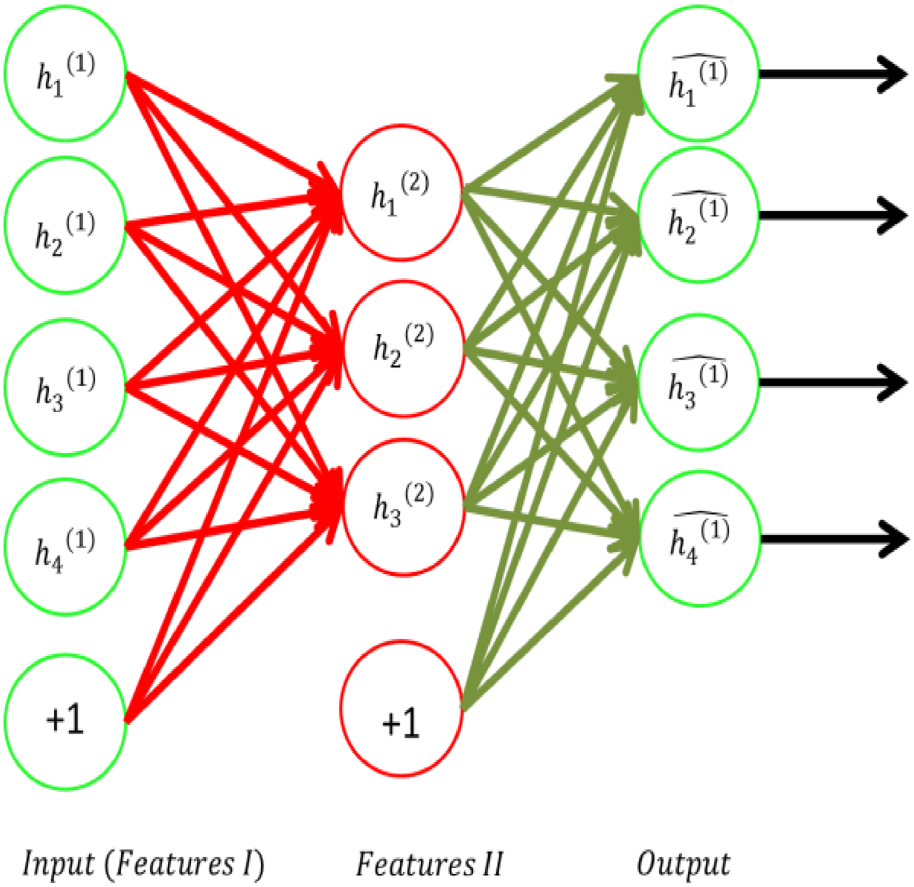


Figure 6. Train second layer of a stacked autoencoder

The supervised learning stage adds a label layer, such as a softmax classifier, as the highest layer. First, the softmax classifier is trained using labeled data. Then the whole multilayer deep network is treated as a feedforward network and trained using backpropagation, starting with weights and biases learned before.

Figure 7 shows an example of a sparse autoencoder with a softmax classifier on top of that [31].

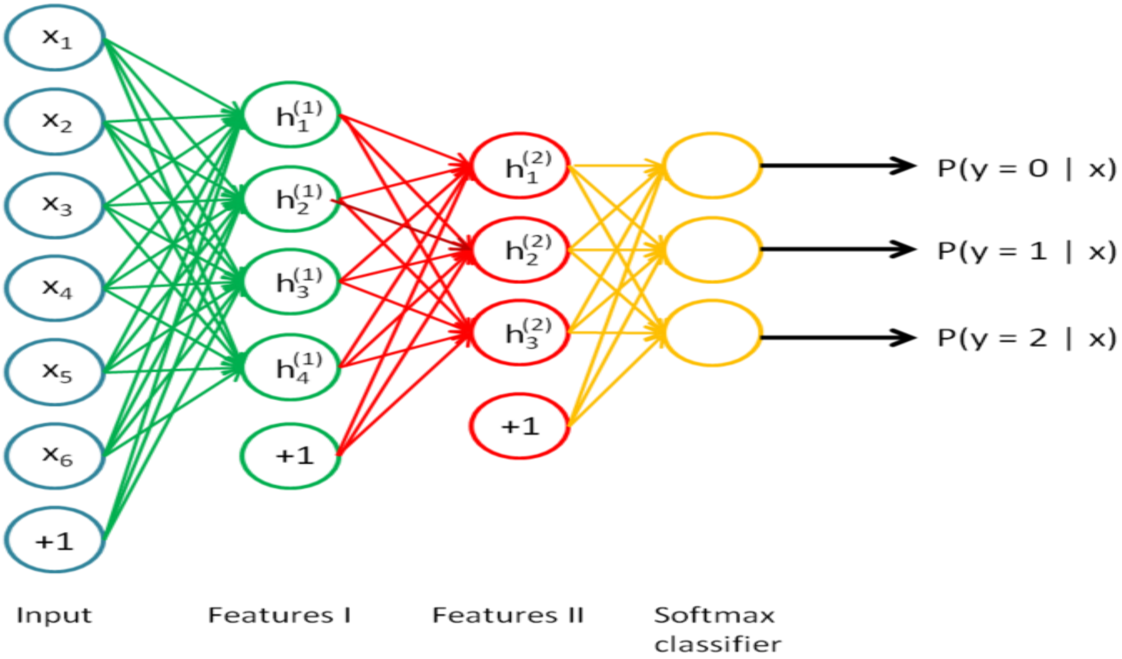


Figure 7. A stacked autoencoder with 2 hidden layers and a softmax classifier on top.

2.2 Protein Model Quality Evaluation

Protein model quality assessment methods can be divided into two main approaches: energy or scoring functions and consensus methods [16]. Basically, energy or scoring functions are designed based on either physical properties at molecule levels [17,18], such as thermodynamic equilibrium or statistics based properties derived based on information from known structures [19, 20]. On the other hand, consensus methods are based on the idea that given a pool of predicted models, a model that is more similar to other models is closer to the native structure [21].

2.2.1 Consensus methods based on structure similarity

A vital part of consensus methods is the measurement of similarity between two 3-D structures. There are three commonly used metrics: the Root-Mean-Squared Deviation (RMSD) Score, Template Modeling Score (TM-score), and Global Distance Test Total Score (GDT_TS) [22,23,24].

Since CASP data is used in this study and GDT-TS is a main metric used in the official CASP evaluation, we use GDT_TS as our main metric of evaluation. It is calculated by 1) superimposing two models over each other and 2) averaging the percentage of corresponding C-α atoms between two models within a certain cutoff. The GDT-TS value between two models is computed as follows:

$$GDT\_TS(U_i, U_j) = (P_1 + P_2 + P_3 + P_4)/4 \tag{6}$$

where Ui and Uj are two 3D models and Pd is the percentage that the C-α atoms in Ui is within a defined cutoff distance d, d∈ {1,2,4,8}, from the corresponding C-α atoms in Uj [18]. GDT_TS values have the range of [0, 1] with higher value means two structures are more similar.

For a model of a protein, its true quality is the GDT_TS value between it and the native structure of the protein, which is called its true GDT_TS score in this report.

Using GDT_TS as the measurement of model similarity, the consensus methods are designed as follows: given a set of prediction models U and a reference set R, the consensus score, the CGDT_TS score, of each model Si is defined as:

$$CGDT\_TS(U_i) = \sum_{j \in R} GDT\_TS\left(U_i, U_j\right) / N \tag{7}$$

where the reference set R can be U or a subset of U. CGDT_TS values also range from 0 to 1 with higher value means better.

2.2.2 Energy or scoring functions

Energy or scoring functions are widely used for assessing quality of a given predicted protein model. In this study, we use 4 state-of-the-art energy functions, OPUS_CA, DFIRE, RW, and DOPE, which have been used widely in practice as well as in CASP competitions, for comparison.

OPUS_CA uses a statistics-based potential function based on the C-α positions in a model. It mainly consists of seven major representative molecular interactions in proteins: distance-dependent pairwise energy with orientation preference, hydrogen bonding energy, short-range energy, packing energy, tri-peptide packing energy, three-body energy, and salvation energy [10].

DFIRE is also a statistics-based scoring function, defined based on a reference state, called the distance-scaled, finite ideal-gas reference state. A residue-specific all-atom potential of mean force from a database of 1011 nonhomologous (less than 30% homology) protein structures with resolution less than 2 A is constructed by using the reference state. DFIRE works better with a full atom model than only backbone and C (beta) atoms. It belongs to distance-dependent, residue-specific potentials [12].

RW is a side-chain orientation dependent potential method derived from random-walk reference state for protein fold selection and structure prediction. It has two major functions: 1) a side chain orientation-dependent energy function and 2) a pairwise distance-dependent atomic statistical potential function using an ideal random-walk chain as reference state [11].

Discrete Optimized Protein Energy (DOPE) is an atomic distance-dependent statistical potential method derived from a sample of native protein structures. Like DFIRE, it is based on a reference state that corresponds to non-interacting atoms in a homogeneous sphere with the radius dependent on a sample native structure. A non-redundant set of 1472 crystallographic structures was used to derive the DOPE potential. It was incorporated into the modeling package MODELLER-8 [8].

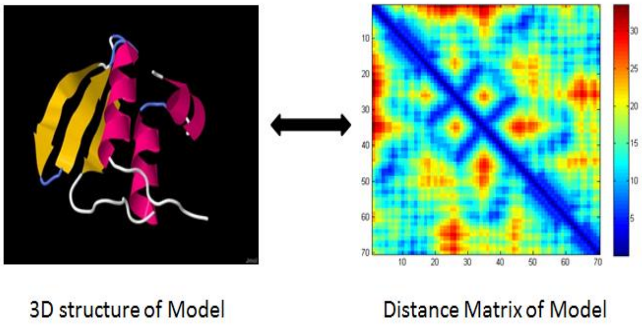
2.3 Distance Matrix

A 3D model with n C-α atoms can be converted into an n by n distance matrix A, i.e. calculating the Euclidean distance of two points in a 3D space, as follows:

$$A_{ij} = \sqrt{\left(U_x^i - U_x^j\right)^2 + \left(U_y^i - U_y^j\right)^2 + (U_z^i - U_z^j)^2} \qquad (8)$$

where $U_{x,y,z}^i$, $U_{x,y,z}^j$ are the 3D coordinates of points i and j, respectively.

Figure 8 shows an example of the 3D structure and its corresponding distance matrix of a protein model.



3D structure of Model

Distance Matrix of Model

Figure 8. The 3D structure and its corresponding distance matrix of a protein model.

2.4 Principal component analysis (PCA)

PCA [25] is a widely used statistical method for linear dimensionality reduction using orthogonal transformation. Normally, the input is normalized to zero mean. Then the singular value decomposition is used on the input's covariance matrix to derive eigenvectors and eigenvalues. A subset of eigenvectors can be used to project the input to a lower-dimensional representation. The eigenvalues indicate how much information is retained when reducing the dimensionality of the input.

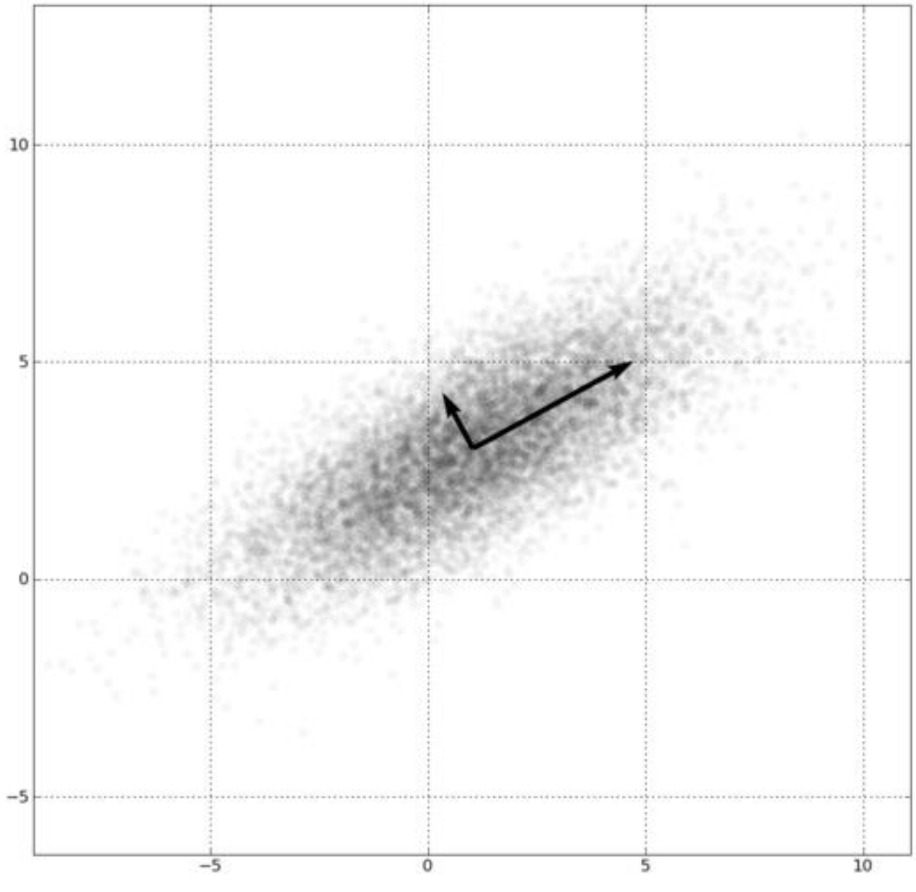


Figure 9. PCA of a multivariate Gaussian distribution

Figure 9 [32] shows and example of a multivariate Gaussian distribution centered at (1,3) with a standard deviation of 3. The eigenvectors of the covariance matrix are presented by 2 arrows. The lengths of arrows are scaled by the square root of the corresponding eigenvalue.

2.5 Continuous Distributed Representation of Protein Sequences

Feature extraction is very essential for data analysis, machine learning and many other things. Continuous distributed representation of protein sequences is a part of the work from Asgari et. al. [45] that has been shown to be a good representation for a protein sequence. They use artificial neural network approaches and represent a protein sequence with a single dense n-dimensional vector. Different experiments have been done with this representation methods to show how accurate it is in representing a protein sequence. One experiment has been done in classification of 324018 protein sequences received from Swiss-Prot and an average family classification accuracy of 93% is obtained, outperforming existing methods. Another experiment that use this representation to predict disordered proteins from structured proteins has been done and achieved almost perfect accuracy.

From the promising result of [45], this research uses the continuous distributed representation of protein sequence as one of the most essential features that will be used to solve our QA task. By combining this information with other evolutionary information, we expect our QA method to have a better performance than not using it.

CHAPTER 3. NOVEL DEEP LEARNING METHOD FOR PROTEIN MODEL QUALITY CLASSIFICATION

3.1 Introduction

Knowledge of three-dimensional (3D) structure of a protein is critical for understanding its function, mutagenesis experiments and drug developments. Several experimental methods such as the X-ray crystallography or Nuclear Magnetic Resonance (NMR) can help determine a good 3D structure but they are very time-consuming and expensive [1]. To address those limitations, computational protein structure prediction methods have been developed, including Modeller [2], HHpred [3], I-TASSER [4], Robetta [5], and MUFOLD [6]. The process of predicting protein structure commonly involves generating a large number of models, from which good models are selected using some quality assessment method.

Although many protein model quality assessment (QA) methods have been developed, such as MUFOLD-WQA [7], QMEANClust [8] MULTICOM [9], OPUS_CA [10], RW [11], etc., they all have various limitations and are not applicable to real applications. The Critical Assessment of Structure Prediction (CASP) is a biennial world-wide event in the structure prediction community to assess the current protein modeling techniques, including QA methods. In CASPs, different prediction software programs from various research groups were given unknown proteins to predict their structures. State-of-the-art single-model quality assessment methods include various energy functions or scoring functions, such as OPUS_CA [10], DFIRE [12], RW [11], DOPE [13], etc. In CASP competitions [14,15], the accuracy of single-model QA methods has been improving consistently, but still not very high in most cases. In contrast, consensus QA methods, such as MUFOLD-WQA and United3D, which are based on structure similarity, performed well on QA tasks, much better than single-model QA methods [14, 15]. The drawback of consensus QA methods is that they require a pool of diverse models to work well, which is not always available. More importantly, they cannot evaluate the quality of a single protein model, which is a very common task in protein predictions and other applications.

In this research, a novel QA method based on deep learning techniques, called DL-Pro, is proposed for single-model quality assessment, specifically the identification of native-like models. Different from existing energy/scoring functions and consensus approaches, DL-Pro is a purely geometry based method. For a protein model, DL-Pro uses its distance matrix that contains pairwise distances between two residues' C-α atoms in the model, which sometimes is also called contact map, as an orientation-independent representation. From training examples of distance matrices corresponding to good and bad models, DL-Pro learns a stacked autoencoder network as a classifier. In experiments using CASP datasets, DL-Pro is compared with existing state-of-the-art energy/scoring functions, including OPUS-CA, DOPE, DFIRE, and RW, and shows improvement in prediction accuracy.

3.2 Methods

3.2.1 Problem formulation

The QA problem is formulated as a classification problem in this report: given a set of predicted models of a protein, classify them into two classes, good (or near-native) and bad.

For the experiments, we prepare the dataset that contains good and bad models, but not intermediate models, as follows. Let A be a set of n predicted models for a target protein of length l, $A=\{a_i,\ 1 \leq i \leq n\}$, and $a_i = \{U^j,\ j \in [1,l]\}$ where U^j is the 3D coordinates of residue j of model a_i. Let $C = \{c_i,\ 1 \leq i \leq n,\ 0 \leq c_i \leq 1\}$ be the true-GDT_TS scores, i.e. the true quality, of models in A. Then, the classification label of a model is P (for near native) if its true GDT_TS score $c_i \geq 0.7$, and label $\overline{P}$ (for not near native) if its true GDT_TS score $c_i < 0.4$. Note that models with true GDT_TS scores between 0.4 and 0.7 are dropped from the dataset. Our focus in the paper is to separate good models from bad models.

In this report, the performance metric of a classification algorithm is classification accuracy T:

$$T=v/n \qquad (9)$$

where v is the number of correctly classified examples and n is the total number of examples.

3.2.2 Classification using energy or scoring functions

For comparison purpose, we adapt existing energy or scoring functions for the classification problem defined in the previous subsection. The general method, call EC (Energy function based Classification), can be applied to existing energy or scoring functions, including the four used in this report, OPUS-CA, DOPE, DFIRE, and RW. For these four scoring functions, smaller values represent better models and near-native models have very negative values. Since the true GDT_TS scores of models are in the range [0-1] and larger value means better, a linear mapping from energy scores to the true GDT_TS scores is first learned from a set of training examples and then the thresholds corresponding to good (true GDT_TS score ≥ 0.7) and bad (true GDT_TS score $c_i < 0.4$) models are determined. Later, the energy scores of test examples are first converted using the linear mapping and then their classes are determined using the learned thresholds.

Table 1 shows the pseudocode of the EC algorithm. The algorithm consists of a training phase, EC_Train, and a test phase, EC_Test. Based on the energy scores and corresponding true GDT_TS scores of a set of models, which constitute the training examples, EC_Train first computes the mean and standard deviation of the energy scores for normalization, performs linear regression, and then determines a threshold to label the positive (good) and negative (bad) examples.

Specifically, EC_Train first flips the sign of energy scores from negative to positive so that bigger value means better. Then energy scores are normalized to zero mean and unit variance. Next, a linear function with parameters $\Theta1$ and $\Theta2$ is learned to fit the data of normalized energy scores and true-GDTTS scores. Two values, s1 and s2, on the normalized energy scores are calculated using the linear function from true GDTTS scores 0.4 and 0.7. Finally, the average of s1 and s1, s0, is the threshold on energy scores for labeling the two classes, good and bad models.

Table 1. Pseudocode of the EC (Energy function based Classification) algorithm.

```
Algorithm: EC(S, G, S')
Input:  S and G, energy scores and corresponding true GDT_TS scores of a set of training examples (predicted models)
        S', energy scores of a set of test examples (predicted models)

    1. $[s_0, \mu, \sigma] \leftarrow EC\_Train(S, G)$
    2. $[L] \leftarrow EC\_Test(S', s_0, \mu, \sigma)$

**Output**: $L$, predicted labels of the test set


**Function** $EC\_Train(S, G)$
**Input:** $S$ and $G$, energy scores and true GDT_TS scores
    1. $S \leftarrow -1 * S$
    2. $\mu \leftarrow \sum_{i=1}^{n} S_i / n$
    3. $\sigma \leftarrow \sqrt{\frac{1}{n} \sum_{i=1}^{n} (S_i - \mu)^2}$
    4. $S \leftarrow (S - \mu)/\sigma$
    5. Learn a Linear Regression model with
            $G = \Theta_1 + \Theta_2 * S$
    6. Derive $s_1$ & $s_2$ values from $G = 0.4$ & $G = 0.7$ respectively.
            $s_1 \leftarrow (0.4 - \Theta_1)/\Theta_2$
            $s_2 \leftarrow (0.7 - \Theta_1)/\Theta_2$
    7. $s_0 \leftarrow (s_1 + s_2)/2$
**Return** $s_0$, threshold for classification
           $\mu$, mean of energy scores
           $\sigma$, standard deviation of energy scores


**Function** $EC\_Test(S', s_0, \mu, \sigma)$
**Input:** $S'$, energy scores of test examples
           $s_0$, threshold for classification
           $\mu$, mean for normalization
           $\sigma$, standard deviation for normalization
    1. $S' \leftarrow -1 * S'$
    2. $S' \leftarrow (S' - \mu)/\sigma$
    3. If $S' \geq s_0$
            $L \leftarrow P$
       else
            $L \leftarrow \overline{P}$
**Return** $L$, predicted labels of test examples
```

On test examples, EC_Test first normalizes the energy score of a test example using the training example mean and standard deviation. Then, the example gets a positive label if the normalized energy score is larger than the threshold s0 and gets a negative label otherwise.

3.2.3 New QA methods based on C-α atom distance matrix

In this section, a new approach based on C-α atoms distance matrix and machine learning methods is proposed for single-model quality assessment and the identification of native-like models. Different from existing energy/scoring functions and consensus approaches, this new approach is purely geometry based. Various supervised machine learning algorithm can be used in this approach and three algorithms based on deep learning networks, support vector machines (SVM), and feed-forward neural networks (FFNN), respectively, are presented next.

3.2.3.1 DL-Pro, a new deep learning QA algorithm using C-α atom distance matrix

DL-Pro is a novel QA algorithm based on deep learning techniques. For a protein model, DL-Pro uses its distance matrix that contains pairwise distances between two residues' C-α atoms in the model, which sometimes is also called contact map, as an orientation-independent representation. From training examples of distance matrices corresponding to good and bad models, DL-Pro learns a stacked sparse autoencoder classifier to classify good and bad models.

Table 2 shows the pseudocode of the DL-Pro algorithm. DL-Pro consists of a training phase, DL-Pro_Train, and a test phase, DL-Pro_Test. Based on the 3D structures and labels of a set of training models, DL-Pro_Train first computes the distance matrix composed of pairwise distances between every pair of residues' C-α atoms in the model.

Then, the distance matrix is normalized to mean 0 and standard deviation 1 based on its mean and standard deviation, which are kept for future use in testing. Next, PCA is applied to reduce the dimension of the distance matrices to generate the inputs of training examples. Significant reduction can be achieved even when 99% of information is kept, i.e., keeping 99% variance of the original data set.

The top eigenvectors are kept for future use in testing. Finally, a deep learning network consisting of one or more layers of sparse autoencoders followed by a softmax classifier is trained using the training examples.

On test examples, DL-Pro_Test first pre-processes a test model by calculating its distance matrix, normalizing the matrix using learned mean and standard deviation, and reducing the matrix dimension using PCA with learned eigenvectors. Then, the learned deep learning network classifier is used to classify the data.

Table 2. Pseudocode of the DL-Pro algorithm, a novel QA algorithm based on deep learning and model distance matrix of pairwise distances between two residues' C-α atoms in a model.

```
Algorithm: DL-Pro(U, L, U')

Input:  U and L, 3D structures and corresponding labels of training examples
        U', 3D structures of test examples

    [DL_params,V, $\mu$, $\sigma$] ← DL-Pro_Train(U, L)
    [L''] ← DL-Pro_Test(U', DL_params, V, $\mu$, $\sigma$)

Output: L'', predicted labels of test examples

Function DL-Pro_Train(U, L)
Input:  U and L, 3D structures and corresponding labels
    M ← S2D (U)
    $\mu \leftarrow \sum_{i=1}^{n} M_i / n$
    3. $\sigma \leftarrow \sqrt{\frac{1}{n}\sum_{i=1}^{n}(M_i - \mu)^2}$
    4. $M \leftarrow (M - \mu)/\sigma$
    5. [M', V] ← PCA(M)
    6. DL_params ← DL_Train(M', L)
Return   DL_params, parameters for deep learning classifier
         V, eigenvectors for dimension reduction
         $\mu$, mean for normalization
         $\sigma$, standard deviation for normalization

Function DL-Pro_Test(U, DL_params, V, $\mu$, $\sigma$)
    1. M ← S2D(U)
    2. $M \leftarrow (M - \mu)/\sigma$
    3. [M'] ← PCA_Test(M, V)
    4. [L'] ← DL(DL_params,M')
Return   L'
```

Function *M = S2D(U)*

This function uses Eq. (3) to convert the 3D structure of model *U* to distance matrix. Because the matrix is symmetric, only its upper triangular part is kept in *M*.

Function *[M', V] = PCA(M)*

This function uses singular value decomposition to derive eigenvectors, *V*, and eigenvalues of *M*. Then, *M* is converted to M*'* in reduced dimensions spanned by a subset of the most significant eigenvectors of *V*.

Function *DL_Train(M', L)*

This function trains a deep learning network using input data *M'* and corresponding labels *L*. The deep learning network consists of one or more layers of sparse autoencoders and a final layer of a softmax classifier.

Function *PCA_Test(M, V)*

This function uses the eigenvectors *V* to convert *M* to a reduced size *M'*.

Function *DL(DL_params, M')*

This function uses the learned deep learning network classifier, DL_params, to classify test examples M'.

3.2.3.2 SVM-Pro, a new Support Vector Machine (SVM) QA algorithm using C-α atom distance matrix

Instead of stacked autoencoder classifiers, other classifiers such as SVM can also be used in the approach based on C-α atom distance matrix. The algorithm using SVM is similar to the DL-Pro algorithm in Table 2, with only two differences: 1) Step 6 of DL-Pro_Train is replaced by training a SVM classifier using the examples to get SVM parameters. 2) Step 4 of DL-Pro_Test is replaced by SVM classification [30].

Table 3. Pseudocode of the SVM-Pro algorithm, a novel QA algorithm based on support vector machine and model distance matrix of pairwise distances between two residues' C-α atoms in a model.

```
Algorithm: SVM-Pro(U, L, U')

Input:  U and L, 3D structures and corresponding labels of training examples
        U', 3D structures of test examples

    1. [SVM_params,V, $\mu$, $\sigma$] ← SVM-Pro_Train(U, L)
    2. [L''] ← SVM-Pro_Test(U', SVM_params, V, $\mu$, $\sigma$)

Output: L'', predicted labels of test examples

Function SVM-Pro_Train(U, L)
Input:  U and L, 3D structures and corresponding labels
    1. M ← S2D (U)
    2. $\mu \leftarrow \sum_{i=1}^{n} M_i / n$
    3. $\sigma \leftarrow \sqrt{\frac{1}{n}\sum_{i=1}^{n}(M_i - \mu)^2}$
    4. $M \leftarrow (M - \mu)/\sigma$
    5. [M', V] ← PCA(M)
    6. SVM_params ← SVM_Train(M', L)
Return  SVM_params, parameters for deep learning classifier
        V, eigenvectors for dimension reduction
        $\mu$, mean for normalization
        $\sigma$, standard deviation for normalization
```

```
Function SVM-Pro_Test(U, DL_params, V, μ, σ)
    1. M ← S2D(U)
    2. M ← $(M - \mu)/\sigma$
    3. [M'] ← PCA_Test(M, V)
    4. [L'] ← SVM(SVM_params,M')
Return  L'

Function M = S2D(U)
   This function uses Eq. (3) to convert the 3D structure of model U to distance matrix.
Because the matrix is symmetric, only its upper triangular part is kept in M.

Function [M', V] = PCA(M)
   This function uses singular value decomposition to derive eigenvectors, V, and
eigenvalues of M. Then, M is converted to M' in reduced dimensions spanned by a subset
of the most significant eigenvectors of V.

Function SVM_Train(M', L)
  This function trains a SVM classifier using input data M' and corresponding labels L.

Function PCA_Test(M, V)
  This function uses the eigenvectors V to convert M to a reduced size M'.

Function SVM(SVM_params, M')
This function uses the learned SVM classifierto classify test examples M'.
```

3.2.3.3 FFNN-Pro, a new Feedforward Neural Network (FFNN) algorithm using C-α atom distance matrix

In this algorithm, FFNNs, instead of deep learning classifiers or SVMs, are used to perform supervised learning and classification. Again, this algorithm is similar to DL-Pro with Step 6 of DL-Pro_Train and Step 4 of DL-Pro_Test are replaced by FFNN training and testing.

Table 4. Pseudocode of the SVM-Pro algorithm, a novel QA algorithm based on support vector machine and model distance matrix of pairwise distances between two residues' C-α atoms in a model.

```
Algorithm: FFNN-Pro(U, L, U’)

Input:  U and L, 3D structures and corresponding labels of training examples
        U’, 3D structures of test examples

    1. [FFNN _params,V, $\mu$, $\sigma$] ← FFNN -Pro_Train(U, L)
    2.  [L”] ← FFNN -Pro_Test(U’, FFNN _params, V, $\mu$, $\sigma$)

Output: L”, predicted labels of test examples

Function FFNN -Pro_Train(U, L)
Input:  U and L, 3D structures and corresponding labels
    1. M ← S2D (U)
    2. $\mu \leftarrow \sum_{i=1}^{n} M_i / n$
    3. $\sigma \leftarrow \sqrt{\frac{1}{n}\sum_{i=1}^{n}(M_i - \mathrm{\mu})^2}$
    4. M ← $(M - \mathrm{\mu})/\sigma$
    5. [M’, V] ← PCA(M)
    6. FFNN _params ← FFNN _Train(M’, L)
Return   FFNN _params, parameters for deep learning classifier
         V, eigenvectors for dimension reduction
         $\mu$, mean for normalization
         $\sigma$, standard deviation for normalization

Function FFNN -Pro_Test(U, FFNN _params, V, $\mu$, $\sigma$)
    1. M ← S2D(U)
    2. M ← $(M - \mathrm{\mu})/\sigma$
    3. [M’] ← PCA_Test(M, V)
    4. [L’] ← FFNN (FFNN _params,M’)
Return   L’

Function M = S2D(U)
   This function uses Eq. (3) to convert the 3D structure of model U to distance matrix.
Because the matrix is symmetric, only its upper triangular part is kept in M.

Function [M’, V] = PCA(M)
   This function uses singular value decomposition to derive eigenvectors, V, and
eigenvalues of M. Then, M is converted to M’ in reduced dimensions spanned by a subset
of the most significant eigenvectors of V.
```

Function *FFNN _Train(M’, L)*
This function trains a FFNN classifier using input data *M’* and corresponding labels *L*.

Function *PCA_Test(M, V)*
This function uses the eigenvectors *V* to convert *M* to a reduced size *M’*.

Function *FFNN (FFNN _params, M’)*
This function uses the learned FFNN classifier to classify test examples *M’*.

3.3 Experimental results

3.3.1 Data set

CASP dataset: 20 CASP targets with sequence length from 93 to 115 are selected. Each target has approximately 200 predicted models. To reduce redundancy, all models that have the same GDT_TS score are removed. All models shorter than 93 residues are also removed. To make all examples the same input size, all models longer than 93 are truncated at the beginning and end, and the middle segment of 93 residues are kept. In the end, the dataset has good and bad 1,117 models.

Protein native structure dataset: The native structures of a set of protein with sequence length from 93 to 113 are downloaded from Protein Data Bank's website. These native structures are compared with the native structures of the 20 CASP targets selected. If a native structure is more than 80% similar to a CASP target, it is removed to make sure that the training set and test set used in our experiments do not overlap. Similarly to the CASP set, structures longer than 93 are truncated on both ends to get to 93. In the end, the dataset has 972 structures.

For a model of length 93, the size of the upper triangle portion of the 93 by 93 distance matrix is 4278, a very high dimensional input to a typical classifier. After applying PCA with 99% information retained, the input dimension is reduced to 358, much more manageable.

3.3.2 Classification performance of energy functions

In this experiment, the EC (Energy function based Classification) algorithm in Table 1 is applied to the CASP dataset with different energy scores obtained from OPUS-

CA, DOPE, DFIRE, and RW, respectively. The experiment results are from 4-fold cross-validation: the dataset is divided into 4 folds, each containing models of 5 targets. The EC algorithm is run 4 times total, each using 3 folds as training examples and 1-fold as test examples. The final result is the average of the 4 runs.

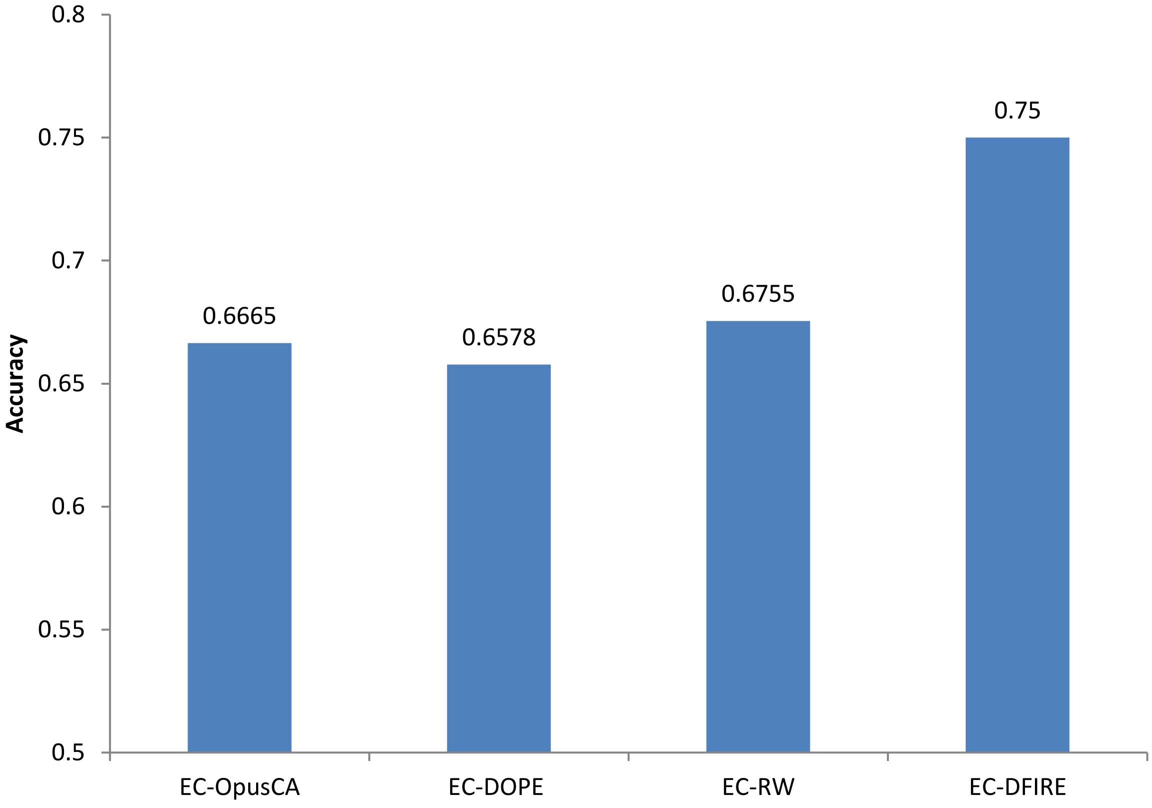


Figure 10. Classification performance of energy function based classification algorithms. The EC algorithm in Table 1 is applied to the CASP dataset with different energy scores obtained from OPUS-CA, DOPE, DFIRE, and RW, respectively.

Figure 10 shows classification accuracy of the four energy function based algorithms. EC-DFIRE achieves the best performance, 75% accuracy, while the other three have similar results, around 66%. Table 5 shows the confusion matrix of EC-DFIRE.

Positive examples are predicted very accurately, 660 out of 740, 89%, whereas prediction accuracy on negative examples is much lower, 182 out of 375, 49%.

Table 5. Confusion matrix of EC-DFIRE on the CASP dataset

	Predicted Positive	*Predicted Negative*
Actual Positive	660	82
Actual Negative	193	182

3.3.3 Classification performance of QA algorithms based on C-α atom distance matrix: DL-Pro, SVM-Pro, and FFNN-Pro

In this experiment, the CASP dataset is again divided into 4 folds, each containing models of 5 targets, and the results of 4-fold cross-validation are reported. The SVM-Pro and FFNN-Pro algorithms only use the CASP dataset, whereas DL-Pro uses something extra, the protein native structure dataset, in its unsupervised autoencoder learning stage.

For the FFNN algorithm, 1 hidden layer networks with different hidden units (25, 50, 100, 150, 200, and 250) were tried. For the DL-Pro algorithm, 1 and 2 hidden layer networks were tried. For 1-hidden-layer configurations (referred to as DL-Pro1), various numbers of hidden units (50, 100, 150, 200, and 250) were tried. For 2-hidden-layer configurations (referred to as DL-Pro2), the first hidden layer is fixed at 300 hidden units, while the 2nd hidden layer has various numbers of hidden units (100, 200, 300, 400, and 500). Other parameters are listed in Table 6. For each configuration, DL-Pro and FFNN ran for 10 times from random initial weights and their average results are reported.

Figure 11 shows classification accuracy of DL-Pro1 (DL-Pro with one-hidden-layer configurations), DL-Pro2 (DL-Pro with two-hidden-layer configurations), and FFNN with various hidden units. Their performance changes slightly as the number of hidden units changes. DL-Pro1 with 100 hidden units yields the best result with accuracy of 0.78.

Figure 12 compares classification performance of EC-DFIRE (the best of energy functions), SVM-Pro, FFNN-Pro, DL-Pro1 and DL-Pro2. SVM-Pro with quadratic kernel function and DL-Pro algorithms are better than FFNN-Pro. Both DL-Pro1 and DL-Pro2 are slightly better than EC-DFIRE, with DL-Pro1 achieving 78% accuracy, the best overall. The performance difference between FFNN-Pro and DL-Pro shows that deep learning is able to learn better features from both labeled and unlabeled data to achieve improved performance over traditional neural networks.

Table 7 shows the confusion matrix of SVM-Pro with a pretty low number of False Negative predictions. Table 8 shows the confusion matrix of FFNN-Pro with 1 hidden layer of 150 hidden units. Its accuracy is 81% on positive examples and 47% on negative examples. Finally, Table 9 shows the confusion matrix of DL-Pro1 with 1 hidden layer of 100 hidden units. Its accuracy on positive examples is excellent, 95%, but not so good on negative examples, only 45%.

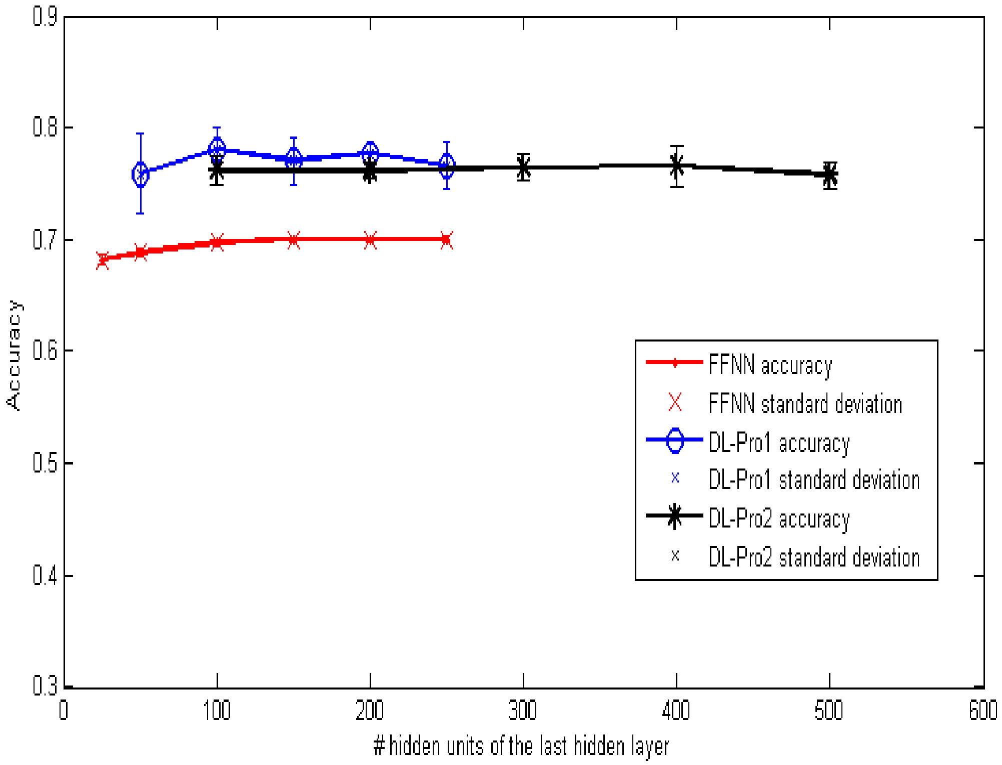


Figure 11. Classification performance of DL-Pro1 (DL-Pro with one-hidden-layer configurations), DL-Pro2 (DL-Pro with two-hidden-layer configurations), and FFNN with various hidden units.

Table 6. Parameters of sparse autoencoder training in the DL-Pro algorithm used in the experiments.

Parameter	Value
Sparsity	0.1
Weight decay λ	3e-3
Weight of sparsity penalty β	3
Maximum number of iterations	500
Optimization method	'lbfgs'

Table 7. Confusion matrix of the SVM-Pro QA algorithm based on C-α atom distance matrix

	Predicted Positive	*Predicted Negative*
Actual Positive	740	2
Actual Negative	263	112

Table 8. Confusion matrix of FFNN-Pro with 1 hidden layer of 150 hidden units.

	Predicted Positive	*Predicted Negative*
Actual Positive	604	138
Actual Negative	198	177

Table 9. Confusion matrix of DL-Pro1 with 1 hidden layer of 100 hidden units.

	Predicted Positive	*Predicted Negative*
Actual Positive	704	38
Actual Negative	207	168

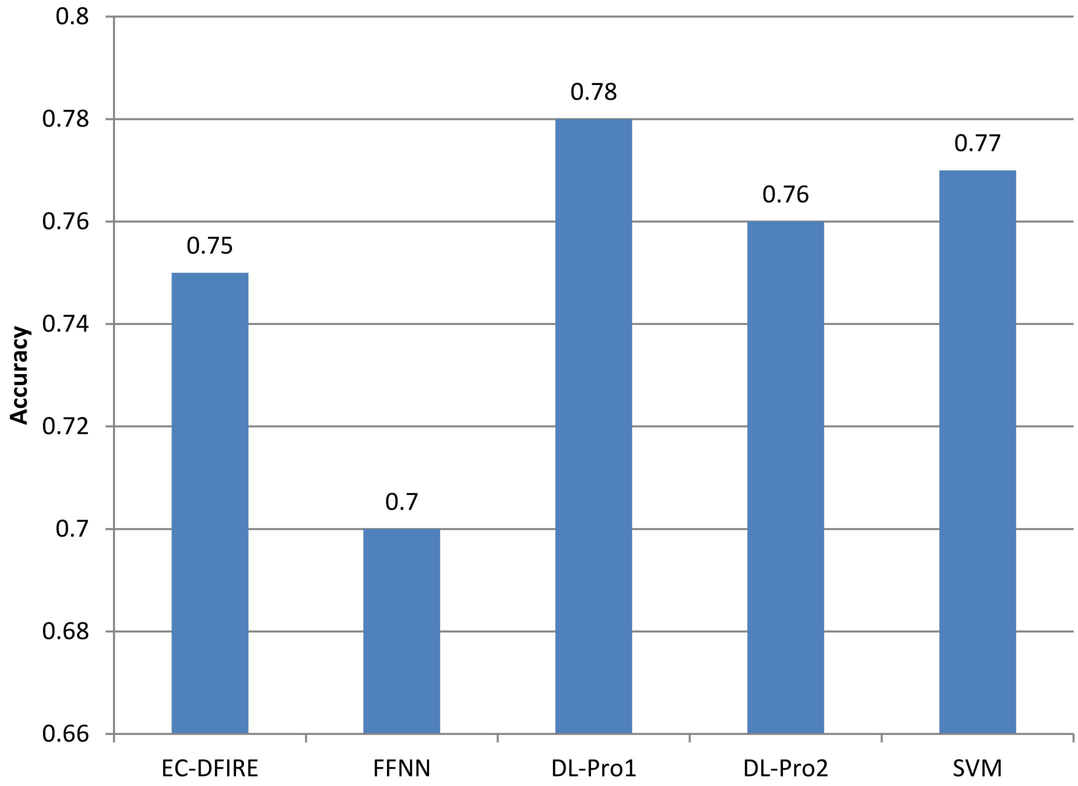


Figure 12. Classification performance of EC-DFIRE (the best of energy functions), SVM, FFNN, DL-Pro1 and DL-Pro2.

CHAPTER 4. DEEP NETWORKS AND CONTINUOUS DISTRIBUTED REPRESENTATION OF PROTEIN SEQUENCES FOR PROTEIN QUALITY ASSESSMENT

4.1 Introduction

Knowledge of three-dimensional (3D) structure of a protein is critical for understanding its function, mutagenesis experiments and drug developments. Several experimental methods such as the X-ray crystallography or Nuclear Magnetic Resonance (NMR) can help determine a good 3D structure but they are very time-consuming and expensive [1]. To address those limitations, computational protein structure prediction methods have been developed, including Modeller [2], HHpred [3], I-TASSER [4], Robetta [5], and MUFOLD [6]. The process of predicting protein structure commonly involves generating a large number of models, from which good models are selected using some quality assessment method.

Although many protein model quality assessment (QA) methods have been developed, such as MUFOLD-WQA [7], QMEANClust [8] MULTICOM [9], ProQ2 [81], MQAPmulti [82], etc. they all have various limitations and are not applicable to real applications. The Critical Assessment of Structure Prediction (CASP) is a biennial world-wide event in the structure prediction community to assess the current protein modeling techniques, including QA methods. In CASPs, different prediction software programs from various research groups were given unknown proteins to predict their structures. State-of-the-art single-model quality assessment methods include various methods using evolutionary information or scoring functions, such as ProQ2 [81], MULTICOM [9], MQAPmulti [82], etc. In CASP competitions [14,15], the accuracy of single-model QA methods has been improving consistently, but still not very high in most cases. In contrast, consensus QA methods, such as MUFOLD-WQA and Pcons-net [83], which are based on structure similarity, performed well on QA tasks, much better than single-model QA methods [14, 15]. The drawback of consensus QA methods is that they require a pool of diverse models to work well, which is not always available. More importantly, they cannot evaluate the quality of a single protein model, which is a very common task in protein predictions and other applications.

In this research, a novel QA method based on deep learning techniques, called DeepCon-QA, is proposed for single-model quality assessment. Different from existing single model QA methods and consensus approaches, DeepCon-QA is combining deep convolutional neural network (DCNN) [12,31] and a novel set of features called protein vector representation [34]. For a protein model, DeepCon-QA uses its distance matrix that contains pairwise distances between two residues' C-α atoms in the model, which sometimes is also called contact map, as an orientation-independent representation. From training samples with input as features from protein vector representation as well as other useful evolutionary information and output as distance matrix of protein model, DeepCon-QA learns a DCNN to solve the protein quality assessment problem. In experiments using datasets from CASP11, DeepCon-QA is compared with existing state-of-the-art QA methods, including ProQ2 and MQAPmulti, and shows improvement in promising results.

4.2 Methods

4.2.1 Problem formulation

The QA problem is formulated as follow: Let A be a set of predicted structures for a target protein P, and q be the size of A, A = {a_i, 1≤i≤q}. Let Y = {y_i, 1≤i≤q} be the true quality score of each predicted model a_i. This score is usually a similarity measurement score between 2 structures. The GDT_TS score is used in this research as the similarity score. The goal here is to derive a QA score s_i for each a_i, 1≤i≤q so that X = {x_i, 1≤i≤q} has strongest correlation with Y [7].

In this research, Pearson correlation coefficient ρ between X and Y is used as the correlation metric to evaluate performance of our QA method:

$$\rho = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{10}$$

with -1 ≤ ρ ≤ 1, x and y are the mean of X and Y respectively. ρ = 1 means a prefect correlation. Now, our objective is to generate QA score X to maximizes the correlation ρ [7]:

$$\underset{X}{\operatorname{argmax}}\,\rho \tag{11}$$

The evaluation metric is the root mean square error (RMSE) between 2 distance matrices M and M'. A lower value means a better predicted target protein structure. The RMSE value is calculated as shown in (6).

$$RMSE = \sqrt{\frac{1}{|M|}\sum_{i=1}^{|M|} ||M - M'||^2} \tag{12}$$

4.2.2 System Architecture

Our method is divided into two parts: 1) feature generation from PDB data and training DCNN; and 2) QA prediction with DCNN on target protein. The term “target” refers to the protein that need to be evaluated.

4.2.2.1 Protein Vector generator

Given a target sequence, a matrix of feature represented by protein vector is generated by replacing every 3-residues of sequence with a continuous vector of 100 dimensions from ProtVec [45]. For example, given a target sequence with length n, every nearby 3-residues is replaced by a vector of 100 dimensions. Finally, a feature matrix with size (n-2)x(100) is generated.

4.2.2.2 Profile generator

Given a target sequence, a matrix of features represented by profile information is generated by first running HHpred [3] program and then apply script buildAli.pl on the generated data [3]. For example, given a target sequence with length n, a profile matrix with size (n)x(100) is generated. This profile information is a commonly used feature for many methods in protein structure prediction [4,5,6] and protein quality assessment [9,10,11].

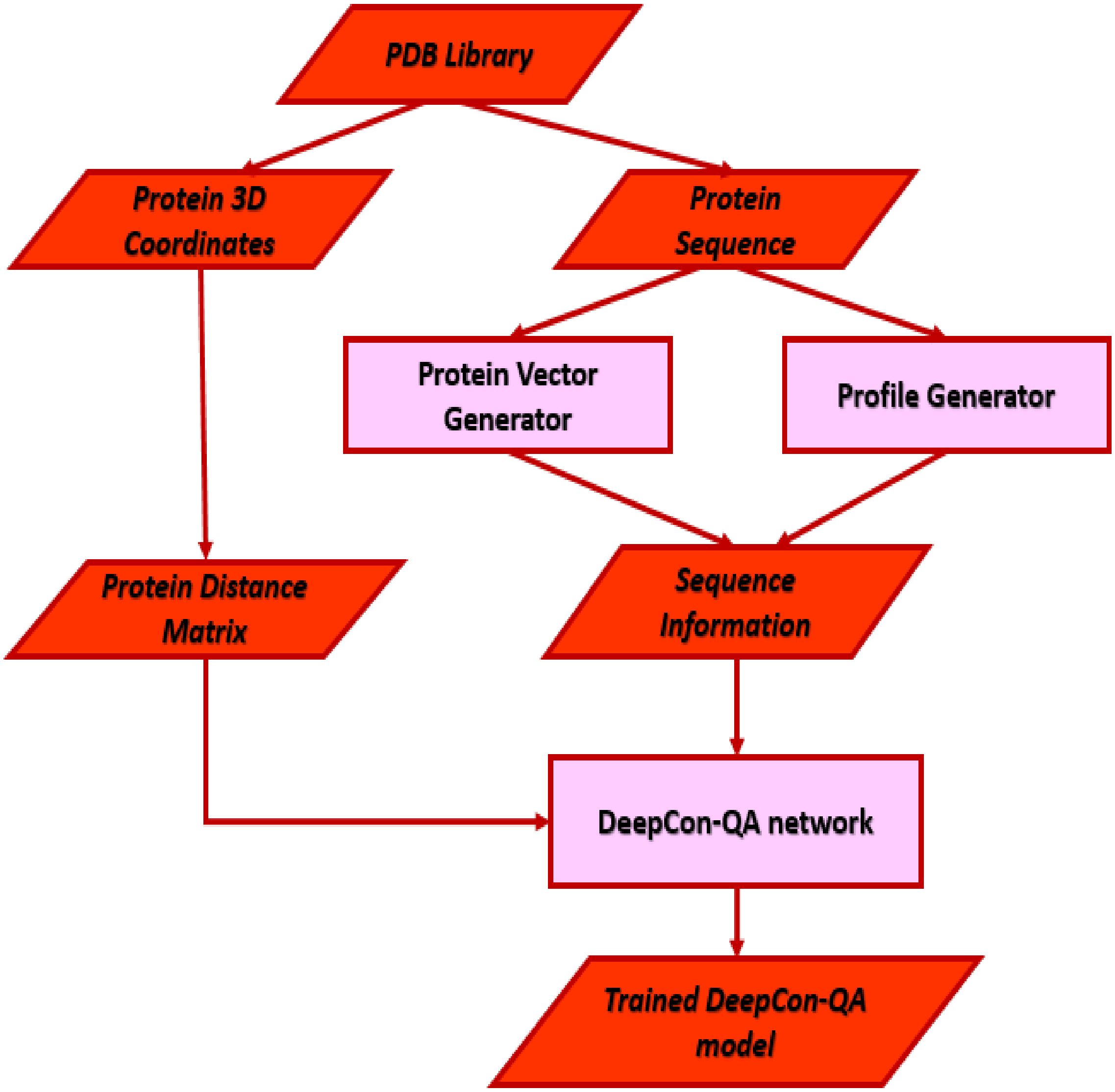


Figure 13. Training DeepCon-QA Network

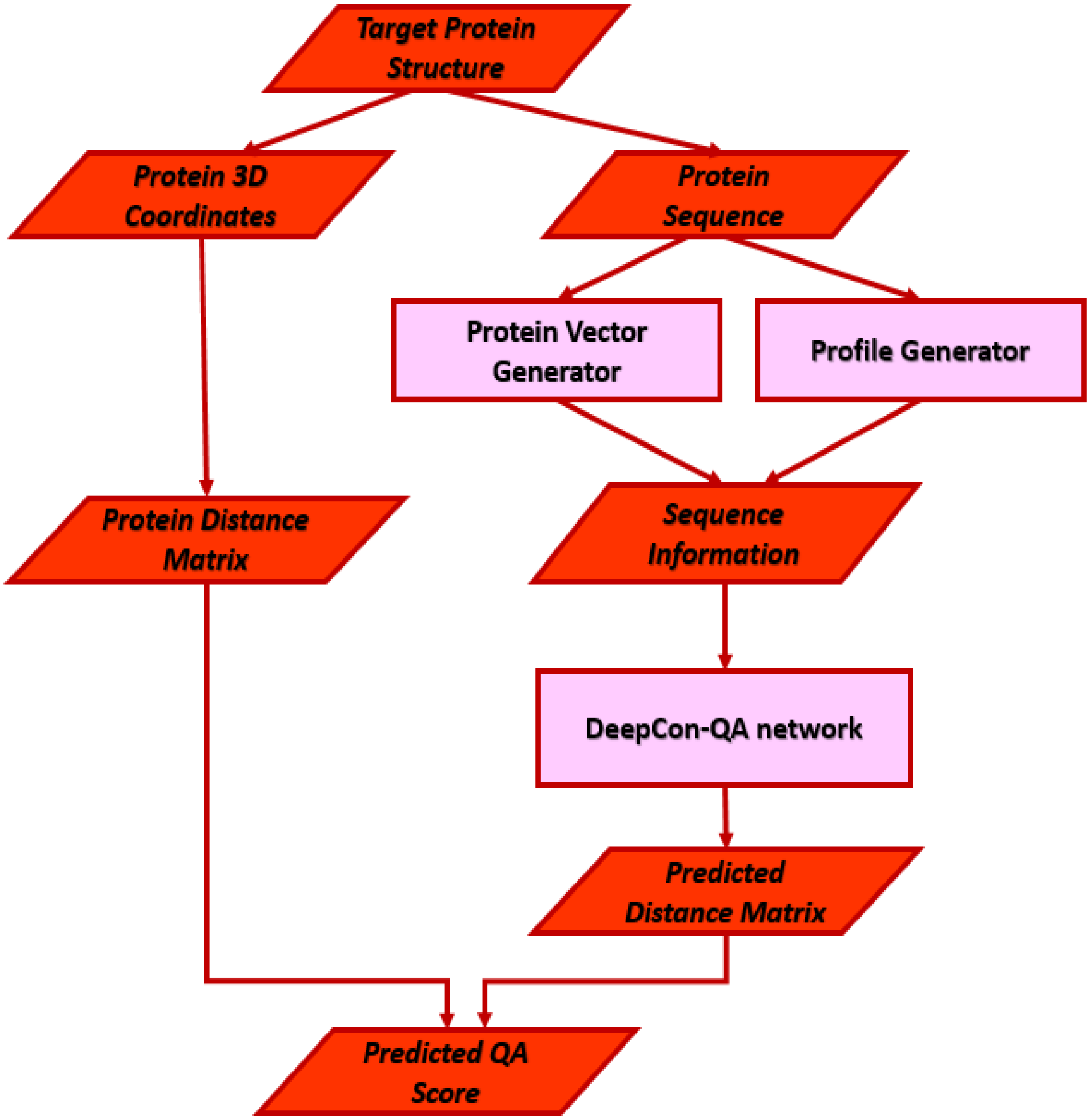


Figure 14. Predicting QA score with DeepCon-QA Network

4.2.2.3 DeepCon-QA network

DeepCon-QA is a Deep Convolutional Neural Networks that takes a matrix of features generated from protein sequence information as input and then output a distance matrix that represent the structure of given protein sequence. This DCNN composed of input layer; multiple convolutional layers with each convolution layer contains a convolution operation, a Batch Normalization (BN) operation, and a rectified linear unit (ReLU) activation function; and a fully connected layer.

The loss function is created by measuring the similarity of the reconstructed matrix and the ground truth. We use the L2 distance between the predicted distance matrix of missing region and the distance matrix of ground truth. In the following equation, z is the input to the Generator Network (G), x is the ground truth.

$$Loss_{recon} = \|G(z) - x\|_2^2 \qquad (13)$$

4.2.2.4 Model Training with DeepCon-QA

Figure 13 shows the flowchart of training DeepCon-QA networks. First, proteins with known structures in PDB library [35] are taken as training set for DeepCon-QA networks. Protein sequence is extracted from each target protein in the training set. Protein vector generator and profile generator will be used on the sequence of each protein to create a combine matrix of feature. A set of these matrices will be used as input for the networks. Then, the 3D coordinates of each protein are extracted and converted into distance matrix. This set of distance matrices will be used as the expected output for training the DeepCon-QA networks. Finally, a trained DeepCon-QA model will be derived and used for QA predictions later. Multiple network structures of DeepCon-QA will be tested to determine the best network structure for the QA task.

Beside using combined features extracted from protein vector generator and profile generator, DeepCon-QA is also tested with only features extracted from profile generator.

The purpose is to evaluate performance of DeepCon-QA with profile features versus with combined features.

4.2.2.5 QA Prediction with DeepConQA

Figure 14 shows the flowchart of predicting QA score with DeepCon-QA network for a target protein. Initial steps of QA prediction are similar to model training. Sequence information will be generated from Protein Vector Generator and Profile Generator which run on protein sequence. Then, that sequence information is used as input for DeepCon-QA network. It will output a predicted distance matrix given the sequence information. On the other hand, 3D coordinates of target protein are also extracted and converted into protein distance matrix. Finally, a RMSE value between predicted distance matrix and protein distance matrix is calculated. This RMSE value is the final predicted QA score from our method.

This research uses the term DeepCon-QA method as our QA method to evaluate quality of a protein structure prediction while the term DeepCon-QA network refers to the DCNN which is used to predict the target distance matrix given sequence string.

4.3 Results and Discussion

4.3.1 Dataset and experimental settings

Two datasets were used in our experiments.

The first dataset, called set PDB25, is a subset of Protein Data Bank [35] in which any two proteins share less than 25% sequence identity with each other. Proteins with length less than 50 residues are discarded. Proteins with length greater than 50 residues are cut at both side to have length of 50 residues. Totally, there are 5715 proteins selected in this set PDB25.

The second dataset, called set QA20, is a subset of CASP11 Stage 1 dataset [36] that was used for testing performance of QA methods with 88 targets with each target has 20 protein structure predictions. In set QA20, only targets with length less than 300 residues, single-domain and belongs to Free Modeling categories [36] are selected. Targets in Free Modeling categories are hard targets which are very hard to do predictions because not much evolutionary information was found on these ones. Totally, there are 8 targets with 20 protein predictions for each target in this set QA20. The term target here refers to a protein that need to be predicted for 3D structure given its sequence string as input.

There are 2 current state-of-the-art methods including ProQ2 [81] and MQAPmulti [82] that have been used as the baseline to compare with our newly developed methods for QA problem. These 2 methods are chosen because they have been proved to be among the best methods in the single model QA methods group from the evaluation of CASP11 competition organizer [36]. Experimental results of ProQ2 and MQAPmulti on set QA20 are acquired directly from the official repository of data of CASP11 [36].

The average computation time of DeepCon-QA on each training task is 3 CPU hours on a Linux server with Intel Xeon CPU E5-2650 v4 @ 2.20GHz, memory 26GB, graphics card NVIDIA GeForce GTX TITAN X.

DeepCon-QA is implemented based on Tensorflow [27].

4.3.2 Determining the best configuration for DeepCon-QA

In this experiment, Set PDB25 is used to find the best con-figuration for DeepCon-QA so that it can be used later on QA prediction tasks. Specifically, DeepCon-QA network is applied on set PDB25 with different number of hidden layers. Number of epochs is set at 500. For each setting of hidden layers, DeepCon-QA ran 10 times with different random initialization and the average RMSE scores are reported. Set PDB25 is divided into 3 sets: training, validation and test with number of proteins are 85%, 15% and 10% respectively on total number of proteins.

Figure 5, 6, 7 show performance of DeepCon-QA on training, validation and test set respectively. Four different networks structures are tested: 8 layers, 14 layers, 20 layers and 26 layers including input and fully connected layers. Two versions of DeepCon-QA method are evaluated with one, called "Profile", uses only profile information as input feature while the other one, called "Combine", uses both profile and protein vector representation information. The number of epoch is set at 500 for all experiments.

Figure 15 shows the average RMSE of DeepCon-QA method on all proteins of training set with Profile features only versus Combined features. The first configuration of 8 layers gives best scores (lowest) for both versions of DeepCon-QA with average RMSE of 2.15 and 1.77 for Profile and Combine versions respectively. This means the simpler network can fit the data faster than the more complicated networks which have more layers that need more epochs to fit the data.

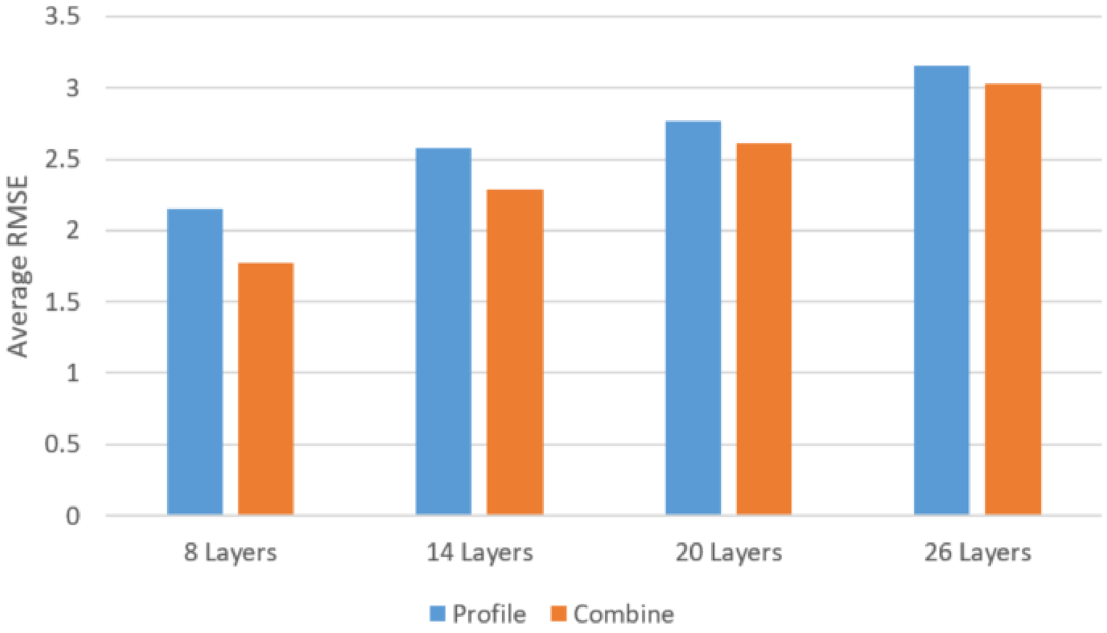


Figure 15. Training set - Average RMSE of DeepCon-QA with Profile features only versus Combined features. The experiment is executed 10 times to get average RMSE.

Figure 16 and 17 show the average RMSE of DeepCon-QA method on all proteins of validation set and test set with Profile features only versus Combined features. On validation set, both versions of DeepCon-QA achieve best performance at 14 layers with average RMSE of 3.06 and 3.08 respectively. Similarly, both versions of DeepCon-QA also achieve best performance at 14 layers on test set with average RMSE of 3.26 and 3.22 respectively. These experiments show that DeepCon-QA works best with 14 layers' configuration. Hence, this configuration will be selected to use for our task of QA prediction later.

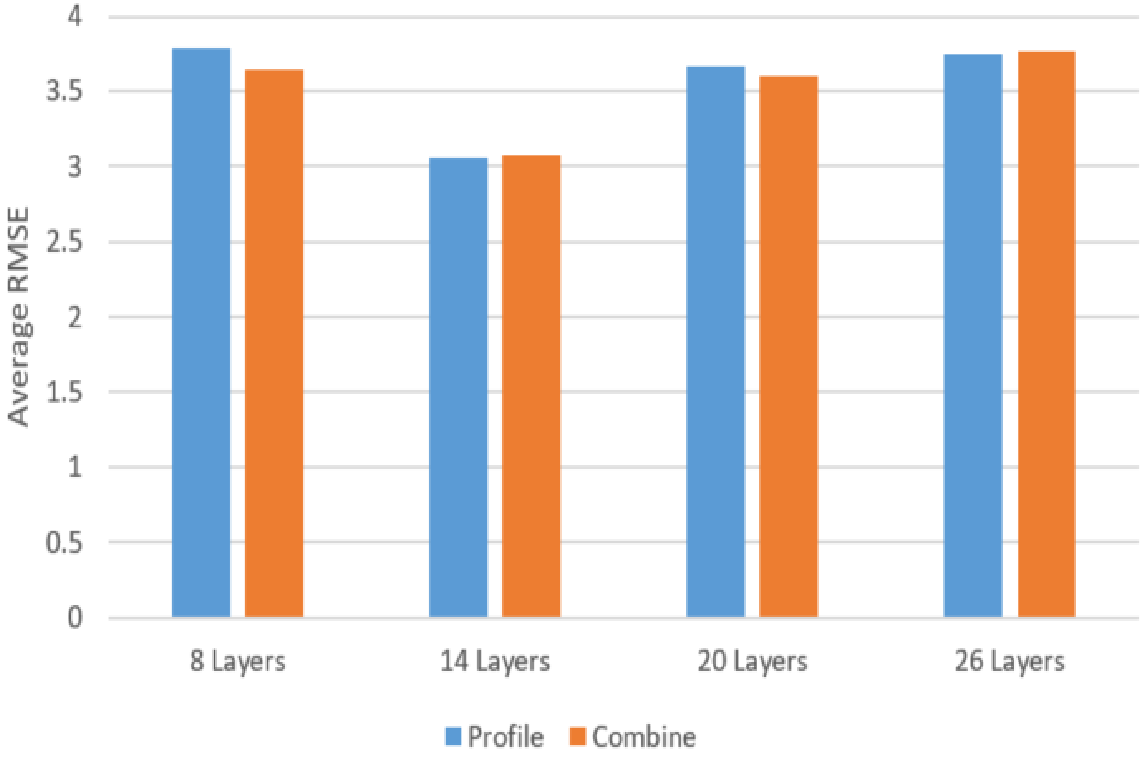


Figure 16. Validation set – Average RMSE of DeepCon-QA with Profile features only versus Combined features. The experiment is executed 10 times to get average RMSE.

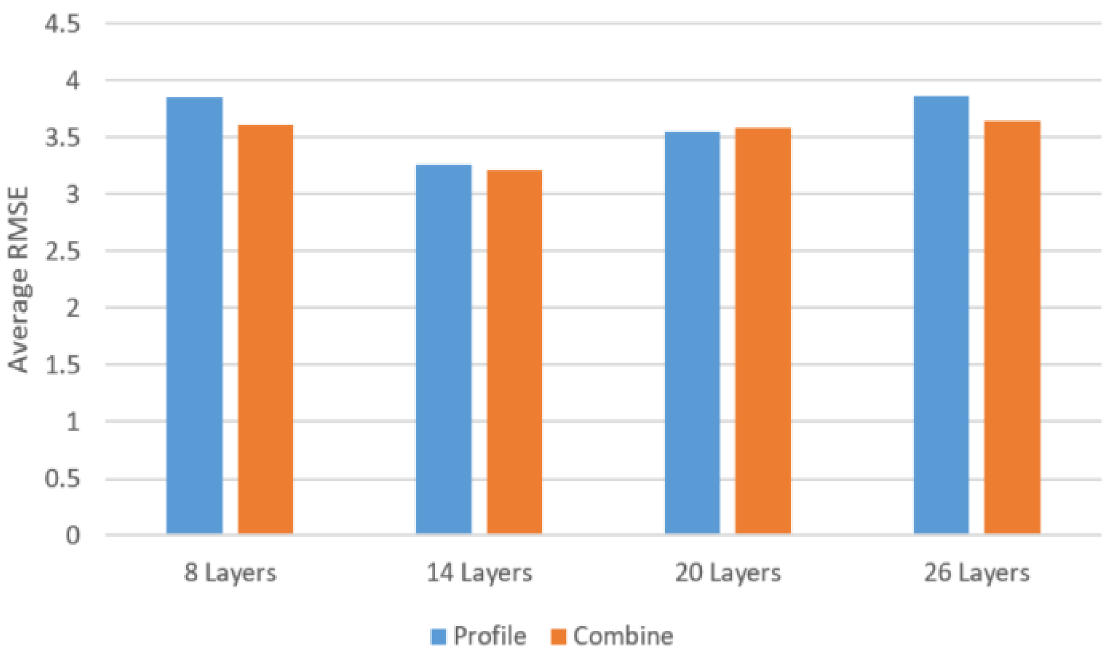
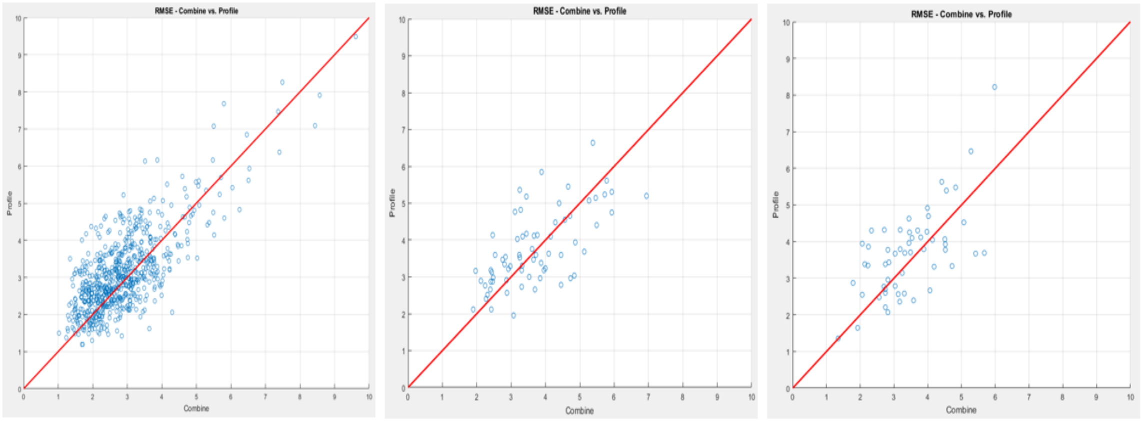


Figure 17. Test set - Average RMSE of DeepCon-QA with Profile features only versus Combined features. The experiment is executed 10 times to get average RMSE.



a) b) c)

Figure 18. Scatter plot of proteins with NEFF value smaller than 3. These proteins are hard to predict given sequence string since not much evolutionary information can be found. X axis represents RMSE score from our QA method using combined information. Y axis represents RMSE score from our QA method using only profile information. a) Proteins in training set. b) Proteins in validation set. c) Proteins in test set.

4.3.3 Comparing performance of DeepCon-QA using only profile features versus combine features

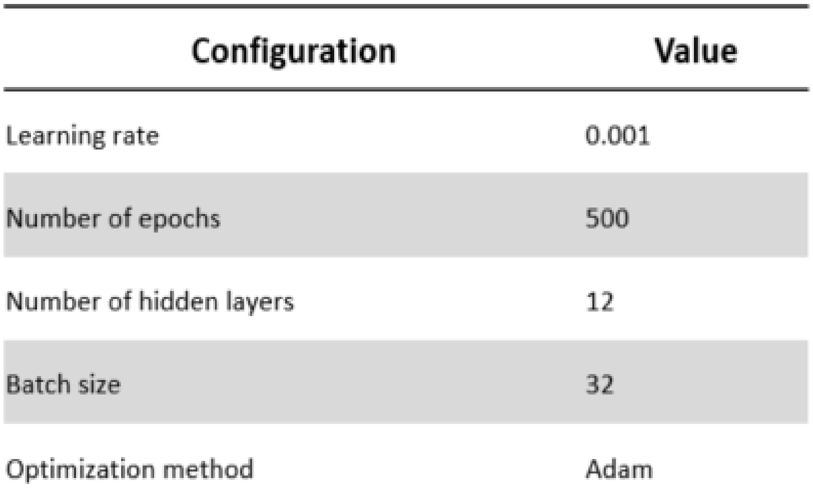
The results in figure 16 and 17 show that both versions of DeepCon-QA work best at 14 layers. Between them, the performance of both versions is similar with DeepCon-QA Profile version performs best in validation set while DeepCon-QA Combine version performs best in test set. To find out what area these two versions are different in performance, a subset of PDB25 is used. In this subset, called set PDB25-hard, all proteins with NEFF scores greater than 3 are excluded. The remaining proteins are hard target that are hard to predict with very few evolutionary information found.

Figure 18 shows the performance of DeepCon-QA with 14 layers on this subset PDB25-hard. The proteins in figure 8 (blue circle) skew to the upper triangle mean they have lower RMSE score with DeepCon-QA Combine version which are better. On the other hand, the ones skew to the lower mean they have lower RMSE score with DeepCon-QA Profile version. Generally, more proteins clustered in upper triangle means the DeepCon-QA Combine version performs better than the DeepCon-QA Profile version and vice versa.

Figure 18a shows performance on training set of subset PDB25-hard. The average RMSE of Profile version and Combine version are 3.08 and 2.84 respectively. It means the Combine version is better than Profile version in this training set. Figure 18b shows performance on validation set of subset PDB25-hard. The average RMSE of Profile version and Combine version are 3.75 and 3.68 respectively. Again, the Combine version is better than Profile version in this training set. Similarly, figure 18c shows performance on test set with average RMSE of Profile version and Combine version are 3.68 and 3.44 respectively. The Combine version significantly outperforms Profile version in this test set.

In conclusion, the results in subset PDB25-hard shows that the DeepCon-QA Combine version significantly outperforms Profile version in training, validation and test set. In the bigger set PDB25, the performance of both versions is very similar which cannot determine a dominant version. As a result, DeepCon-QA Combine version will be used as our QA method to test on another dataset. It has 14 layers with 12 hidden layers and use Combine feature. Table 10 shows detail configuration for DeepCon-QA network.

Table 10**.** Configuration of DeepCon-QA network after training.



Configuration	Value
Learning rate	0.001
Number of epochs	500
Number of hidden layers	12
Batch size	32
Optimization method	Adam

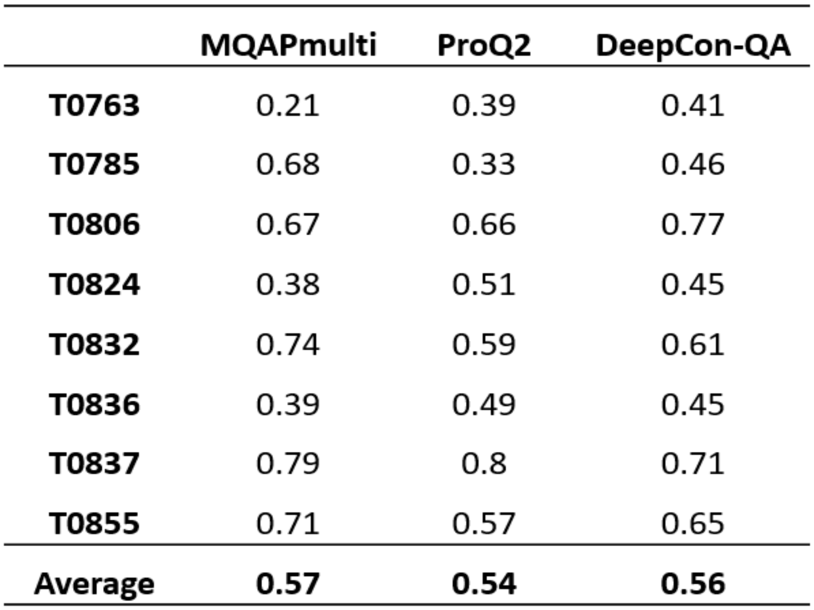
4.3.4 Performance comparison of QA methods

Table 11 shows the results of existing QA methods as well as our DeepCon-QA method on set QA20. Totally, there are 8 targets with each target has 20 protein predictions that need to get QA scores. The values shown in tables for each target is the Pearson correlation between QA scores from QA method versus true GDT_TS score between protein predictions and their ground truths.

In this experiment, since our QA method is predicting a RMSE score with lower score means better while the other QA methods that are used for comparison are predicting similarity score with higher score means better. To compare our method with other QA methods, our scores need to be on a same scale with others. Specifically, after the DeepCon-QA outputs the predicted distance matrix (in figure 3), multidimensional scaling technique [20] is used to convert this predicted distance matrix into predicted 3D structure. Similarly, the protein distance matrix in figure 3 will be converted into protein 3D structure. Finally, the similarity score GDT_TS between these 2 structures will be calculated as our QA score. This QA score scale well and is comparable with other methods' QA scores.

From the results of table 11, MQAPmulti method is the best QA method with average of correlation is 0.57. Our DeepCon-QA method is the second best one with average of correlation is 0.56. The last one is ProQ2 method. Specifically, our method achieves best results over other methods on 2 targets T0763 and T0806. Individually, our method does outperform the best method MQAPmulti in 50% of cases with 4 targets: T0763, T0806, T0824 and T0836. Even though ProQ2 method has the lowest average score, it does outperform other methods very well on 2 targets: T0824 and T0837.

Table 11. Pearson correlation coefficient values of our DeepCon-QA method and other QA methods on dataset QA20.



	MQAPmulti	ProQ2	DeepCon-QA
T0763	0.21	0.39	0.41
T0785	0.68	0.33	0.46
T0806	0.67	0.66	0.77
T0824	0.38	0.51	0.45
T0832	0.74	0.59	0.61
T0836	0.39	0.49	0.45
T0837	0.79	0.8	0.71
T0855	0.71	0.57	0.65
Average	**0.57**	**0.54**	**0.56**

4.4 Conclusion

This research presents a new protein quality assessment method DeepCon-QA based on deep learning (DCNN) and distance matrix representation of 3-D structures. From the rotation and translation independent representation of distance matrix, DCNN is able to learn a mapping function to predict an expected distance matrix that can be used later for evaluating quality of protein structure prediction. To the best of our knowledge, this is the first attempt to combine deep learning, geometric information and protein vector representation for protein quality assessment problem. Moreover, this research also shows that protein vector representation is a very useful set of features that can be used for QA tasks compare with profile features which are a very commonly used in many QA methods.

It opens new promising research directions to solve other problems in protein prediction area.

Experiments on common benchmark dataset CASP11 of hard targets have shown promising results. Compared to the best QA method MQAPmulti, our DeepCon-QA method achieves equal result when comparing in percentage of number of cases while being comparable when comparing in the average value. DeepCon-QA does perform best in some cases such as T0763 and T0806.

Due to the limitation of computing power and time, experiments have been done on a limited sets of different network configurations. Exploring more network configurations and combine more features with other evolutionary information can help improve the overall performance of QA tasks. This would be a good direction for future work.

CHAPTER 5. NEW DEEP LEARNING METHODS FOR PROTEIN LOOP MODELING

5.1 Introduction

Knowledge of three-dimensional (3D) structure of a protein is critical for understanding its function, mutagenesis experiments and drug development. Some experimental methods such as the X-ray crystallography or Nuclear Magnetic Resonance (NMR) can determine a good 3D structure but they are very time consuming and expensive [1]. To address those limitations, many protein structure prediction methods have been developed. Due to its ability to predict protein structure fast and accurately, template-based protein structure prediction is being used in many biological applications [6,46,47,48]. This approach is mainly based on the idea of finding good template in the Protein Data Bank (PDB) [49] with a high level of sequence similarity of a query protein [50]. In many cases, parts of the protein structure cannot be predicted due to no alignment to known structures for these parts. Thus, it is necessary to predict the structure of these missing parts to complete the final protein structure prediction. This is called the loop modeling problem. In some experimental structures, loop modeling is also needed due to lack of structural information in certain parts of the protein.

Loop modeling methods can be divided into two main approaches: ab initio and template-based approaches. The ab initio approach is based on the idea of using biological and physical properties of the loop region in the protein to determine the best loop conformation. Some ab initio methods such as LOOPY [51], PLOP [52], and MODELLER [53] can predict the loops with short length (below seven) quite well [6]. On the other hand, the template-based approaches, such as LIP [54], NGK [55], Galaxy PS1 [56], or Galaxy PS2 [57] tries to derive some the loop structure candidates and possibly combine with some energy functions or selection methods to finish the loop modeling task.

The methods mentioned above are different from one another in details, but most of them still share a general framework that starts with sampling a large number of feasible conformations.

Among ab initio methods, the LOOPY algorithm is based on the random tweak algorithm [58] and carries out loop closure while avoiding steric clashes. MODELLER constructs and samples loop conformations with a bond-scaling and relaxation method. Basically, it combines conjugate gradient minimization and molecular dynamics with simulated annealing [6]. On the other hand, PLOP uses a systematic dihedral angle based build-up process to sample conformation space.

Basically, the template-based methods try to extract loop structures from experimental structural databases such as the Protein Data Bank (PDB) first. Then, further sampling and optimization methods are applied on the extracted loop structures to come up with a good set of candidate structures. Finally, a ranking method such as energy or scoring function is employed to get the best loop structure [50].

Among template-based methods, NGK performs sampling for loop conformation based on the idea of combining intensification of torsion and parameter annealing strategies. Both Galaxy PS1 and Galaxy PS2 are loop refinement methods that starts with an inaccurate loop structure. The energy of Galaxy PS1 is optimized for application to the refinement of template-based models, while Galaxy PS2 is developed for higher performance for the near-native models as well [56].

In this research, five novel methods are proposed for loop modeling, called NearLooper, ConLooper, HyLooper1, ResLooper and HyLooper2. The NearLooper method is based on the idea of selecting loop templates with a similar structure of surrounding environment. ConLooper uses Convolutional Neural Networks (CNN) [59-63] to predict candidate loop template. By combining the results of NearLooper and ConLooper, the HyLooper1 method takes advantages of both methods. ResLooper uses Residual Networks to predict candidate loop template. And HyLooper2 method is the combination of NearLooper and ResLooper. ConLooper and ResLooper use distance matrix that contains pairwise distances between two residues' C α atoms in the model as an orientation-independent representation of protein 3D structure. Experiment results show that these methods achieve comparable results to other state-of-the-art methods and outperform them in some cases.

5.2 Methods

5.2.1 Problem formulation

The loop modeling problem is formulated as a partial protein structure construction in this research: given an incomplete protein structure that has a gap region (also called loop) with unknown structure coordinates inside, predict the conformation (structure coordinates) of the gap region. It is indeed the missing value problem in machine learning which can be defined as: given an input N-dimensional matrix with some areas of miss-ing values, predict an output N-dimensional matrix so that the areas of missing values are filled in. Chao et. al. [65] is a good example of this. As an example, Figure 20(a) shows a protein structure with a gap in the middle and 20(b) shows the result of a complete structure with the gap region predicted.

Let S = {s_i, $1 \leq i \leq n$} be the amino acid sequence of a target protein of length n, and A = {a_i, $1 \leq i \leq n$} be its incomplete 3D structure, where a_i represents the 3-D coordinates of the ith amino acid, that contains a gap region between two indices u and v, $1 < u \leq i \leq v < n$. Suppose a predicted model of the gap region is M = {m_i, $u \leq i \leq v$} and the truth structure of the gap region is D= {d_i, $u \leq i \leq v$}, the goal is to find an M given A so that the root-mean-square deviation (RMSD) between M and D is minimized.

To calculate RMSD between M and D, which measures the structural similarity between two 3-D structures of the same length, M is first optimally superimposed over D. Then, the root-mean-square deviation (RMSD) is calculated using the coordinates of the corresponding main chain atoms (N, Cα, C and O) in the two structures as follows:

$$\text{RMSD} = \sqrt{\frac{1}{k}\sum_{i=1}^{k}||m_i - d_i||^2} \qquad (14)$$

where k is the number of pairs of corresponding atoms.

5.2.2 New methods for Protein Loop modeling

In this section, five new methods are presented for loop modeling, i.e., finding M. They are NearLooper, ConLooper, ResLooper, Hylooper1 and HyLooper2. Figure 19 shows the framework of the five methods that share the following common pre-processing and post-processing steps:

(1) Generate subsequence. A subsequence of the target protein with a gap region is extracted from the original sequence. This subsequence is selected so that it is longer than the gap region and contains the gap region at the center. As an example, Figure 2(c) shows an input sequence with a gap region LVKEQWIL and 2(d) shows the selected subsequence of length 50 that contains the gap region in the middle.

(2) Generate candidate template bank. Here, the term “template” is defined as a protein with complete 3D structure (including the loop region) that was found from alignment tools. A pool of templates (candidate 3D structures) for the selected subsequence are generated using sequence-based alignment tools on a protein structure database (e.g. PDB), such as PSI-BLAST [16] and HHSearch [17]. This pool of templates will be used later to train models.

(3) Loop modeling. The five different methods generate loop models using the pool of templates in different ways.

(4) Superimpose the generated loop model over the target protein structure based on their overlapping regions.

(5) The gap region in the target protein structure is filled in with coordinates from the superimposed loop model. As an example, Figure 2(b) shows the result for input 2(a).

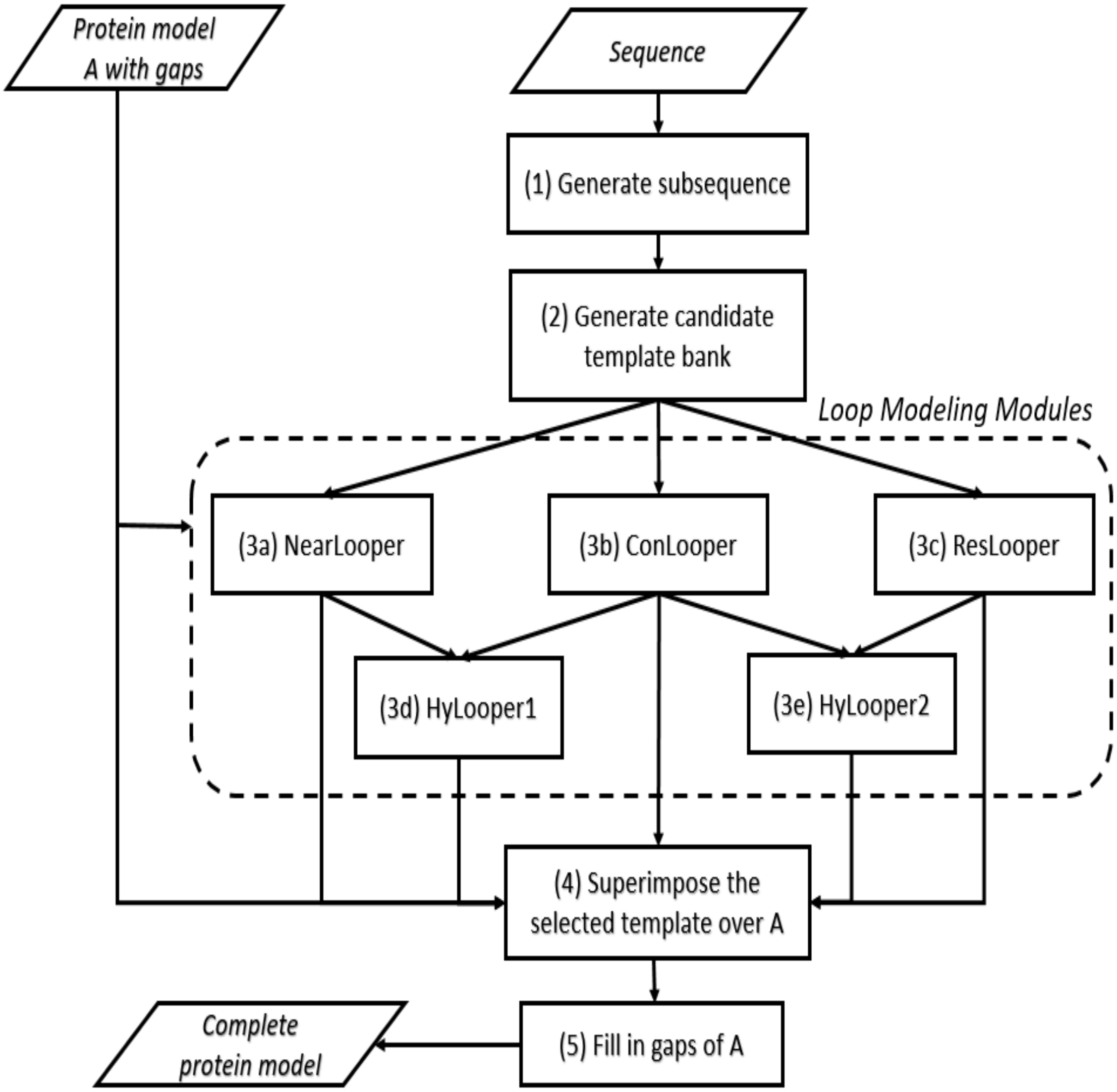
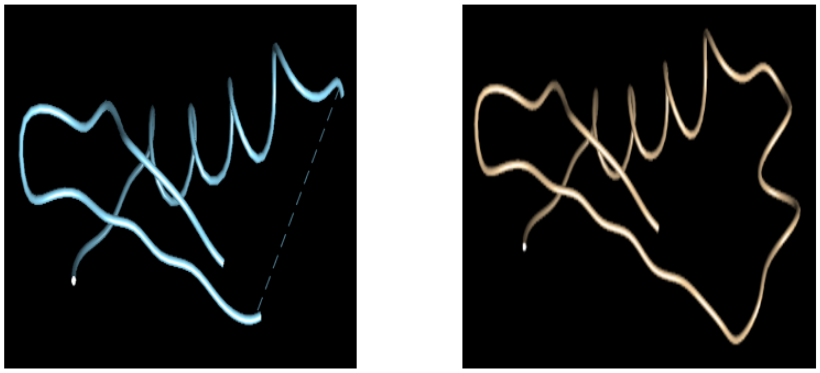


Figure 19. The framework of the five proposed loop modeling methods, NearLooper, ConLooper, HyLooper1, ResLooper and HyLooper2.



a) b)

DQRRSKLRVVGGHPGNSPWTVSLRNRQGQHFCGGS***LVKEQWIL***TARQCFS
SCHMPLTGYEVWLGTLFQNPQHGEPSLQRVPVAKMVC

c)

GNSPWTVSLRNRQGQHFCGGS***LVKEQWIL***TARQCFSSCHMPLTGYEVWLG

d)

Figure 20. An example of the loop modeling problem. a) A protein 3D structure with a gap region in the middle; b) The protein 3D structure with the gap region filled in; c) The original protein sequence with the gap region in the middle marked as bold and italic text; d) An extracted subsequence of length 50 from the original sequence that contains the gap region.

5.2.2.1 NearLooper, a nearest neighbor algorithm for loop modeling

NearLooper first finds a pool of templates for an extended loop region (the loop region extended on both sides) using sequence-based search from a protein structure database, and select from them a template whose structure optimally matches with the target structure (A) in their overlapping region. Then, the selected template is superimposed over the original incomplete structure A based on their overlapping region (on both sides of the gap region) and the coordinates of the template in the gap region is outputted as the predicted loop model M, which can be inserted into A to make A complete.

Following the framework in Figure 1, NearLooper uses the nearest neighbor idea to select one template from the template pool in step (3). To determine the similarity between different templates and the target structure, the RMSD value of the overlapping region between a template and the target protein is used as the selection metric: the template with the lowest RMSD is selected.

5.2.2.2 ConLooper, a novel deep learning method for loop modeling

In this method, a novel idea of applying CNN on the distance matrix representation of 3-D structures is employed for predicting loop conformation instead of using the 3-D coordinates of protein directly. A protein that is rotated in space will have many different sets of values of coordinates. That leads to difficulty for machine learning methods to "learn" the data. On the other hand, distance matrix is an orientation-independent representation of protein structure that eliminates the problems of rotated proteins.

In recently years, deep learning based on deep neural networks (DNN) has had a huge impact on computer science and achieved unprecedented performance on many machine learning problems. Deep-learning methods take raw data to automatically discover the multiple layers of representations needed for detection or classification. A key aspect of deep learning is that these layers of representation are not designed by human engineers, but learned from data. Deep convolutional neural networks (DCNN) have made breakthroughs in processing audio, image, video, and playing games. DCNNs are made up of neurons that have learnable weights and biases. Each neuron takes some inputs, perform some calculations and produce an output. DCNNs typically have layered structures including convolutional (CONV) layer, rectify linear units (RELU) layer [65], pooling (POOL) layer and fully connected (FC) layer [39,66].

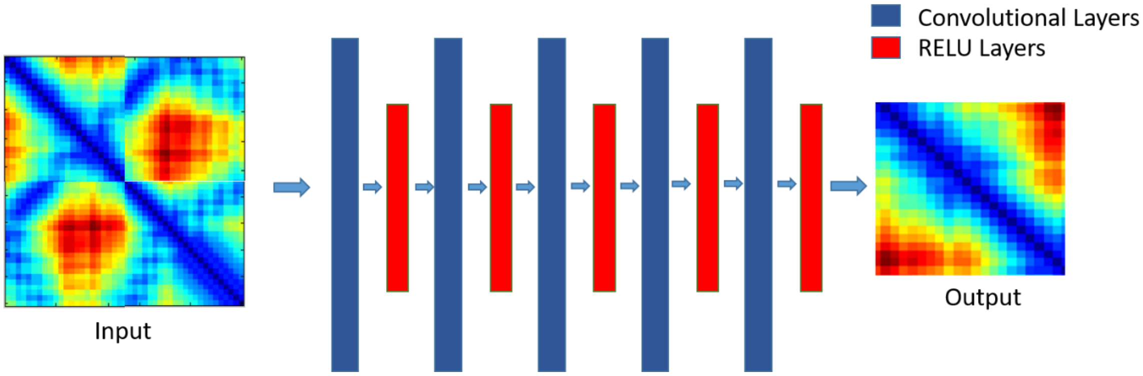


Figure 21. DCNN architecture of the ConLooper method.

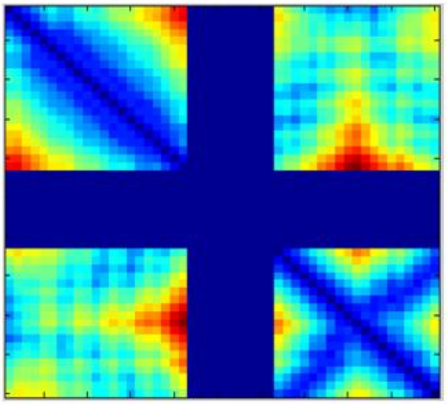


Figure 22. An example of the distance matrix of a protein structure containing a gap region in the middle (the blue bands), the ConLooper method is used to predict the gap region's 3-D structure.

The DCNN architecture of the ConLooper method is shown in Figure 21: taking the input of a distance matrix representation of a 3-D structure, the network applies several convolutional layers followed by RELU layer alternatively and finally generates a predicted distance matrix as the output. The DCNN is trained to learn a mapping from an input distance matrix to an output distance matrix.

In this network architecture, there are no pooling or dense layers used since the network is trying to do a regression task to map input distance matrix to output distance matrix instead of trying to do a classification prediction as in other classical DCNN models. If dense layers are used at the end of DCNN network, the output would be a vector of features instead of a 2D matrix as the expected output. Thus, the final layer of DCNN should be just a convolutional layer which represented by a 2D matrix as the expected output. The key operation of our DCNN model is Convolutional layers where it convolutes the input distance matrix to map it to another output distance matrix. This type of DCNN model has been shown that it can perform the regression task efficiently given the condition of input and output are 2D matrix. A reference of this can be found from the work of Oxford Visual Geometry Group [67].

Following the framework in Figure 19, in step 3, ConLooper first trains a DCNN using the pool of templates and then predict a loop model using the trained DCNN based the partial structure of the target protein structure. As an example, Figure 22 shows the input, a distance matrix with a gap region, for a DCNN to predict the gap region's 3-D structure.

Figure 23 shows a training example of the DCNN. Given a template found by some 3rd party alignment tools, the input and output of the DCNN, both distance matrices, are derived from the template's distance matrix. The goal is to learn the mapping from the context structure of the gap region to the structure of the gap region.

Details of ConLooper are as follow:

In training, the candidate templates are first converted into distance matrices using Equation (8) and normalized. Then, each distance matrix generates a training example, a pair of input and output sub-distance matrix, in the way illustrated in Figure 6. Finally, a DCNN is trained using the train examples, i.e., learning its paramaters.

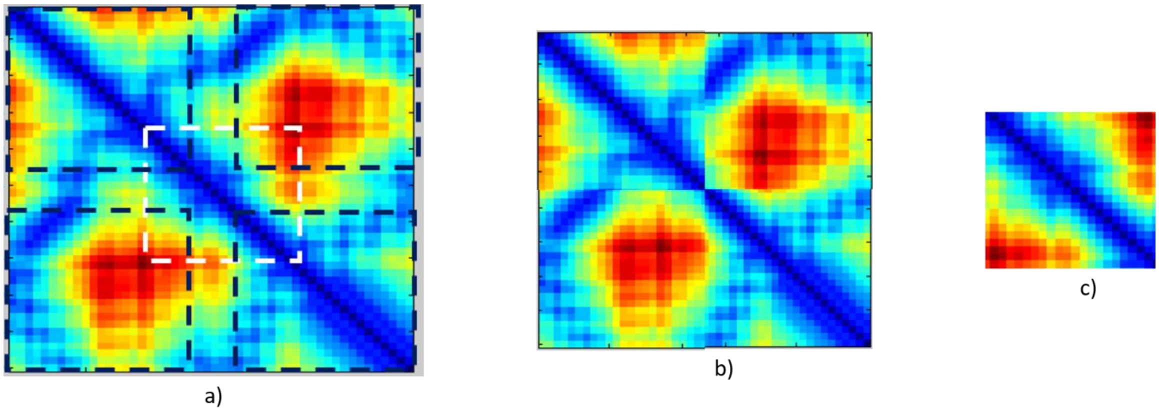


Figure 23. A training example of the DCNN in ConLooper. a) The distance matrix of a template found by a 3rd party alignment tool. b) Input distance matrix for DCNN. It is a combination of 4 corner squares marked as black boxes in a). The boundaries of these black box-es are derived from the gap position in the structure. c) Output distance matrix for CNN. It is created from the white box in a).

In prediction, coordinates of Cα atoms in the original target protein structure are extracted. Then, its sub-structure corresponding to the templates is cropped out. Note that just like the templates, this sub-structure contains the gap region, but are bigger. Next, the distance matrix of the sub-structure is calculated and normalized using the normalization parameter calculated in the training phase, while any distance involving an atom in the gap region is set to 0. Figure 22 shows an example.

The area of non-gap region is used as the input to the trained DCNN. The output of the DCNN is a distance matrix for the extended gap region (the gap region with some extra amino acid on both sides), similar to the one in Figure 23(c). The output distance matrix is unnormalized and converted into a 3-D structure (Cα atoms only) using the multidimensional scaling method. Finally, Pulchra [68] is used to generate the full atom model from the predicted Cα model.

The full atom model of the extended gap region predicted by the DCNN will be used in the step 4 of the framework in Figure 19. The model is superimposed over the target protein structure based on their overlapping region.

5.2.2.3 ResLooper, another deep learning method for loop modeling

Similar to ConLooper, ResLooper uses the same idea of applying deep learning method for predicting output loop conformation. Specifically, ResLooper uses Resid-ualdue Neural Networks [69] to predict loop confor-mation instead of CNN as in ConLooper.

The Residual Neural Network (ResNet) was introduced recently in the ImageNet competition and the results were quite impressive. The basic structure of this network architecture is the building block, in which there are stacked activation layers and convolutional layers. If the input of a building block is X_l, then the output of this block is X_l+X_(l+1), in which X_(l+1) is the non-linear transformation of X_l. We define a function F indicating the difference between the input and output of the building block, then F is called residual function. This feature of the network gives it the ability to learn something different from what the input has already encoded. Also, this network handles the vanishing gradient problem well.

In our network, we use the pre-activation [70] configuration of the building block, as shown in Figure 24. We have two activation layers and two convolutional layers. We use ReLU as the activation function.

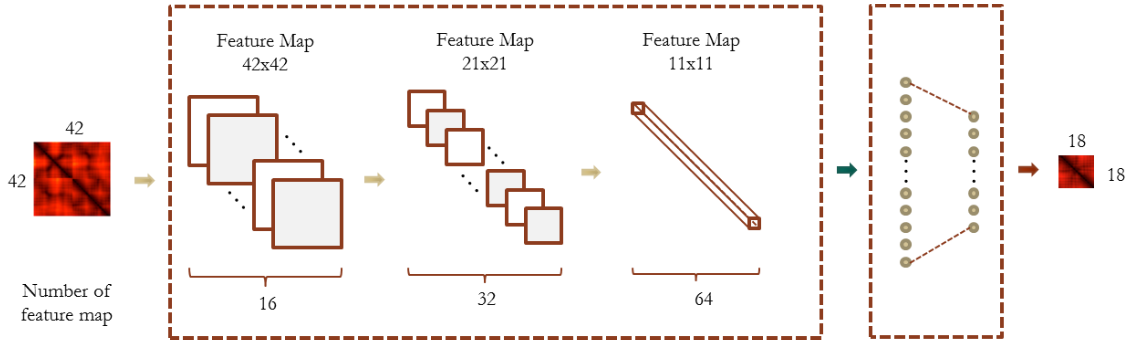


Figure 24. ResNet architecture of the ResLooper method.

The ResLooper network architecture is shown in Figure 25. The input and output format of the ResLooper is the same as ConLooper, with the input as a 42 by 42 distance matrix and output as an 18 by 18 distance matrix. The network contains 9 residual blocks, plus the first initial convolutional layer and the last fully connected layer. There are 20 weighted layers totally in our network. The first initial convolutional layer has a kernel size of 3 and the number of filters is 16. Then there are 9 residual blocks and we divided them into 3 parts, each part has 3 residual blocks. Between each part, we use stride 2 to perform down sampling, which results in the feature map sizes {42, 21, 11}, respectively. Then number of filters are {16, 32, 64}, respectively for each part. The kernel size is 3 for all convolutions. After the final fully connected layer, we reshape the output vector to a distance matrix.

In the building block, for the shortcut connection, there are two cases. For the case that the depth dimension changes, we use 1 by 1 convolution to match the dimension. For the case that the depth dimension does not change, we use identity connection.

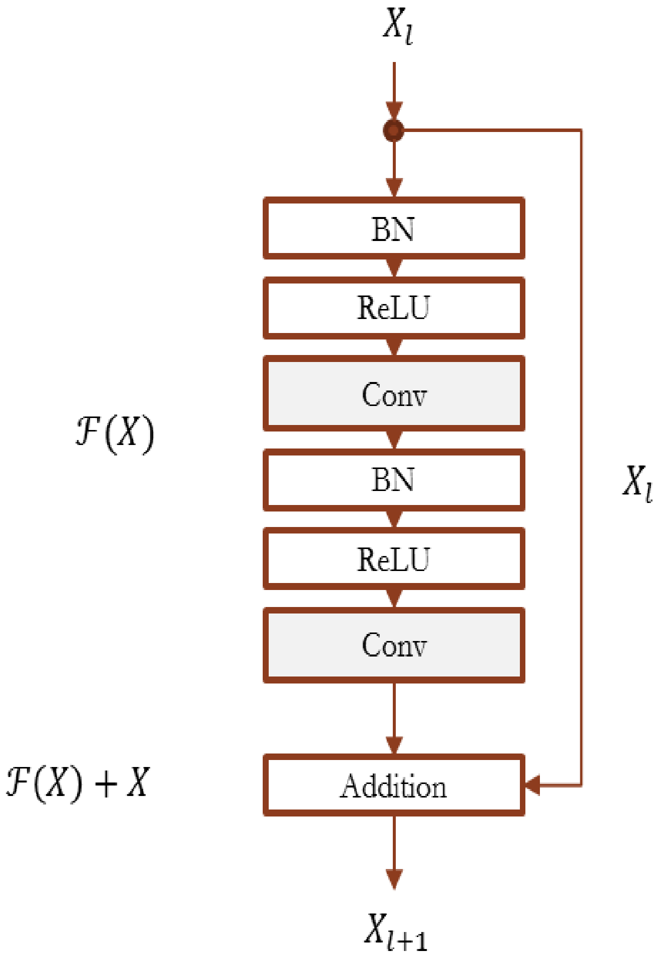


Figure 25. Configuration of the pre-activation building block in our residual neural network.

5.2.2.4 HyLooper1 and HyLooper2, hybrid methods for loop modeling

The HyLooper1 method combines the results of Near-Looper and ConLooper as follows:

1) Run NearLooper to output one candidate model, T1.

2) Run ConLooper to output another candidate model, T2.

3) Select between T1 and T2 based on the RMSD value of the overlapping region between each and the tar-get protein structure. The one with smaller RMSD is selected as the model for the extended gap region.

4) Continue steps 4 and 5 of the framework in Figure 1.

Basically, HyLooper1 is a hybrid method, aiming to improve over the two individual methods NearLooper and ConLooper.

The HyLooper2 method is very similar to HyLooper1 method but it uses ResLooper instead of ConLooper together with NearLooper to solve the loop modeling problem. HyLooper2 also follows the 4 steps as Hy-Looper1 but the method ConLooper in step 2 is replaced by ResLooper.

5.3 Results and discussion

5.3.1 Dataset and experimental setting

Four datasets were used in our experiments: The first dataset, called Set CASP, is chosen from The Critical Assessment of protein Structure Prediction experiments year 2014 (CASP11) [71]. This is a biennial world-wide event in the protein structure prediction community to assess state-of-the-art protein structure prediction methods. More specifically, the protein sequence search program (HHSearch) is used to search for possible candidate templates of target protein sequences in CASP11. The tool is configured to remove native-like protein from its results. The top template with its known 3D structure from the Protein Data Bank (PDB) dataset will be added to the Set CASP dataset if it satisfies the following criteria: (1) has a gap of length between 5 and 12; and (2) both sides of the gap can be extended to have a total length of 50 residues. In total, 9 proteins were selected for this training dataset.

The two other datasets from public data of Park et al. [57] are common benchmark sets used in the loop modeling community for testing performance of loop modeling methods. It composed of two subsets of 20 8-residues loops and 20 12-residues, called Set 8Res and Set 12Res, respectively. To create these sets, the first 2-ns molecular dynamics (MD) simulations at 300 K was performed, starting from the energy-minimized crystal structures using the AMER12 package [72]. Then, the Generalized Born/Surface Area (GB/SA) implicit solvation model [74, 75] and the AMBER99SB force field [73] were used [57].

The last dataset is called TBM set. It is comprised of 22 proteins with loops in more inaccurate environment of template-based models with the loop length varies from 6-11 residues. This dataset is also derived directly from the publication of Park et. al. [57] which is the template-based model set with 23 proteins. One exception is that the protein 1DF6A is removed from this set since its length is shorter than 50 residues. It was created by running MODELLER 9.6 [2] with the templates and multiple sequence alignments taken from the SALIGN benchmark study [76,77] on the HOMSTRAD set [78]. All targets for which the template-based models have GDTTS smaller than 70 or greater than 90 are excluded. The model consensus method for detecting unreliable modeled regions were used to select the target loop regions [57].

Totally, there are 71 proteins from 4 sets of data that have been used in our experiments. The loop length varies from 6 to 12 residues. The first dataset CASP is used for model selection while the other 3 datasets including set 8Res, 12Res and TBM are used for testing.

The training and validation are done for each protein of loop modeling. That means given a protein that is needed to do loop modeling, first a set of candidate templates will be created from Protein Data Bank by using 2 alignment tools PSI-BLAST [54] and HHSearch [55] on the sequence of given protein. All native-like templates in this set of templates are removed to make sure that there is no ground-truth data included in the training phase. The number of templates varies from 250 minimum to 617 maximum and 415 on average. Each template could be seen as a candidate to fill in the loop region of the protein. Then a deep learning method will be applied to train on the template set. The training/validation ratio is 75/25 on DCNN model and 90/10 on ResNet model.

Root-mean-square deviation (RMSD) of the main chain atoms N, $C\alpha$, C and O is used as the metric to compare performances between different loop modeling methods. These atoms are used since they can be seen as the representative atoms of the protein and can be used to recover to full atom model. In our experiments, our methods are compared with existing state-of-the-art methods with the results published on Park et. al [57]. These results use N, alpha, C and O atoms for RMSD measurement. Hence, we use the same set of atoms to make it comparable between our methods and other existing methods. This set of atoms has been used widely in different publications [55,56,57] for loop modeling problems. Its formula is defined in Eq. (1).

There are 3 current state-of-the-art methods including NGK [55], Galaxy PS1 [56] and Galaxy PS2 [57] that have been used as the baseline to compare with our newly developed methods for loop modeling problem. Experimental results of NGK, Galaxy PS1 and Galaxy PS2 are acquired directly from the published results of Park et. al. [57].

The average computation time of ConLooper on each loop modeling task is 1 CPU hours on a desktop computer with Intel Core i7, memory 16GB, graphics card NVIDIA GeForce GTX 980. NearLooper takes 0.5 CPU hour while ResLooper takes 1.5 CPU hours. HyLooper1 and HyLooper2 take 1.75 CPU hours and 2 CPU hours respectively.

MatConvNet toolbox version 1.0-beta20 is used as the DCNN implementation [67]. ResNet is implemented based on Keras [79] with Tensorflow [80] backend. The length of subsequence that is used for alignment is set to 50 while the size of overlapping region is set to 5.

5.3.2 Determining the best configuration for ConLooper

In this experiment, Set CASP is used to find the best configuration for ConLooper method so that it can be used later on the other datasets. Specifically, ConLooper method is applied on Set CASP with different number of hidden layers and different number of epochs. For each setting of hidden layers and epochs, ConLooper ran 10 times with different random initialization and the average RMSD and standard deviation on each of 9 proteins in Set CASP are reported.

Table 12. ConLooper with different number of hidden layers on set CASP

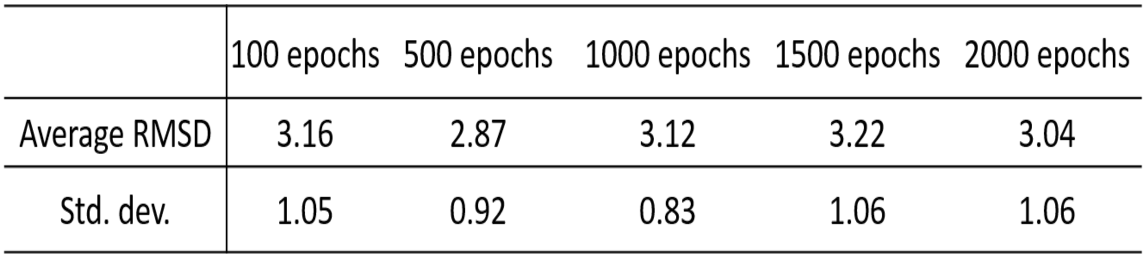
	2 Layers	6 Layers	10 Layers	14 Layers
Average RMSD	3.03	3.09	2.87	3.17
Std. dev.	1.18	0.85	0.92	1.14

Average RMSD and standard deviation are calculated on set CASP with different configurations.

Table 12 shows performance of ConLooper method on different number of hidden layers. Four different settings are tested including 2 layers, 6 layers, 10 layers and 14 layers. The result shows that the setting of 10 layers is the best, with average RMSD of the main chain atoms 2.87 and standard deviation 0.92. In this experiment, the number of epochs in training is set to 500.

Table 13 shows performance of ConLooper on different number of epochs with the number of hidden layers set as 10. Four different settings are tested, including 100 epochs, 500 epochs, 1000 epochs, 1500 epochs and 200 epochs. The result shows that the setting of 500 epochs is the best, with average RMSD of the main chain atoms 2.87 and standard deviation 0.92. Other settings either underfit or overfit the training data. Increasing from 1500 epochs to 2000 epochs seems to improve the RMSD and further increasing number of epoch could help. But due to resource limitation, the max number of epochs is set to 2000.

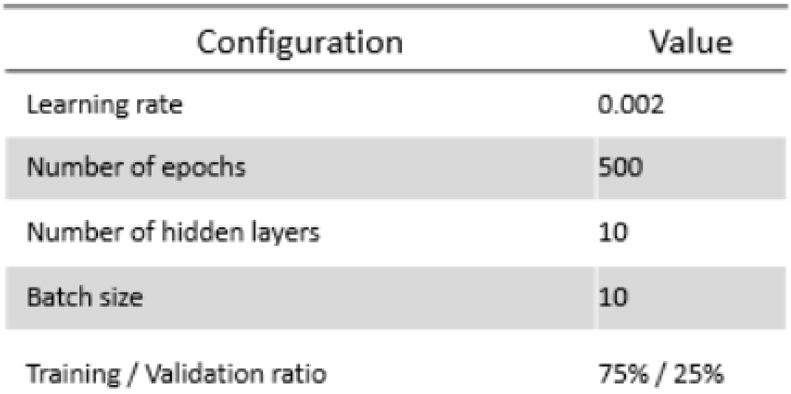
Table 13. ConLooper with different number of epochs on set CASP



	100 epochs	500 epochs	1000 epochs	1500 epochs	2000 epochs
Average RMSD	3.16	2.87	3.12	3.22	3.04
Std. dev.	1.05	0.92	0.83	1.06	1.06

Average RMSD and standard deviation are calculated on set CASP with different configurations.

Table 14. Configuration of CNN in ConLooper method after training on set CASP



Configuration	Value
Learning rate	0.002
Number of epochs	500
Number of hidden layers	10
Batch size	10
Training / Validation ratio	75% / 25%

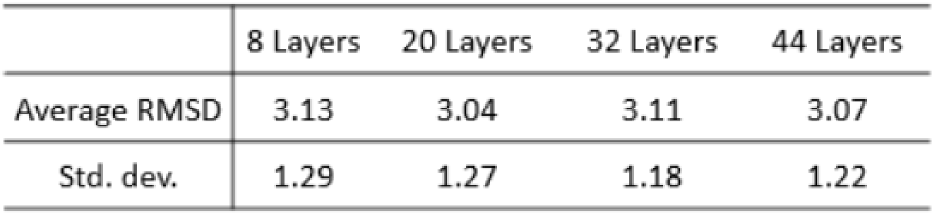
Table 14 shows the final configuration of CNN in ConLooper derived from the Set CASP with 9 proteins. The number of epochs is 500 while the number of hidden layers is 10. This configuration is used by ConLooper when compared with other loop modeling methods on other datasets.

5.3.3 Determining the best configuration for ResLooper

In this experiment, Set CASP is used to find the best configuration for ResLooper method so that it can be used later on the other test datasets. Specifically, ResLooper method is applied on Set CASP with different number of hidden layers and different number of epochs. For each setting of hidden layers and epochs, ResLooper ran 10 times with different random initialization and the average RMSD and standard deviation on each of 9 proteins in Set CASP are reported.

Table 15 shows the performance of ResLooper method on different number of hidden layers. Four different settings are tested including 8 layers, 20 layers, 32 layers and 44 layers. The result shows that the setting of 20 layers is the best, with average RMSD of the main chain atoms 3.04 and standard deviation 1.27. In this experiment, the number of epochs in training is set to 500.

Table 15. ResLooper with different number of hidden layers on set casp

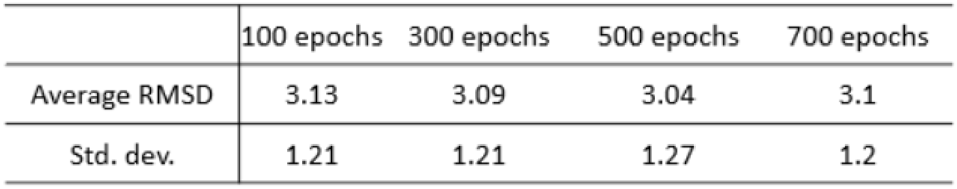


	8 Layers	20 Layers	32 Layers	44 Layers
Average RMSD	3.13	3.04	3.11	3.07
Std. dev.	1.29	1.27	1.18	1.22

Average RMSD and standard deviation are calculated on set CASP with different configurations.

Table 16 shows performance of ResLooper on different number of epochs with the number of hidden layers set as 20. Four different settings are tested, including 100 epochs, 300 epochs, 500 epochs and 700 epochs. The result shows that the setting of 500 epochs is the best, with average RMSD of the main chain atoms 3.04 Å and standard deviation 1.27. Other settings either underfit or overfit the training data.

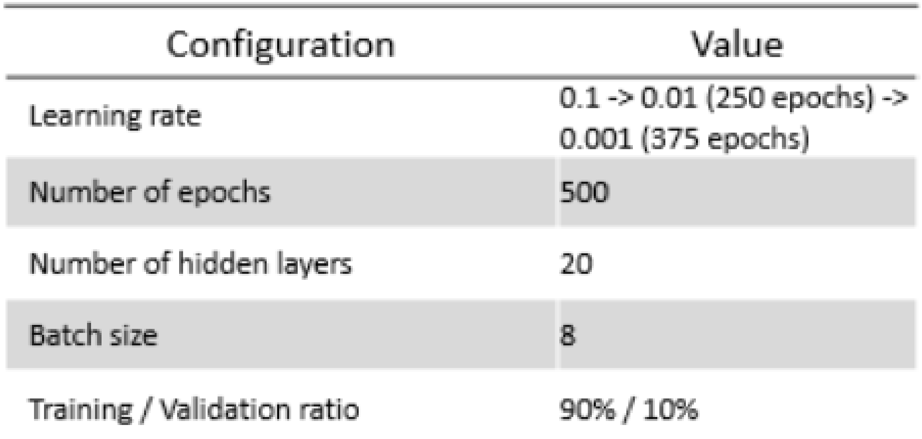
Table 16. ResLooper with different number of epochs on set casp



	100 epochs	300 epochs	500 epochs	700 epochs
Average RMSD	3.13	3.09	3.04	3.1
Std. dev.	1.21	1.21	1.27	1.2

Average RMSD and standard deviation are calculated on set CASP with different configurations.

Table 17. Configuration of ResNet in ResLooper method after training on set casp



Configuration	Value
Learning rate	0.1 -> 0.01 (250 epochs) -> 0.001 (375 epochs)
Number of epochs	500
Number of hidden layers	20
Batch size	8
Training / Validation ratio	90% / 10%

Table 17 shows the final configuration of ResNet in ResLooper derived from the Set CASP with 9 proteins. The number of epochs is 500 while the number of hidden layers is 20. This configuration is used by ResLooper when compared with other loop modeling methods on other test datasets.

5.3.4 Performance comparison of loop modeling methods

In this section, the new methods are compared with 3 state-of-the-art loop modeling methods, including Next-generation KIC (NGK), GalaxyLoop-PS1 (Galaxy PS1) and GalaxyLoop-PS2 (Galaxy PS2) on the Set 8Res, Set 12Res and Set TBM datasets. For each protein in those datasets, ConLooper and ResLooper runs 10 times to derive the average RMSD of the main chain atoms and standard deviation.

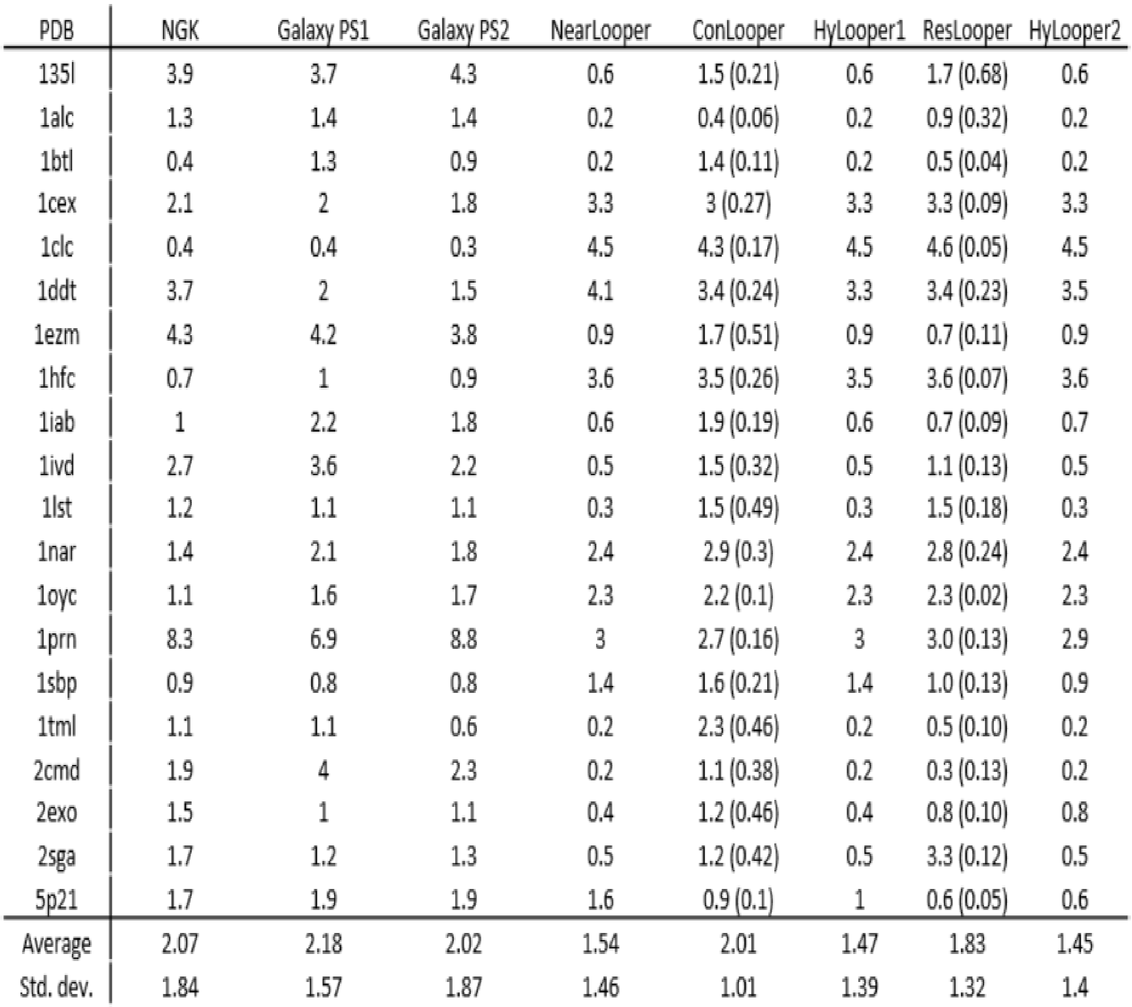
Table 18 shows the performances of the methods on 20 proteins in Set 8Res. Among the 3 existing methods, Galaxy PS2 achieves the best performance of average RMSD 2.02 Å and standard deviation 1.87. All five new methods achieve better results than the existing methods. HyLooper2 achieves the best performance with average RMSD 1.45 Å (28% improvement) and standard deviation 1.4. Individually, it outeperforms Galaxy PS2 in 65% (13/20 proteins) cases in this dataset.

Table 19 shows the performance of the methods on 20 proteins in Set 12Res. Similar to Table 7, among the 3 existing methods, Galaxy PS2 achieves the best performance of average RMSD 2.15 Å and standard deviation 1.4. It is also the best among all methods. Among the new methods, HyLooper1 is the best. Comparing with Galaxy PS2, HyLooper1 is comparable and outperforms Galaxy PS2 in 45% (9/20 proteins) cases in this dataset. Even though Galaxy PS2 achieves the best performance on Set 12Res, it has the disadvantage of needing initial loop information to do refinement while our methods can predict loop region directly without initial loop conformation.

Finally, Table 20 shows the results of loop modeling between different loop modeling methods. All five newly developed methods show good performance on this dataset over the best existing method Galaxy PS2 with average RMSD 3.73 Å and standard deviation 1.46. HyLooper2 achieves the best results with average RMSD 1.69 Å (54% improvement) and standard deviation 1.07. Individually, it outeperforms Galaxy PS2 in 77% (17/22 proteins) cases in this dataset.

Comparing between our five newly developed methods on test datasets, HyLooper2 is the best one in general. Between two deep learning methods, ResLooper is better than ConLooper which implies the better performance of ResNet over DCNN.

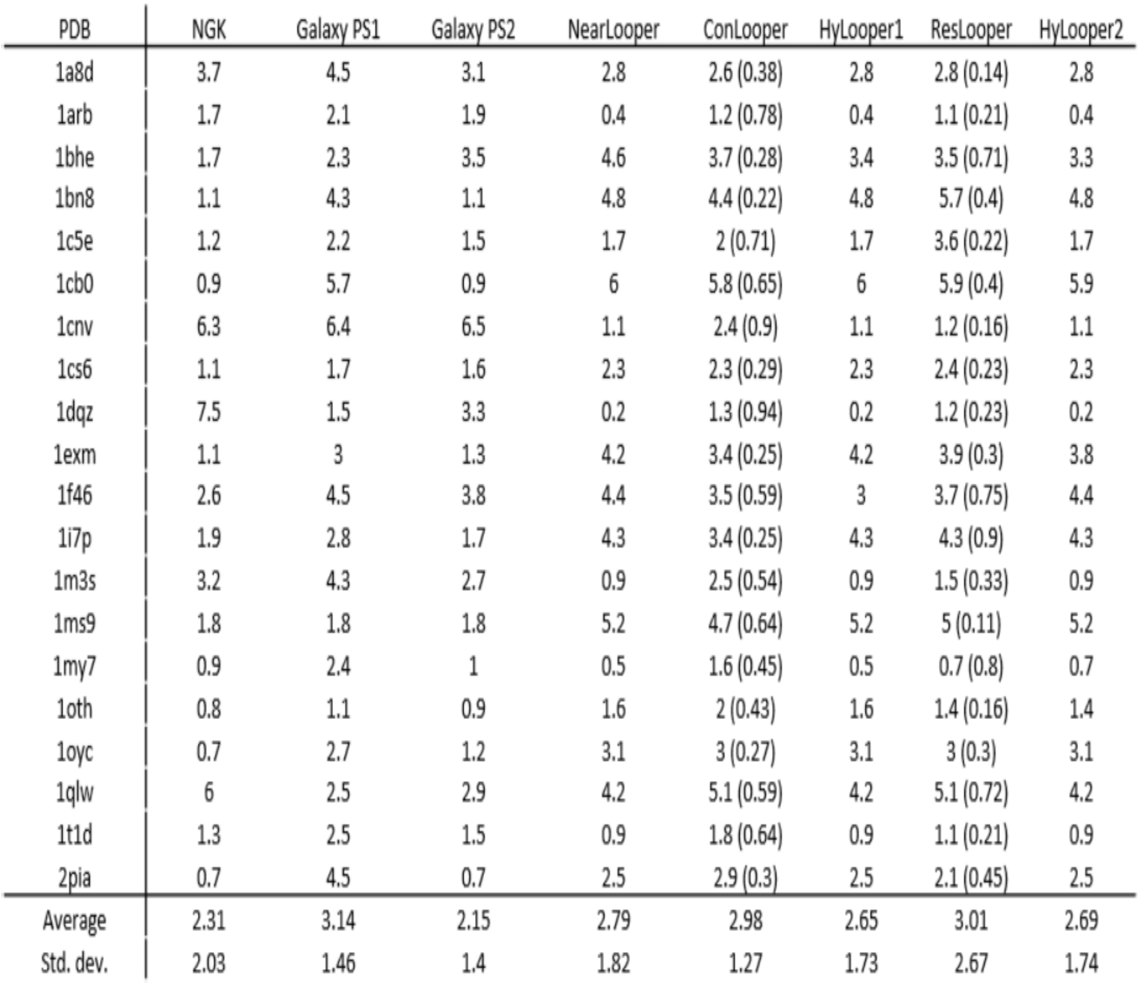
Table 18. Loop reconstruction results for the 8-residue loop dataset set 8Res



PDB	NGK	Galaxy PS1	Galaxy PS2	NearLooper	ConLooper	HyLooper1	ResLooper	HyLooper2
135l	3.9	3.7	4.3	0.6	1.5 (0.21)	0.6	1.7 (0.68)	0.6
1alc	1.3	1.4	1.4	0.2	0.4 (0.06)	0.2	0.9 (0.32)	0.2
1btl	0.4	1.3	0.9	0.2	1.4 (0.11)	0.2	0.5 (0.04)	0.2
1cex	2.1	2	1.8	3.3	3 (0.27)	3.3	3.3 (0.09)	3.3
1clc	0.4	0.4	0.3	4.5	4.3 (0.17)	4.5	4.6 (0.05)	4.5
1ddt	3.7	2	1.5	4.1	3.4 (0.24)	3.3	3.4 (0.23)	3.5
1ezm	4.3	4.2	3.8	0.9	1.7 (0.51)	0.9	0.7 (0.11)	0.9
1hfc	0.7	1	0.9	3.6	3.5 (0.26)	3.5	3.6 (0.07)	3.6
1iab	1	2.2	1.8	0.6	1.9 (0.19)	0.6	0.7 (0.09)	0.7
1ivd	2.7	3.6	2.2	0.5	1.5 (0.32)	0.5	1.1 (0.13)	0.5
1lst	1.2	1.1	1.1	0.3	1.5 (0.49)	0.3	1.5 (0.18)	0.3
1nar	1.4	2.1	1.8	2.4	2.9 (0.3)	2.4	2.8 (0.24)	2.4
1oyc	1.1	1.6	1.7	2.3	2.2 (0.1)	2.3	2.3 (0.02)	2.3
1prn	8.3	6.9	8.8	3	2.7 (0.16)	3	3.0 (0.13)	2.9
1sbp	0.9	0.8	0.8	1.4	1.6 (0.21)	1.4	1.0 (0.13)	0.9
1tml	1.1	1.1	0.6	0.2	2.3 (0.46)	0.2	0.5 (0.10)	0.2
2cmd	1.9	4	2.3	0.2	1.1 (0.38)	0.2	0.3 (0.13)	0.2
2exo	1.5	1	1.1	0.4	1.2 (0.46)	0.4	0.8 (0.10)	0.8
2sga	1.7	1.2	1.3	0.5	1.2 (0.42)	0.5	3.3 (0.12)	0.5
5p21	1.7	1.9	1.9	1.6	0.9 (0.1)	1	0.6 (0.05)	0.6
Average	2.07	2.18	2.02	1.54	2.01	1.47	1.83	1.45
Std. dev.	1.84	1.57	1.87	1.46	1.01	1.39	1.32	1.4

RMSD scores are used to compare performances among methods. For ConLooper and ResLooper, the experiment is executed 10 times to get the average RMSD and standard deviation for each protein. All the numbers are in Å.

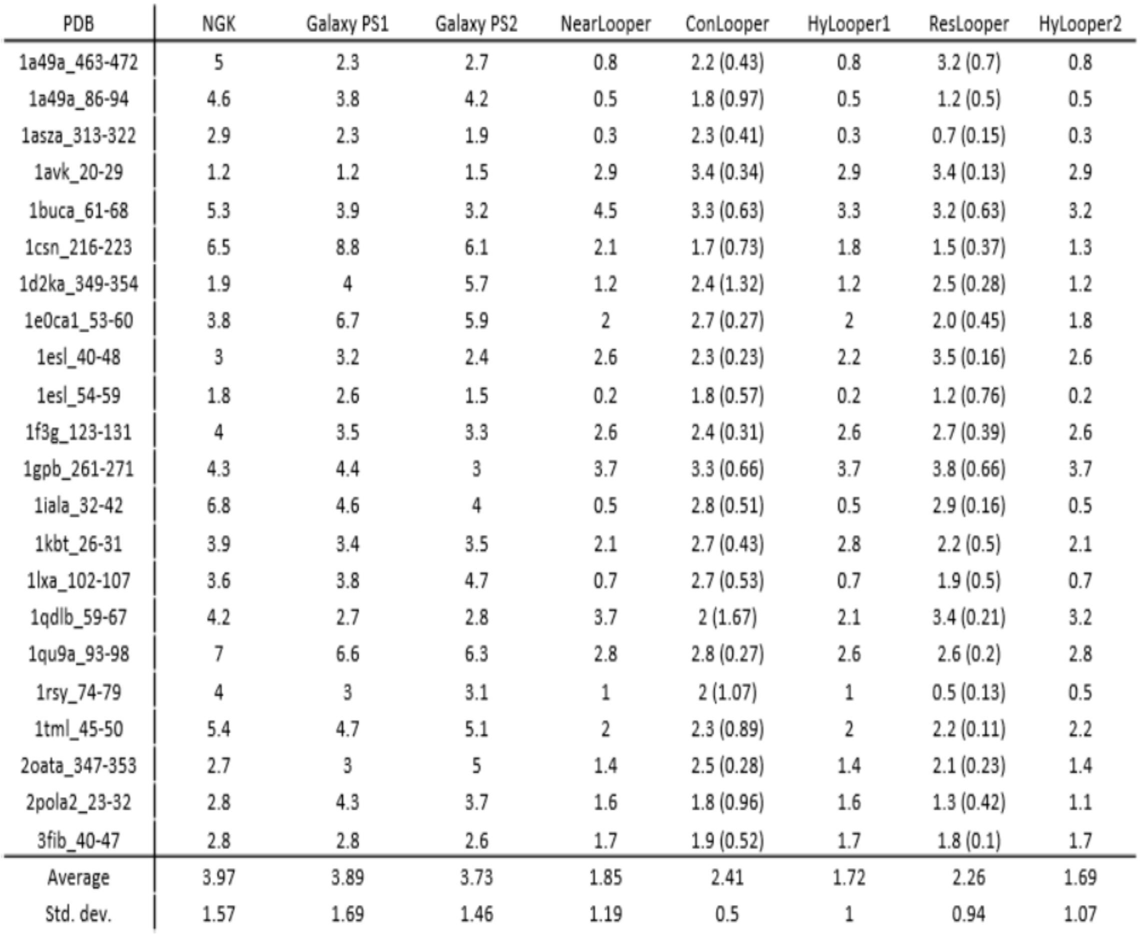
Table 19. Loop reconstruction results for the 12-residue loop data set set 12Res



PDB	NGK	Galaxy PS1	Galaxy PS2	NearLooper	ConLooper	HyLooper1	ResLooper	HyLooper2
1a8d	3.7	4.5	3.1	2.8	2.6 (0.38)	2.8	2.8 (0.14)	2.8
1arb	1.7	2.1	1.9	0.4	1.2 (0.78)	0.4	1.1 (0.21)	0.4
1bhe	1.7	2.3	3.5	4.6	3.7 (0.28)	3.4	3.5 (0.71)	3.3
1bn8	1.1	4.3	1.1	4.8	4.4 (0.22)	4.8	5.7 (0.4)	4.8
1c5e	1.2	2.2	1.5	1.7	2 (0.71)	1.7	3.6 (0.22)	1.7
1cb0	0.9	5.7	0.9	6	5.8 (0.65)	6	5.9 (0.4)	5.9
1cnv	6.3	6.4	6.5	1.1	2.4 (0.9)	1.1	1.2 (0.16)	1.1
1cs6	1.1	1.7	1.6	2.3	2.3 (0.29)	2.3	2.4 (0.23)	2.3
1dqz	7.5	1.5	3.3	0.2	1.3 (0.94)	0.2	1.2 (0.23)	0.2
1exm	1.1	3	1.3	4.2	3.4 (0.25)	4.2	3.9 (0.3)	3.8
1f46	2.6	4.5	3.8	4.4	3.5 (0.59)	3	3.7 (0.75)	4.4
1i7p	1.9	2.8	1.7	4.3	3.4 (0.25)	4.3	4.3 (0.9)	4.3
1m3s	3.2	4.3	2.7	0.9	2.5 (0.54)	0.9	1.5 (0.33)	0.9
1ms9	1.8	1.8	1.8	5.2	4.7 (0.64)	5.2	5 (0.11)	5.2
1my7	0.9	2.4	1	0.5	1.6 (0.45)	0.5	0.7 (0.8)	0.7
1oth	0.8	1.1	0.9	1.6	2 (0.43)	1.6	1.4 (0.16)	1.4
1oyc	0.7	2.7	1.2	3.1	3 (0.27)	3.1	3 (0.3)	3.1
1qlw	6	2.5	2.9	4.2	5.1 (0.59)	4.2	5.1 (0.72)	4.2
1t1d	1.3	2.5	1.5	0.9	1.8 (0.64)	0.9	1.1 (0.21)	0.9
2pia	0.7	4.5	0.7	2.5	2.9 (0.3)	2.5	2.1 (0.45)	2.5
Average	2.31	3.14	2.15	2.79	2.98	2.65	3.01	2.69
Std. dev.	2.03	1.46	1.4	1.82	1.27	1.73	2.67	1.74

RMSD scores are used to compare performances among methods. For ConLooper and ResLooper, the experiment is executed 10 times to get the average RMSD and standard deviation for each protein. All the numbers are in Å.

Table 20. Loop reconstruction results for data set set TBM



PDB	NGK	Galaxy PS1	Galaxy PS2	NearLooper	ConLooper	HyLooper1	ResLooper	HyLooper2
1a49a_463-472	5	2.3	2.7	0.8	2.2 (0.43)	0.8	3.2 (0.7)	0.8
1a49a_86-94	4.6	3.8	4.2	0.5	1.8 (0.97)	0.5	1.2 (0.5)	0.5
1asza_313-322	2.9	2.3	1.9	0.3	2.3 (0.41)	0.3	0.7 (0.15)	0.3
1avk_20-29	1.2	1.2	1.5	2.9	3.4 (0.34)	2.9	3.4 (0.13)	2.9
1buca_61-68	5.3	3.9	3.2	4.5	3.3 (0.63)	3.3	3.2 (0.63)	3.2
1csn_216-223	6.5	8.8	6.1	2.1	1.7 (0.73)	1.8	1.5 (0.37)	1.3
1d2ka_349-354	1.9	4	5.7	1.2	2.4 (1.32)	1.2	2.5 (0.28)	1.2
1e0ca1_53-60	3.8	6.7	5.9	2	2.7 (0.27)	2	2.0 (0.45)	1.8
1esl_40-48	3	3.2	2.4	2.6	2.3 (0.23)	2.2	3.5 (0.16)	2.6
1esl_54-59	1.8	2.6	1.5	0.2	1.8 (0.57)	0.2	1.2 (0.76)	0.2
1f3g_123-131	4	3.5	3.3	2.6	2.4 (0.31)	2.6	2.7 (0.39)	2.6
1gpb_261-271	4.3	4.4	3	3.7	3.3 (0.66)	3.7	3.8 (0.66)	3.7
1iala_32-42	6.8	4.6	4	0.5	2.8 (0.51)	0.5	2.9 (0.16)	0.5
1kbt_26-31	3.9	3.4	3.5	2.1	2.7 (0.43)	2.8	2.2 (0.5)	2.1
1lxa_102-107	3.6	3.8	4.7	0.7	2.7 (0.53)	0.7	1.9 (0.5)	0.7
1qdlb_59-67	4.2	2.7	2.8	3.7	2 (1.67)	2.1	3.4 (0.21)	3.2
1qu9a_93-98	7	6.6	6.3	2.8	2.8 (0.27)	2.6	2.6 (0.2)	2.8
1rsy_74-79	4	3	3.1	1	2 (1.07)	1	0.5 (0.13)	0.5
1tml_45-50	5.4	4.7	5.1	2	2.3 (0.89)	2	2.2 (0.11)	2.2
2oata_347-353	2.7	3	5	1.4	2.5 (0.28)	1.4	2.1 (0.23)	1.4
2pola2_23-32	2.8	4.3	3.7	1.6	1.8 (0.96)	1.6	1.3 (0.42)	1.1
3fib_40-47	2.8	2.8	2.6	1.7	1.9 (0.52)	1.7	1.8 (0.1)	1.7
Average	3.97	3.89	3.73	1.85	2.41	1.72	2.26	1.69
Std. dev.	1.57	1.69	1.46	1.19	0.5	1	0.94	1.07

RMSD scores are used to compare performances among methods. For ConLooper and ResLooper, the experiment is executed 10 times to get the average RMSD and standard deviation for each protein. All the numbers are in Å.

5.4 Conclusion

This research presents five new loop modeling methods, in particular a new approach based on deep learning (DCNN, ResNet) and distance matrix representation of 3-D structures. From the rotation and translation independent representation of distance matrix, DCNN and ResNet are able to learn a mapping function to predict a loop model. To the best of our knowledge, this is the first attempt to combine deep learning and geometric information for loop modeling.

Experiments using selected CASP models and common benchmark datasets have shown promising results. Compared to state-of-the-art loop modeling methods, our HyLooper2 achieves the best result on two test datasets and being comparable on the other test dataset. The ResNet in ResLooper is able to generate good loop candidates in many cases. The HyLooper2 method, which is the combination of ResLooper and NearLooper achieves good performance because both methods are complementary and have their own strengths.

Due to the limitation of computing power, experiments have been done on a limited sets of different network configurations. Exploring more network configurations can help improve the overall performance of loop modeling. This would be a good direction for future work.

CHAPTER 6. SUMMARY

This dissertation presents several new approaches based on C-α atoms distance matrix and deep learning methods for single-model QA as well as loop modeling problem. To the best of our knowledge, this is the first attempt to use purely geometric information of a model and deep learning for these problems. Three new QA algorithms, DL-Pro, FFNN-Pro, and SVM-Pro using different learning methods have been proposed, implemented and tested to classify quality of protein prediction whether it is good or not. For predicting quality of single model, DeepConQA method has been developed. Also, five new methods based on machine learning techniques, called NearLooper, ConLooper, ResLooper, HyLooper1 and HyLooper2 are proposed to solve the problem of loop modeling.

For the classification task of predicting quality of protein models, experiments using selected CASP models and targets show some promising results. Compared to traditional feedforward neural networks, deep learning is better, as demonstrated by the performance difference between DL-Pro and FFNN. Deep learning was able to learn useful features representing good models and DL-Pro achieved the best results, outperforming state-of-the-art energy/scoring functions, including DFIRE, OPUS-CA, DOPE, and RW. Yet, the information used by DL-Pro is far less than other single-model QA methods. With additional model information, DL-Pro is expected to improve further.

For the single-model QA task, experiments on free modeling targets of our method DeepCon-QA has achieved comparable performance to the best single-model QA method. By combining novel features of protein vector representation with existing features of profile information, together with the power of deep networks, our method shows promising results in solving the problem of protein quality assessment. Moreover, this project also shows that protein vector representation is a very useful set of features that can be used for QA tasks compare with profile features which are a very commonly used in many QA methods.

For the loop modeling problem, experiments using selected CASP models and common benchmark datasets have shown promising results. The ResNet in ResLooper method is able to generate good loop candidates in many cases. The HyLooper2 method, which is the combination of ResLooper and NearLooper achieves good performance because both methods are complementary and have their own strengths.

In conclusion, deep learning with the ability of extracting abstracted features as well as learning from a very big amount data has open up promising opportunities for many problems in protein prediction area.

CHAPTER 7. FUTURE WORK

For the QA problem, with some promising initial results, more experiments on bigger data set need to be done to determine in what cases the method works well and vice versa. Next, DL-Pro algorithm can be improved to be able to give a specific QA value instead of a label of good or bad only. Furthermore, local QA can also be done using deep learning. With the availability of many protein structures from Protein Data Bank, deep learning can help extract common features of amino acids in the sequence. Then it can provide more local information for the evaluating model. The work with DeepCon-QA method also shows that protein vector representation is a very promising set of features that can be applied in other applications of protein structure prediction are.

For the model prediction problem, stacked autoencoder can be used instead of deep belief network to do reconstruction over distance matrix. Also, new methods for combining distance matrix from alignments should be considered. The experiments show that a good initial combined distance matrix can significantly improve the quality of prediction.

BIBLIOGRAPHY

[1] M. S. Johnson, N. Srinivasan, R. Sowdhamini, and T. L. Blundell, "Knowledge-based protein modeling," Crit Rev Biochem Mol Biol, vol. 29, pp. 1-68, 1994.

[2] A. Sali and T.L. Blundell, "Comparative protein modelling by satisfaction of spatial restraints," J Mol Biol, 234(3):779-815, Dec 5 1993.

[3] J. Söding, A. Biegert, and A. N. Lupas, "The HHpred interactive server for protein homology detection and structure prediction," Nucleic Acids Res, 33(Web Server issue):W244-8, Jul 1 2005.

[4] Y. Zhang, "I-TASSER server for protein 3D structure prediction," BMC Bioinformatics, 9:40, Jan 23 2008.

[5] D. E. Kim, D. fishChivian, and D. Baker, "Protein structure prediction and analysis using the Robetta server," Nucleic Acids Res, 32(Web Server issue):W526-31, Jul 1 2004.

[6] J. Zhang, Q. Wang, B. Barz, Z. He, I. Kosztin, Y. Shang, and D. Xu, "MUFOLD: A new solution for protein 3D structure prediction," Proteins, 78(5): 1137–1152, April 2010.

[7] Q. Wang, K. Vantasin, D. Xu, and Y. Shang, "MUFOLD-WQA: A New Selective Consensus Method for Quality Assessment in Protein Structure Prediction," Proteins, 79(Suppl 10): 185–195, 2011.

[8] P. Benkert, S. C. Tosatto, and T. Schwede, "Global and local model quality estimation at CASP8 using the scoring functions QMEAN and QMEANclust," Proteins, 77 Suppl 9:173-80, 2009.

[9] J. Cheng, Z. Wang, A. N. Tegge, and J. Eickholt, "Prediction of global and local quality of CASP8 models by MULTICOM series," Proteins, 77 Suppl 9:181-4, 2009.

[10] Y. Wu, M. Lu, M. Chen, J. Li and J. Ma, "OPUS-Ca: A knowledge-based potential function requiring only Ca positions," Protein Science, vol. 16, no. 7, pp. 1449-1463, 2007.

[11] J. Zhang and Y. Zhang, "A Novel Side-Chain Orientation Dependent Potential Derived from Random-Walk Reference State for Protein Fold Selection and Structure Prediction," PLoS ONE, vol. 5, no. 10, pp. 1-13, 2010.

[12] H. Zhou and Y. Zhou, "Distance-scaled, finite ideal-gas reference state improves structure-derived potentials of mean force for structure selection and stability prediction," Protein Sci, 11(11):2714-26, Nov 2002.

[13] M. Y. Shen and A. Sali, "Statistical potential for assessment and prediction of protein structures," Protein Sci, 15(11):2507-24, Nov 2006.

[14] A. Kryshtafovych, K. Fidelis, and A. Tramontano, "Evaluation of model quality predictions in CASP9," Proteins, 79 (Suppl 10):91–106, 2011.

[15] A. Kryshtafovych1, A. Barbato, K. Fidelis1, B. Monastyrskyy, T. Schwede, and A. Tramontano, "Assessment of the assessment: Evaluation of the model quality estimates in CASP10," Proteins: Structure, Function, and Bioinformatics. Aug 2013.

[16] J. Skolnick, "In quest of an empirical potential for protein structure," Curr Opin Struct Biol, vol. 16, pp. 166-171, 2006.

[17] B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. J. States, S. Swaminathann, and M. Karplus "Charmm a program for macromolecular energy, minimization, and dynamics calculations," Journal of Computational Chemistry, 4:187–217, 1982.

[18] Y. Duan, C. Wu, S. Chowdhury, M. C. Lee, G. Xiong, W. Zhang, R. Yang, P. Cieplak, R. Luo, T. Lee, J. Caldwell, J. Wang, and P. Kollman "A point-charge force field for molecular mechanic's simulations of proteins based on condensed-phase quantum mechanical calculations," Journal of Computational Chemistry, 24(16):1999–2012, 2003.

[19] H. Lu and J. Skolnick "A distance-dependent atomic knowledge-based potential for improved protein structure selection," Proteins, 44(3):223-32, Aug 15 2001.

[20] H. Gohlke, M. Hendlich, and G. Klebe "Knowledge-based scoring function to predict protein-ligand interactions," J Mol Biol, 295(2):337-56, Jan 14 2000.

[21] J. Qiu, W. Sheffler, D. Baker, and W. S. Noble "Ranking predicted protein structures with support vector regression," Proteins, 71(3):1175-82, May 15 2008.

[22] C. Anjum, "Protein Tertiary Model Assessment Using Granular," 2012.

[23] Q. Wang, Y. Shang, and D. Xu, "Improving a Consensus Approach for Protein Structure Selection by Removing Redundancy," IEEE/ACM transactions on computational biology and bioinformatics, vol. 8, no. 6, pp. 1708-1715, 2011.

[24] A. Zemla, "LGA: a method for finding 3D similarities in protein structures," Nucleic acids research, vol. 31, pp. 3370-3374, 2003.

[25] H. Hotelling, "Analysis of a complex of statistical variables into principal components," J. Educ. Psych., vol. 24, pp. 417-441, 498-520, 1933.

[26] Y. Le Cun, "Modeles Connexionnistes de L'Apprentissage," PhD Dissertation, PARIS 6, 1987.

[27] H. Bourlard and Y. Kamp, "Auto-Association by multilayer perceptrons and singular value decomposition," Biological cybernetics, Vol. 59, pp. 291-294, 1988.

[28] G. E. Hinton and R. S. Zemel, "Autoencoders, minimum description length, and helmholtz free energy," Advances in Neural Information Processing, pp. 3-10, 1994.

[29] A. Ng, "Sparse autoencoder" Lecture note. Available online at: http://www.stanford.edu/class/cs294a/sparseAutoencoder.pdf

[30] A. Ng, "Support Vector Machine" Lecture note. Available online at: http://cs229.stanford.edu/notes/cs229-notes3.pdf

[31] UFLDL Tutorial. Available via WWW: http://deeplearning.stanford.edu/wiki/index.php/UFLDL_Tutorial

[32] Principal Component Analysis. Available via WWW: http://en.wikipedia.org/wiki/Principal_component_analysis

[33] Crystal structure of ORF52 from Murid herpesvirus 4. Available via WWW: http://www.rcsb.org/pdb/explore.do?structureId=2h3r

[34] D. Wang, “Active labeling in deep learning and its application to emotion prediction”, Dissertation for PhD Degree, 2017.

[35] Z. He, “Methods for protein structure prediction,” Dissertation for PhD Degree, 2017.

[36] Course notes “Introduction to Multi-Layer Perceptron”: Available via WWW: http://www.iro.umontreal.ca/~pift6266/H10/notes/mlp.html

[37] Deep Learning tutorial. Available via WWW: http://deeplearning.net/tutorial/mlp.html

[38] D. Kingma, and B. Jimmy, "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).

[39] Convolutional Neural Networks for Visual Recognition. Available via WWW: http://cs231n.github.io/convolutional-networks/

[40] S. P. Nguyen, Y. Shang, D. Xu, "DL-PRO: A novel deep learning method for protein model quality assessment," (IJCNN) IEEE 2014 International Joint Conference on Neural Networks, 2014.

[41] J. Long, E. Shelhamer, and T. Darrell. Fully Convolutional Networks for Semantic Segmentation. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3431–3440, 2014.

[42] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86(11): 2278–2324, 1998.

[43] M. Fenzi, L. Leal-taixe, J. Ostermann, and T. Tuytelaars. Continuous Pose Estimation with a Spatial Ensemble of muf Regressors. In ICCV, 2015. 2, 5, 6

[44] Y. Taigman, M. A. Ranzato, T. Aviv, and M. Park. DeepFace : Closing the Gap to Human-Level Performance in Face Verification. CVPR, 2014.

[45] E. Asgari, M. R. K. Mofrad, “Continuous Distributed Representation of Biological Sequences for Deep Proteomics and Genomics,” PLoS One, 2015.

[46] D. Petrey, B. Honig, “Protein structure prediction: inroads to biology,” Mol Cell, 20:811–819, 2005.

[47] Z. Wang, J. Moult, “SNPs, protein structure, and disease,” Hum Mutat, 17:263–270, 2001.

[48] A. Fiser, “Protein structure modeling in the proteomics era,” Expert Rev Proteomics, 1:97–110, 2004.

[49] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, P. E. Bourne, “The Protein Data Bank,” Nucleic Acids Res, 28:235–242, 2000.

[50] C. S. Soto, M. Fasnacht, J. Zhu, L. Forrest, and B. Honig, “Loop modeling: Sampling, filtering, and scoring,” Proteins: Structure, Function, Bioinformatics, 2008.

[51] Z. Xiang, C. S. Soto, B. Honig, “Evaluating conformational free energies: the colony energy and its application to the problem of loop prediction,” Proc Natl Acad Sci USA, 99:7432–7437, 2002.

[52] M. P. Jacobson, D. L. Pincus, C. S. Rapp, T. J. Day, B. Honig, D. E. Shaw, R. A. Friesner, “A hierarchical approach to all-atom protein loop prediction,” Proteins, 55:351–367, 2004.

[53] A. Fiser, R. K. Do, A. Sali, “Modeling of loops in protein structures,” Protein Sci, 9:1753–1773, 2000.

[54] E. Michalsky, A. Goede, R. Preissner, “Loops in proteins (LIP)—a comprehensive loop database for homology modelling,” Protein Eng, 16:979–985, 2003.

[55] A. Stein, T. Kortemme, “Improvements to robotics-inspired conformational sampling in rosetta,” PLoS One, 8: e63090, 2013.

[56] H. Park, C. Seok, “Refinement of unreliable local regions in template-based protein models,” Proteins, 80: 1974–1986, 2012.

[57] H. Park, G. R. Lee, L. Heo, C. Seok, “Protein Loop Modeling Using a New Hybrid Energy Function and Its Application to Modeling in Inaccurate Structural Environments,” PLoS ONE, 9(11): e113811, 2014.

[58] P. S. Shenkin, D. L. Yarmush, R. M. Fine, H. J. Wang, C. Lev-inthal, “Predicting antibody hypervariable loop confor-mation. I. Ensembles of random conformations for ringlike structures,” Biopolymers, 26:2053–2085, 1987.

[59] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel, “Backpropagation applied to handwritten zip code recognition,” Neural computation, 541–551, 1989.

[60] A. Krizhevsky, I. Sutskever, G.E. Hinton, “ImageNet classification with deep convolutional neural networks,” Advances in Neural Information Processing Systems, 1097–1105, 2012.

[61] W. Ouyang, P. Luo, X. Zeng, S. Qiu, Y. Tian, H. Li, S. Yang, Z. Wang, Y. Xiong, C. Qian, Z. Zhu, R. Wang, C.C. Loy, X. Wang, X. Tang, “Deepid-net: multi-stage and deformable deep convolutional neural networks for object detection,” arXiv preprint arXiv:1409.3505, 2014.

[62] N. Zhang, J. Donahue, R. Girshick, T. Darrell, “Part-based RCNNs for fine-grained category detection,” European Conference on Computer Vision, 834–849, 2014.

[63] Y. Sun, Y. Chen, X. Wang, X. Tang, “Deep learning face representation by joint identification-verification,” Advances in Neural Information Processing Systems, 1988–1996, 2014.

[64] C. Yang, X. Lu, Z. Lin, E. Shechtman, O. Wang, H. Li, “High-Resolution Image Inpainting using Multi-Scale Neural Patch Synthesis,” arXiv, 2016

[65] V. Nair, G.E. Hinton, “Rectified linear units improve restrict-ed Boltzmann machines,” International Conference on Machine Learning, 807–814, 2010.

[66] Y. LeCun and Y. Bengio, “Convolutional Networks for imag-es, speech, and time series,” The Handbook of Brain Theory and Neural Networks, 255–258. MIT Press, Cambridge, Massachusetts, 1995.

[67] Convolutional Neural Networks toolbox. Available online at: http://www.vlfeat.org/matconvnet/

[68] P. Rotkiewicz, J. Skolnick, “Fast procedure for reconstruction of full-atom protein models from reduced representations,” Journal of computational chemistry, 29(9):1460-5, 2008.

[69] K. He, X. Zhang, S. Ren, J. Sun, "Deep residual learning for image recognition," Proceedings of the IEEE Conference on Com-puter Vision and Pattern Recognition, 2016.

[70] K. He, X. Zhang, S. Ren, J. Sun, "Identity mappings in deep residual networks," European Conference on Computer Vision. Springer In-ternational Publishing, 2016.

[71] Available online at: http://www.predictioncenter.org/casp11/index.cgi

[72] D. A. Case, T. A. Darden, T. E. Cheatham, C. L. Simmerling, J. Wang, et al. AMBER 12. University of California, San Francisco. 2012

[73] V. Hornak, R. Abel, A. Okur, B. Strockbine, A. Roitberg, et al, “Comparison of multiple amber force fields and development of improved protein backbone parameters,” Proteins, 65: 712–725, 2006.

[74] W. C. Still, A. Tempczyk, R. C. Hawley, T. Hendrickson, “Semianalytical Treatment of Solvation for Molecular Mechanics and Dynamics,” Journal of the American Chemical Society, 112: 6127–6129, 1990.

[75] D. Qiu, P. S. Shenkin, F. P. Hollinger, W. C. Still, “The GB/SA continuum model for solvation. A fast analytical method for the calculation of approximate Born radii,” Journal of Physical Chemistry A, 101: 3005–3014, 1997.

[76] MA. Marti-Renom, MS. Madhusudhan, A. Sali, “Alignment of protein sequences by their profiles,” Protein Science, 13:1071–1087, 2004.

[77] H. Braberg, BM. Webb, E. Tjioe, U. Pieper, A. Sali, et al., “SALIGN: a web server for alignment of multiple protein sequences and structures,” Bioinformatics, 28:2072–2073, 2012.

[78] K. Mizuguchi, CM. Deane, TL. Blundell, JP. Overington, “HOMSTRAD: a database of protein structure alignments for homologous families,” Protein Science, 7:2469–2471, 1998.

[79] F. Chollet, Keras, GitHub repository: https://github.com/fchollet/keras

[80] Abadi, Martín, et al. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems." arXiv, 1603.04467, 2016.

[81] A. Ray, E. Lindahl, B. Wallner, " Improved model quality assessment using ProQ2, " BMC Bioinformatics, 2012 ; 13():224.

[82] A. Ray, E. Lindahl, B. Wallner, " Improved model quality assessment using ProQ2, " BMC Bioinformatics, 2012 ; 13():224.

[83] B. Wallner, P. Larsson, A. Elofsson, “Pcons.net: protein structure prediction meta server, ” Nucleic Acids Res, 2007.

VITA

Son Phong Nguyen is a Ph.D. student in the Department of Electrical Engineering and Computer Science, University of Missouri, USA. He received the MS degree in Computer Science from the Department of Electrical Engineering and Computer Science, University of Missouri, USA in 2014. He is also a member of Upsilon Pi Epsilon – Honor Society for the Computing and Information Disciplines since 2011. His current research interests focus on machine learning, big data analytics, deep learning and bioinformatics.