A DEEP LEARNING METHOD FOR PROTEIN MODEL QUALITY ASSESSMENT

A Thesis

Presented to

The Faculty of the Graduate School

At the University of Missouri

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science

By

SON PHONG NGUYEN

Dr. Yi Shang, Advisor

JULY 2014

The undersigned, appointed by the dean of the Graduate School, have examined the thesis entitled

A DEEP LEARNING METHOD FOR PROTEIN MODEL QUALITY ASSESSMENT

Presented by Son Phong Nguyen

A candidate for the degree of

Master of Science

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

Dr. Yi Shang

Dr. Dong Xu

Dr. Ioan Kosztin

ACKNOWLEDGEMENTS

First, I would like to thank my advisor, Dr. Yi Shang, who has helped me from the very beginning of my academic life. His deep knowledge in our research area helps me look through the nature of problem and overcomes difficulties that we always had during the process of doing research. These experiences will be invaluable treasures for my career in the future.

I would also like to thank my friends and colleagues in our lab for their assistance and advices. It has been a great pleasure for me to know and work with them.

My love and appreciation also goes out to my family, my wife and my son, who always support me through the ups and downs of my life.

Finally, I would like to thank my committee members Dr. Dong Xu and Dr. Ioan Kosztin for their support on this thesis.

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 MUFOLD-WQA and United3D, 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 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 thesis, a new approach based on C-α atoms distance matrix and machine learning methods is proposed for single-model QA and the identification of native-like models. Different from existing energy/scoring functions and consensus approaches, this new approach is purely geometry based. Furthermore, a novel algorithm based on deep learning techniques, called DL-Pro, is proposed. 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 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.

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.

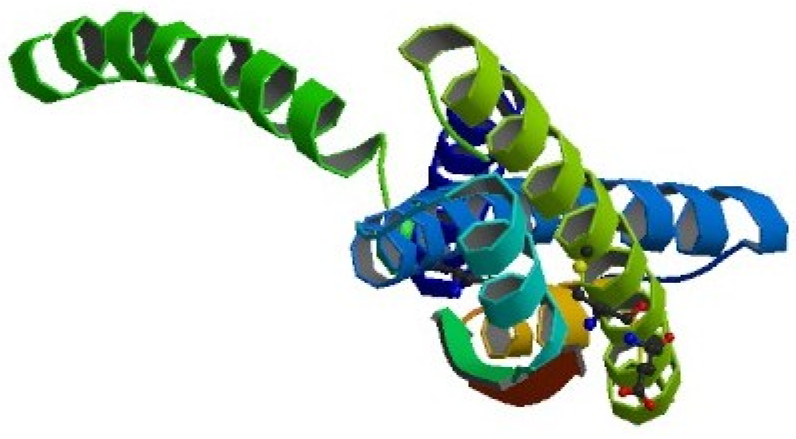

Figure 1. 3D structure of ORF52 from Murid herpesvirus 4 [33]

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 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 thesis, 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.

This thesis is organized as follows. Section 2 introduces the basics of major techniques used in the proposed method and some related works. Section 3 presents the new method DL-Pro. Section 4 presents experimental results on CASP datasets. Finally, Section 5 concludes the thesis.

2. BACKGROUND & RELATED WORK

2.1 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].

2.1.1 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:

$$\mathrm{GDT\_TS}(U_i,U_j)=(P_1+P_2+P_3+P_4)/4 \tag{1}$$

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 thesis.

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:

$$CGDT\_TS(U_i) = \sum_{j\in R} GDT\_TS(U_i, U_j) \,/\, N \tag{2}$$

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.

2.1.2 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].

2.2 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:

$$A_{ij} = \sqrt{\left(U_x^i - U_x^j\right)^2 + \left(U_y^i - U_y^j\right)^2 + (U_z^i - U_z^j)^2} \quad (3)$$

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

Figure 2 shows an example of the 3D structure and its corresponding distance matrix of a protein model.

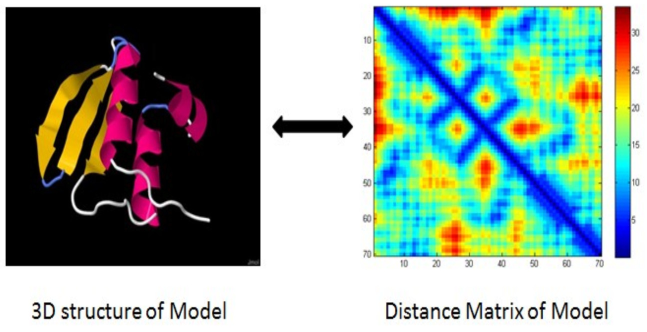

Figure 2. The 3D structure and its corresponding distance matrix of a protein model.

2.3 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.

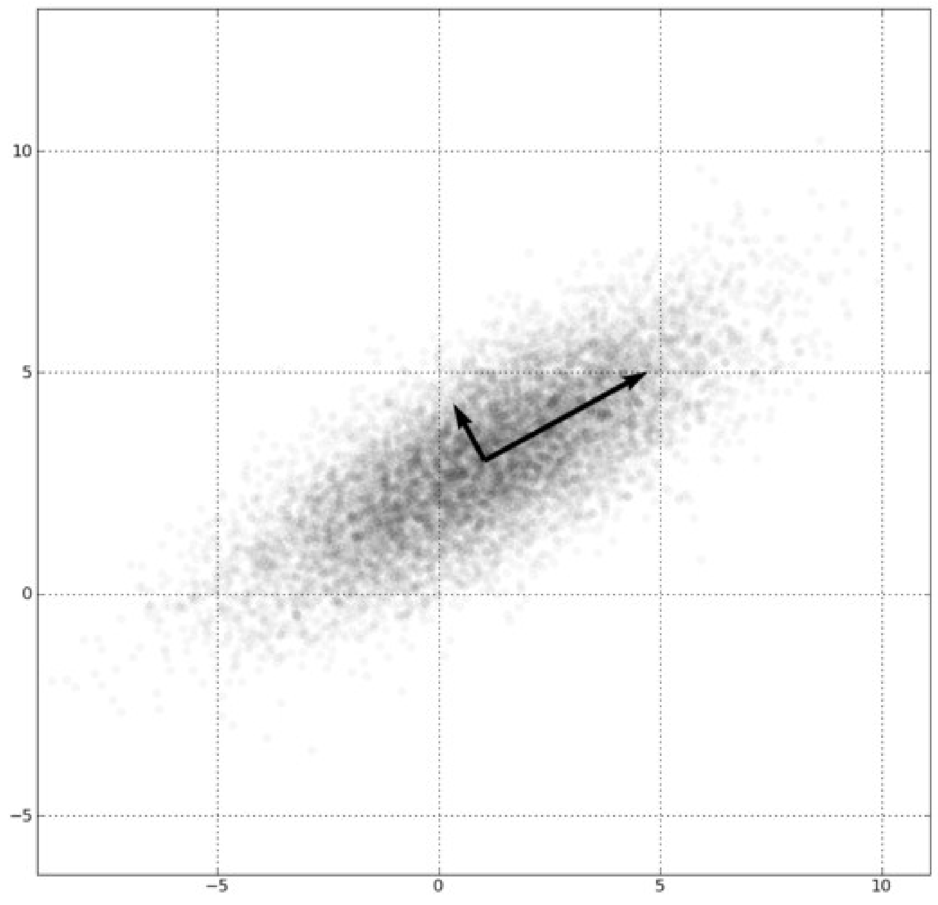

Figure 3. PCA of a multivariate Gaussian distribution

Figure 3 [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 length of arrows are scaled by the square root of the corresponding eigenvalue.

2.4 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:

$$\underset{W,b}{argmin} \quad J(W,b) = || h_{W,b}(x) - x || \tag{4}$$

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

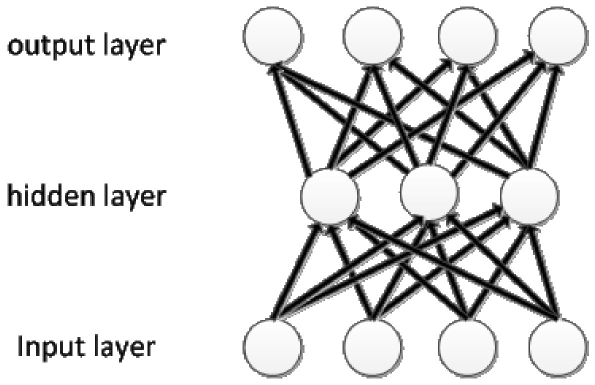

Figure 4. An example of autoencoder

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

$$\hat{p}_j = \frac{1}{m}\sum_{i=1}^{m}\left[a_j^{(2)}(x^{(i)})\right] \qquad (5)$$

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. (4):

$$R = \sum_{j=1}^{s_2} p log \frac{p}{\hat{p}_j} + (1-p) log \frac{1-p}{1-\hat{p}_j} \qquad (6)$$

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

$$J_{sparse}(W,b) = J(W,b) + \beta R \qquad (7)$$

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

Given the objective function in Eq. (7), its derivatives w.r.t. Wand 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.

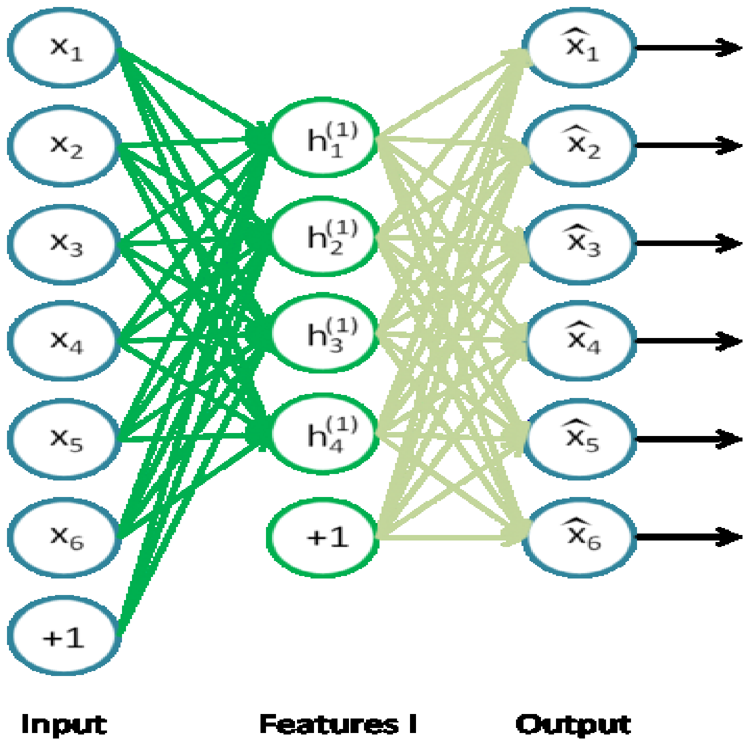

Figure 5. Train first layer of a stacked autoencoder

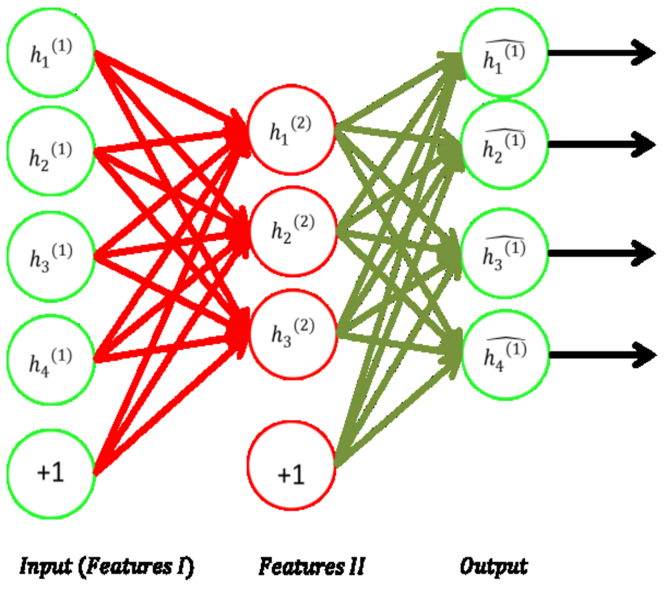

Figure 6. Train second layer of a stacked autoencoder

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].

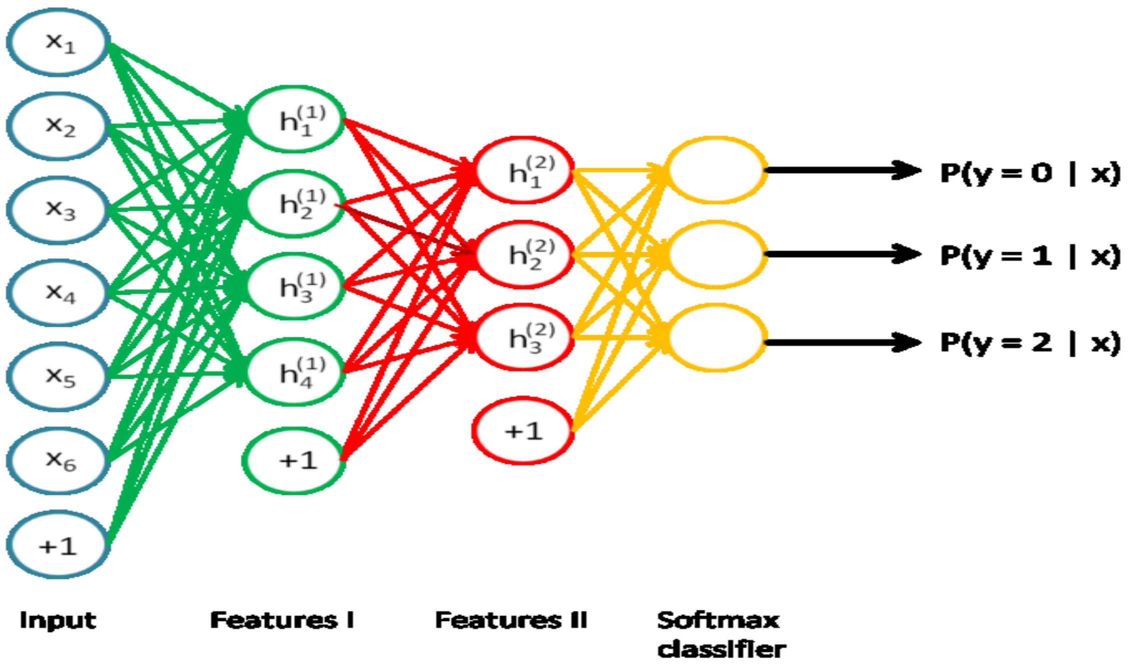

Figure 7. A stacked autoencoder with 2 hidden layers and a softmax classifier on top.

3. A DEEP LEARNING METHOD FOR PROTEIN MODEL QA

3.1 Problem formulation

The QA problem is formulated as a classification problem in this thesis: 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, $A=\{a_i,\ 1 \le i \le n\}$, and $a_i = \{U^j,\ j \in [1,l]\}$ where U^j is the 3D coordinates of residue j of model a_i. Let $C = \{c_i,\ 1 \le i \le n,\ 0 \le c_i \le 1\}$ 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 $c_i \ge 0.7$, 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 thesis, the performance metric of a classification algorithm is classification accuracy T:

$$T=v/n \tag{8}$$

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

3.2 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 thesis, 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 ci < 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 Θ1 and Θ2 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. [$s_0$, $\mu$, $\sigma$] ← EC_Train(S, G)
   2. [L] ← 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.

3.3 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.

3.3.1 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

    1. [DL_params, V, $\mu$, $\sigma$] $\leftarrow$ DL-Pro_Train(U, L)
    2. [L''] $\leftarrow$ 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
    1. M $\leftarrow$ 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] $\leftarrow$ PCA(M)
    6. DL_params $\leftarrow$ 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 $\leftarrow$ S2D(U)
    2. M $\leftarrow (M - \mu)/\sigma$
    3. [M'] $\leftarrow$ PCA_Test(M, V)
    4. [L'] $\leftarrow$ 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'.

3.3.2 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, μ, σ] ← SVM-Pro_Train(U, L)
   2. [L"] ← SVM-Pro_Test(U', SVM_params, V, μ, σ)

Output: L", predicted labels of test examples
```

```
Function SVM-Pro_Train(U, L)
Input: U and L, 3D structures and corresponding labels
    1. $M \leftarrow S2D(U)$
    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. $[M', V] \leftarrow PCA(M)$
    6. $SVM\_params \leftarrow SVM\_Train(M', L)$
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, $\mu$, $\sigma$)
    1. $M \leftarrow S2D(U)$
    2. $M \leftarrow (M - \mu)/\sigma$
    3. $[M'] \leftarrow PCA\_Test(M, V)$
    4. $[L'] \leftarrow SVM(SVM\_params, M')$
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'*.

3.3.3 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, μ, σ] ← FFNN -Pro_Train(U, L)
    2. [L"] ← FFNN -Pro_Test(U', FFNN _params, V, μ, σ)

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. $\sigma \leftarrow \sqrt{\frac{1}{n}\sum_{i=1}^{n}(M_i - \mu)^2}$
    4. $M \leftarrow (M - \mu)/\sigma$
    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
        μ, mean for normalization
        σ, standard deviation for normalization

Function FFNN -Pro_Test(U, FFNN _params, V, μ, σ)
    1. M ← S2D(U)
    2. $M \leftarrow (M - \mu)/\sigma$
    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'*.

4. EXPERIMENTAL RESULTS

4.1 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.

4.2 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.

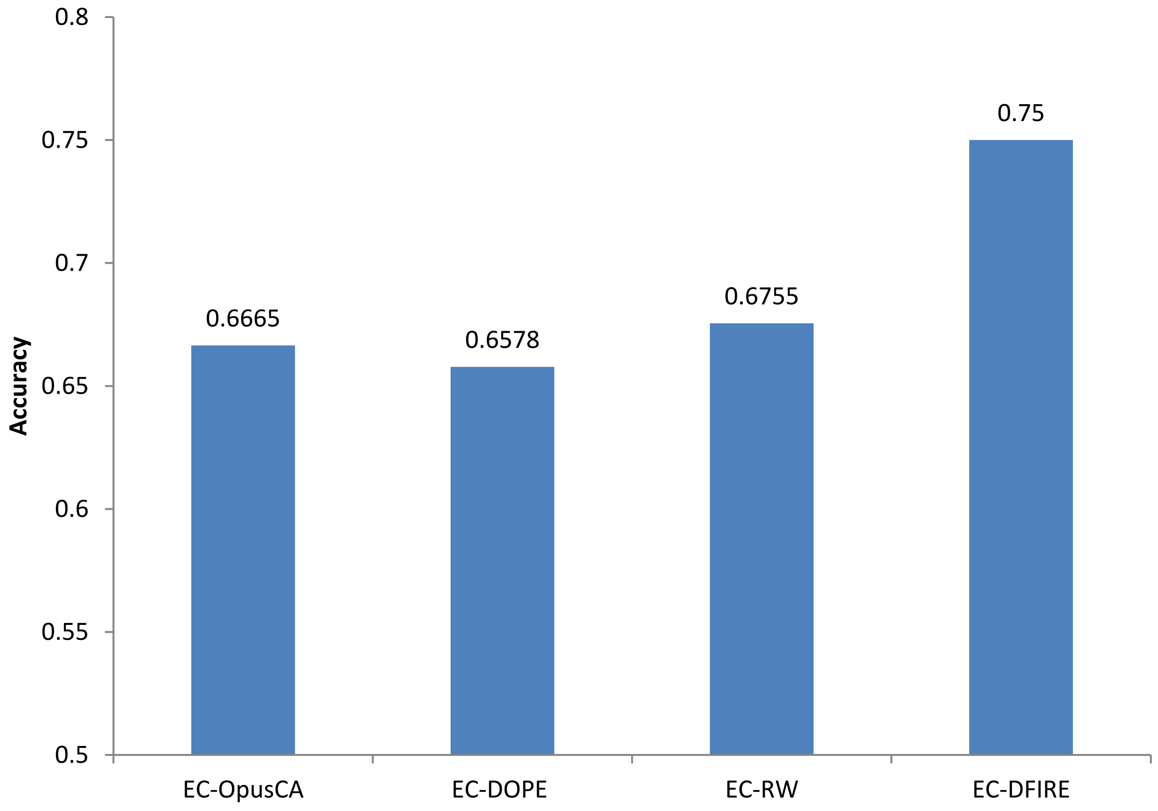

Figure 8. Classification performance of energy function based classification algorithms. The EC algorithm in Table 1 is applied to the CASP dataset with different energy scores obtained from OPUS-CA, DOPE, DFIRE, and RW, respectively.

Figure 8 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

4.3 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 9 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 10 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%.

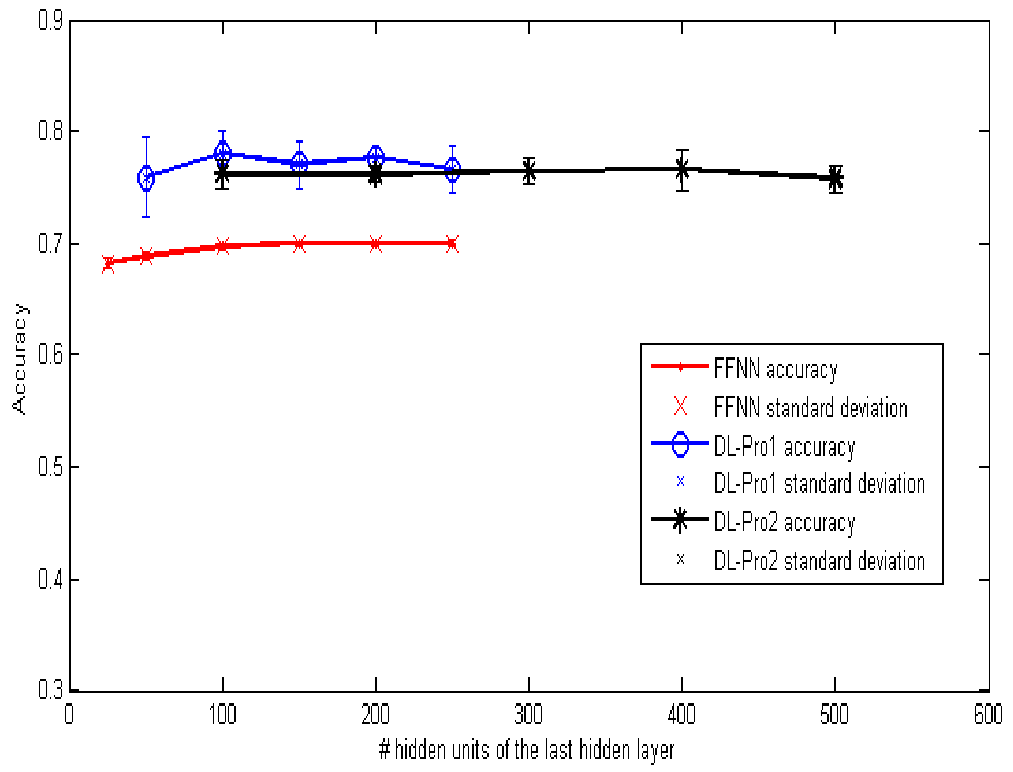

Figure 9. Classification performance of DL-Pro1 (DL-Pro with one-hidden-layer configurations), DL-Pro2 (DL-Pro with two-hidden-layer configurations), and FFNN with various hidden units.

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

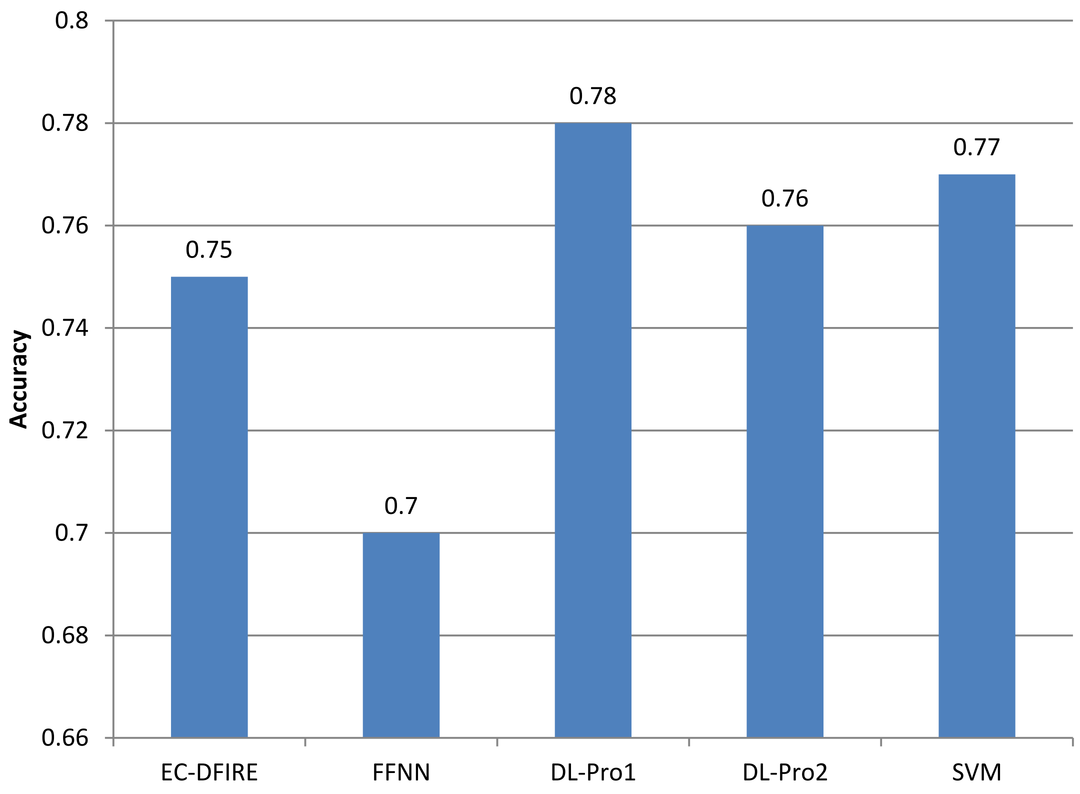

Figure 10. Classification performance of EC-DFIRE (the best of energy functions), SVM, FFNN, DL-Pro1 and DL-Pro2.

5. SUMMARY

This thesis presents a new approach based on C-α atoms distance matrix and machine learning methods for single-model QA. To the best of our knowledge, this is the first attempt to use purely geometric information of a model and deep learning for single-model QA. 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.

6. 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. Chivian, 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 mechanics 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