Protein Structure Prediction Based on Neural Networks

Jing Zhao

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfilment of the requirements
For the M.Sc. in Electronic Business Technologies

University of Ottawa

Abstract

Proteins are the basic building blocks of biological organisms, and are responsible for a variety of functions within them. Proteins are composed of unique amino acid sequences. Some has only one sequence, while others contain several sequences that are combined together. These combined amino acid sequences fold to form a unique three-dimensional (3D) shape. Although the sequences may fold proteins into different 3D shapes in diverse environments, proteins with similar amino acid sequences typically have similar 3D shapes and functions. Knowledge of the 3D shape of a protein is important in both protein function analysis and drug design, for example when assessing the toxicity reduction associated with a given drug. Due to the complexity of protein 3D shapes and the close relationship between shapes and functions, the prediction of protein 3D shapes has become an important topic in bioinformatics.

This research introduces a new approach to predict proteins' 3D shapes, utilizing a multilayer artificial neural network. Our novel solution allows one to learn and predict the representations of the 3D shape associated with a protein by starting directly from its amino acid sequence descriptors. The input of the artificial neural network is a set of amino acid sequence descriptors we created based on a set of probability density functions. In our algorithm, the probability density functions are calculated by the correlation between the constituent amino acids, according to the substitution matrix. The output layer of the network is formed by 3D shape descriptors provided by an information retrieval system, called CAPRI. This system contains the pose invariant 3D shape descriptors, and retrieves proteins having the closest structures. The network is trained by proteins with known amino acid sequences and 3D shapes. Once the network has been trained, it is able to predict the 3D shape descriptors of the query protein. Based on the predicted 3D shape descriptors, the CAPRI system allows the retrieval of known proteins with 3D shapes closest to the query protein. These retrieved proteins

may be verified as to whether they are in the same family as the query protein, since proteins in the same family generally have similar 3D shapes.

The search for similar 3D shapes is done against a database of more than 45,000 known proteins. We present the results when evaluating our approach against a number of protein families of various sizes. Further, we consider a number of different neural network architectures and optimization algorithms. When the neural network is trained with proteins that are from large families where the proteins in the same family have similar amino acid sequences, the accuracy for finding proteins from the same family is 100%. When we employ proteins whose family members have dissimilar amino acid sequences, or those from a small protein family, in which case, neural networks with one hidden layer produce more promising results than networks with two hidden layers, and the performance may be improved by increasing the number of hidden nodes when the networks have one hidden layer.

Acknowledgements

First and foremost, I would like to express my gratitude to my supervisors, Dr Herna L Viktor and Dr Eric Paquet, for their encouragement, guidance and support throughout the thesis process, from my first thoughts to the final conclusions.

And I would also like to thank my parents, Leping Zhao and Yumin Zhang, for their trust, encouragement and supports in this and all aspects of my life.

Contents

List of Figures

List of Tables

List of Algorithm

Chapter 1

Introduction

Proteins are macromolecules that constitute living organisms. They participate in biochemical reactions within organisms, and maintain structural or mechanical functions. For example, enzymes in the digestive system help animals assimilate nutrition from food. Research on protein functions contributes to understanding how our bodies work [1]. Protein functions are largely influenced by their 3D structures,

and knowledge of a protein's 3D structure improves the analysis of protein functions, particularly for drug design [2]. Being aware of the 3D structure of a protein enables pharmacologists to select a binding protein to moderate its functions. Pharmacologists may choose a drug which is easy to synthesize, and does not cause adverse effects. Thus, the determination of protein 3D structures plays a crucial role in drug design, and this research explores a cost-effective approach for 3D shape prediction [3].

Currently, the two methods used for protein 3D structure determination in drug design are experimental and computational. The experimental methods consist of X-ray crystallography and Nuclear Magnetic Resonance (NMR). Both these approaches are tedious, labour-intensive and time consuming, and NMR only works on small molecules [4]. The computational method is known as homology-based modelling. It predicts the 3D shape of an unknown protein based on its sequence alignment with a template protein, which has a known 3D structure. Although the computational method has several advantages over the experimental methods, its accuracy is template-intensive. This method assumes that the assignments from the chemical properties to the spatial regions between the query protein and the template protein are the same. However, this is not the case in reality, especially when the template protein and the query protein have partly different folds or different functions. For example, the prediction becomes complex, when the homology-based modelling applies to predicting proteins with similar structures but different chemical makeups, and thus, it requires more computing time [3].

In contrast, amino acid sequences may be acquired through very efficient automated, high throughput experimental methods [4]. The amino acid sequences of a vast number of proteins have been identified, while a relatively small number of 3D shapes are known. It follows that it is important to bridge this knowledge gap. That is, there is an urgent need to establish the relationship between proteins' amino acid sequences and 3D shapes, in order to enhance prediction [5]. This thesis addresses this issue.

Due to the progress in artificial intelligence, neural networks are now widely used for

prediction. Although these neural networks have been applied to explore protein structure prediction, they are mainly used to predict the secondary structures from the amino acid sequences [6]. There are still very few studies addressing the prediction of the secondary structure and the 3D shape of proteins.

1.1 Motivations

This research proposes a new approach to directly predict the 3D shape of a given protein from its amino acid sequences. As stated previously, the 3D shape of a protein is responsible for the interactions between proteins, and it is critical for scientists to understand the functions and mutual relationships [6] [7]. The prediction of 3D shapes is a complex task. Unlike the secondary structure of the protein, 3D shapes do not have obvious basic geometrical elements to define them [7].

To address this issue, we propose an approach based on a feed-forward neural network. Since the operation of the neural network is numerically-based, we create algorithms to represent the amino acid sequences and 3D shapes, so that their relationships may be analysed and learned by neural networks. In this supervised learning (or classification) setting, the amino acid sequence representations are used as inputs of the neural network, and the 3D shape representations are used as the outputs. The network is trained by proteins with known amino acid sequences and 3D shapes. Once the network has been trained, it is used to predict the 3D shape of the query protein. In the research, we explore different neural network architectures and optimization algorithms. Further, we consider different protein family sizes, shapes and sequences.

1.2 Objectives

The current methods used for 3D shape prediction is not as efficient as those used to acquire amino acid sequences. Thus, compared with the large number of known amino acid sequences, the number of known 3D shapes is relatively small. In this thesis, we

aim to predict the proteins' 3D shapes directly from their amino acid sequences, because proteins having similar amino acid sequences generally have similar 3D shapes.

Neural networks are good at analysing continuous-valued data and works by establishing the complex relationship between the inputs and outputs (classes). So, we use neural networks to analyse and establish the complex relationship between amino acid sequences and 3D shapes. Then, the trained neural network would be able to predict the 3D shapes of the unknown proteins directly from their amino acid sequences. In order to verify the accuracy of our predicted 3D shapes, we use the protein family membership, because proteins in the same family generally have similar 3D shapes. The proteins families are unknown during the neural network training process, and they are only used for verification.

1.3 Contribution

Our contribution is twofold. Firstly, we propose an approach to predict proteins' 3D shapes from their amino acid sequences by using neural networks. Although neural networks have been used in protein structure prediction, as far as the author is aware, they are only used in the proteins' secondary structure prediction, which concerns the local structure prediction. However, the 3D shape concerns the global structure of protein, which is highly relevant for studying protein interactions. Compared with current methods used for 3D shape prediction, which are either experimental or computational, our approach is more efficient, both in terms of training time and in terms of cost. Further, our method produces good prediction results for known 3D shapes, as will be discussed in Chapter 5.

As a second contribution, we created an algorithm to represent amino acid sequences in terms of a single descriptor, which are highly suitable for neural network training. In order to use neural networks to predict proteins' 3D shapes from their amino acid sequences, we need to model the amino acid sequences that act as input to our network.

Although amino acid sequences are involved in the secondary structure prediction, the methods usually use a sliding window to establish the relationship between sub-sequences and local structures. Since the 3D shape prediction concerns the global structure, we need to consider the whole chains of amino acid sequences. Our second contribution concerns the development of an algorithm that enables us to represent the entire amino acid sequence in a compact descriptor

1.4 Thesis Organization

This thesis consists of six chapters. Chapter 2 provides a literature review of protein structures. A review of current approaches used for protein 3D shape prediction in drug design is also provided. Chapter 3 is dedicated to neural networks. Since this thesis focuses on protein 3D shape prediction using neural networks, a detailed explanation of neural networks is provided. Chapter 4 presents an experimental design, which includes a detailed description of the algorithms we designed and the methodology we followed. In Chapter 5, the criteria used to evaluate prediction accuracy are presented, and the results are discussed. Chapter 6 concludes and summarizes this research, and proposes potential future research directions.

PART I

LITERATURE REVIEW

Chapter 2

Proteomics

Recall that the aim of this research is to use neural networks to predict the 3D shapes of proteins directly from their amino acid sequences. This chapter discusses background related to this task. We introduce protein structures in Section 2.1. We continue the discussion, in Section 2.2, of widely used methods for protein 3D shape

prediction, including both experimental and computational methods.

2.1 Protein Structures

Proteins bind to one another through diverse interactions, and then construct various living organisms. Each protein molecule has a complicated three-dimensional shape, formed by the folding of its linear amino acid sequences [1]. The structure of a protein may be considered using four distinct perspectives, namely the primary sequence, the secondary structure, the tertiary structure and the quaternary structure.

The primary structure of a protein refers to the linear sequence of its amino acids [8], and the amino acids of each protein are arranged in a unique order [7]. The secondary structure of a protein is formed by low-level folding of its primary structure [9] into regular local sub-structures (e.g. the alpha helix, the beta strands and the coils) [6]. A higher level of folding, based on the folding of the secondary structure, is described by its tertiary structure [9]. The tertiary structure of a protein is a description of its three-dimensional (3D) structure, and is defined by the coordinates of its constituent atoms in space [6]. The quaternary structure of a protein refers to the three-dimensional structures of several binding amino acid sequences [7], such as the three-dimensional structures of proteins consisting of more than one amino acid sequence [4].

The order of the amino acids in a protein is a primary source of information regarding its shape and functions [1]. Although the 3D shape of the protein is largely determined by its primary structure, knowing only the primary structure is not enough to deduce its secondary structure, or to predict its 3D shape [6]. However, proteins with similarities in their amino acid sequences often exhibit similar spatial structures, properties and functions [6], so the tertiary structure of a given protein may be identified by analysing the structures of the proteins with similar amino acid sequences [7].

Recall that the aim of this research is to predict the 3D shape of a protein (either the

tertiary structure or the quaternary structure of the protein, depending on the complexity of the protein) based on the analysis of relationships between amino acid sequences and 3D shapes.

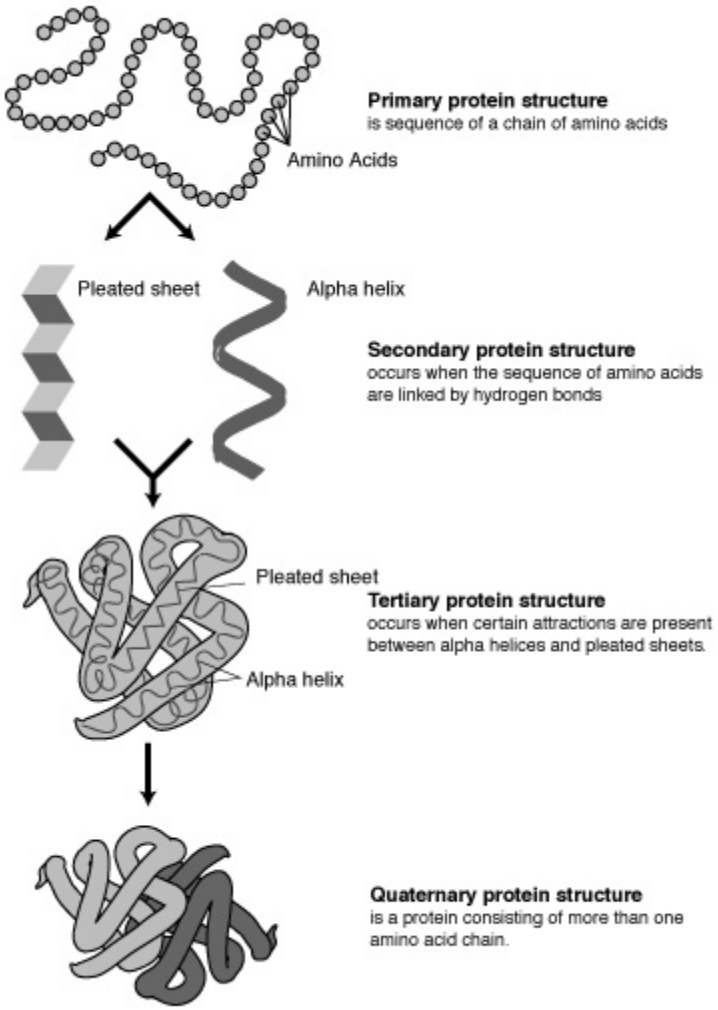


Figure 1. Protein Structures [10]

The information about the amino acid sequences may be found in the Protein Data Bank, an online resource. Specifically, in this research, we employ the RSCB (Research Collaboratory for Structural Bioinformatics) Protein Data Bank (PDB) to retrieve amino acid sequences, and then use our created algorithm to represent the sequences.

2.1.1 RSCB (Research Collaboratory for Structural Bioinformatics) Protein Data Bank (PDB)

The Protein Data Bank (PDB) is a repository of files that describe the positions of constituent atoms of each protein. These files may be used to construct various 3D representations of proteins. Structural data in the PDB are typically obtained by X-ray crystallography or Nuclear Magnetic Resonance (NMR), and then submitted by biologists and biochemists from around the world. The PDB is freely accessible on the Internet [11]. There are currently four portals for data viewing, namely RCSB PDB (USA), PDBe (Europe), PDBj (Japan) and BMRB (USA). Up to Sep 11 2012, 84,508 structures had been deposited in the PDB [12]. Each protein in the PDB is represented by its PDB ID, a four-character alphanumeric identifier (e.g. 1tyv for the tailspike protein). When inputting the PDB ID of a protein, users are able to view and download the PDB file and related data about that protein [11].

Recall that we use a neural network to learn the relationship between the amino acid sequence and the 3D shape. Both of these are represented in the form of an index or descriptor. In this research, we created an algorithm to create an index for the amino acid sequences that is based on the BLOSUM Matrix, as discussed below.

2.1.2 BLOSUM Matrix

The BLOSUM Matrix, first introduced by Henikoff and Henikoff, is used to score alignments between evolutionarily divergent protein sequences. Positive scores in the matrix are given to more likely substitutions, and negative scores to less likely substitutions [13]. In 1979, Miyata et al. showed that, in proteins that have preserved their structure throughout their evolution, the frequency of amino acid substitution correlates with the physico-chemical similarities between the exchanged amino acids [14]. Thus, the BLOSUM Matrix reflects the similarity between amino acids from a physio-chemical point of view. As shown in Figure 2, the letters in the row and column

represent the amino acids. Each cell is assigned a value that reflects the correlations between the amino acids in the corresponding row and column. A larger assigned number indicates a higher correlation between the amino acid in the row and the one in the column [13]. Section 4.2.1 explains how we use the BLOSUM Matrix in our algorithm.

According to the structural and evolutionary relationships, proteins may be grouped into families, and this information may be found in the SCOP system.

Blosum 62

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	4	-1	-2	-2	0	-1	-1	0	-2	-1	-1	-1	-1	-2	-1	1	0	-3	-2	0
R	-1	5	0	-2	-3	1	0	-2	0	-3	-2	2	-1	-3	-2	-1	-1	-3	-2	-3
N	-2	0	6	1	-3	0	0	0	1	-3	-3	0	-2	-3	-2	1	0	-4	-2	-3
D	-2	-2	1	6	-3	0	2	-1	-1	-3	-4	-1	-3	-3	-1	0	-1	-4	-3	-3
C	0	-3	-3	-3	9	-3	-4	-3	-3	-1	-1	-3	-1	-2	-3	-1	-1	-2	-2	-1
Q	-1	1	0	0	-3	5	2	-2	0	-3	-2	1	0	-3	-1	0	-1	-2	-1	-2
E	-1	0	0	2	-4	2	5	-2	0	-3	-3	1	-2	-3	-1	0	-1	-3	-2	-2
G	0	-2	0	-1	-3	-2	-2	6	-2	-4	-4	-2	-3	-3	-2	0	-2	-2	-3	-3
H	-2	0	1	-1	-3	0	0	-2	8	-3	-3	-1	-2	-1	-2	-1	-2	-2	2	-3
I	-1	-3	-3	-3	-1	-3	-3	-4	-3	4	2	-3	1	0	-3	-2	-1	-3	-1	3
L	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4	-2	2	0	-3	-2	-1	-2	-1	1
K	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5	-1	-3	-1	0	-1	-3	-2	-2
M	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5	0	-2	-1	-1	-1	-1	1
F	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6	-4	-2	-2	1	3	-1
P	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7	-1	-1	-4	-3	-2
S	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4	1	-3	-2	-2
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5	-2	-2	0
W	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11	2	-3
Y	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	-1
V	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4

Figure 2. BLOSUM Matrix [13]

2.1.3 SCOP System

Researchers have observed that, often, the structures of different proteins may be closely related. That is, there are proteins that share similar structures, and may have a common evolutionary origin. Knowledge of these relationships is crucial, not only for our understanding of the evolution of proteins, but also for analysis of the sequence data [15]. The Structural Classification of Proteins system (SCOP) was established to reveal the structural and evolutionary relationships between the proteins whose structure is in the PDB. Proteins are classified hierarchically into three major levels,

namely, Family, Superfamily and Fold [15]. In the SCOP system, proteins having similar shapes and sharing some similar sequences, or functions, are classified into the same Family. Protein families having similar shapes, but less sequence or function similarities, are placed into Superfamily. Structurally similar superfamilies with different characteristic features are classified into Folds [16]. Thus, the proteins in the same family generally share some similarities, in terms of their 3D shapes [6] [7] and their behaviour The **SCOP** database was first created by manual inspection and has been extended by a number of automated methods. As such, it provides a broad survey of all known protein folds together with detailed information about the close relatives of any particular protein. We use the SCOP system as a framework in our study [6].

2.2 Protein Structure Prediction for Drug Design

Recall from the introduction that there are two methodologies for protein 3D shape prediction, namely experimental methods and computational methods. In this section, we further explain how these methodologies proceed.

Recall that the most commonly used experimental methods for protein structure determination in drug design are X-ray crystallography and Nuclear Magnetic Resonance (NMR). In order to estimate the structure of proteins, the X-ray method uses the crystallization of the protein, and then analyses the scattering angles of the X-rays to obtain information about the structures. NMR identifies atoms and estimates their spatial coordinates based on the magnetic properties of their nuclei [1]. Unfortunately, both these methods are expensive and time consuming, and some proteins are not suitable for these techniques; NMR, for example, may only handle small molecules [17]. Despite rapid advances, the throughput of these experimental methods still lags behind the number of known proteins. Thus, only a fraction of known proteins have detailed information of their tertiary and quaternary structures [1]. As the experimental methods for protein structure determination are labour-intensive, computational techniques for structure prediction from amino acid sequences were

developed [17].

The computational approach for protein structure prediction is termed homology-based modelling, also known as comparative modelling. It constructs the shape of a query protein from its amino acid sequence. The construction is based on a model sequence to which the query protein is aligned for which the three-dimensional structure of the model sequence (the template protein) is known. Homology-based modelling typically involves four steps, namely template selection, target-template alignment, model construction and model assessment [18]. The underlying methods used for template selection assume that both the query protein and the template protein have the same assignment of chemical properties to spatial regions [3].

However, this is not always the case in real world, particularly for proteins with somewhat different folds or functions. It is complex for the homology-based modelling to select a template, when this approach applies on proteins with similar structures, but different chemical constitutions, and so, it requires extra computing time to solve. Currently, solutions for this take hours to find a template for a protein from a database with 1,500 templates [3].

After selection of the template protein, the side chains and loops of the query protein are modelled. For side-chain modelling, the model is first initiated by using knowledge-based methods, and then redefined by the energy minimization or molecular dynamics. However, both these side-chain modelling methods are computer-intensive, and may only be applied on one or a few proteins. The quality of the derived model is highly sensitive to the accuracy of the template input to the homology-based modelling tools. Loops modelling aims to model the loops that are not modelled by the template, but exist in the query protein structure. However, loops with more than five residues are still difficult to model [3].

Homology-based modelling for protein structure prediction is substantially dependent on the sequence alignments. Since it is template-intensive, its use is limited. In addition, it is time consuming when it deals with proteins that have similar

structures but different chemical constitutions. Thus, a more general and cost-effective approach for protein structure prediction is needed. This thesis addresses this need.

Although neural networks are proved to work well to establish complex relationships between their inputs and outputs, such as the relationship between amino acid sequences and 3D shapes, they have not, as far as the author is aware, been used to predict proteins' 3D shapes directly from their amino acid sequences. Since the 3D shape prediction concerns the global structures, it needs to concern a protein's composing amino acid sequences, no matter one or multiple sequences. Thus, methods to represent amino acid sequences and 3D shape by specific number of entries are needed before neural networks may be used for 3D shape prediction.

2.3 Summary

This chapter introduces protein structures and discusses the approaches used for protein structure prediction. Protein structures may be described from four perspectives, namely the amino acid sequences, the secondary structure, the tertiary structure and the quaternary structure. Information about proteins, such as their amino acid sequences, is provided by the RSCB PDB. The BLOSUM matrix, which reflects the correlation between two amino acids, was presented. Proteins may be classified into families, according to their structural and evolutionary relationships. This information is provided by the SCOP system. Next, we discuss the neural networks we used in our methodology.

Chapter 3

Artificial Neural Network

Recall that artificial neural networks (ANN) were originally proposed by psychologists and neurobiologists, who intended to simulate the computational processes of the human brain. As a highly complex, nonlinear parallel information processing system, the brain is able to organize its components and perform certain computations and analysis [19]. The brain is composed of a huge number of neurons, interconnected by

synapses. The neurons and synapses are used by the brain to transfer and analyse information, and then proceed to produce actionable results or feedback. Similar to the brain, an ANN is composed of artificial neurons, called processing units, and interconnections. In the simple ANN depicted in Figure 3, the processing units are represented by nodes and the interconnections by edges [19].

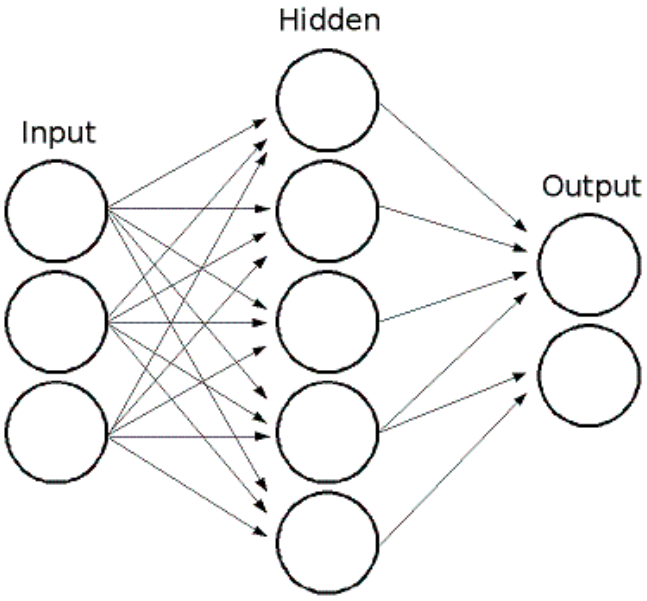


Figure 3. A three-layer Artificial Neural Network [20]

A neural network consists of a set of connected input and output units, with each connection associated to a weight [21]. In principle, ANN functionality is aiming to be similar to that of our current understanding of the brain; both acquire knowledge by learning or training with examples [19]. During the learning processes, the network compares its predicted outputs and the correct outputs, and minimizes the difference between them through adjusting the weights [21]. Acquired knowledge is stored in weights. Thus, we may describe the ANN as a formalized adaptive machine, and define it as follows [19]:

"An ANN is a massive parallel distributed processor made up of simple processing units. It has the ability to learn experiential knowledge expressed through inter-unit connection strengths, and may make such knowledge available for use [19]".

Neural networks have been widely used in many industries [22]. For example, its application may be found in image processing, signal processing, pattern recognition, robotics, automatic navigation, prediction and forecasting, and a variety of simulations, amongst others [23].

In this research, as stated before, we use an ANN to learning the relationship between amino acid sequences and 3D shapes. In Section 3.1 we present the strengths of neural networks. Section 3.2 introduces the use of neural network when aiming to learn under class imbalance. Class imbalance refers to the situation that one class is represented by a large number of training examples, while the other is represented by only a few. In our research, the class refers to the 3D shape of a protein. For a relationship between amino acid sequences and 3D shapes, there may exist a large number of proteins that can be used for neural network training, or there may not. Thus, in our experiments, class imbalance is caused by the differences in the number of proteins having similar amino acid sequences and 3D shapes, when compared with the query protein. In Section 3.3, we will state the operations of the neural network and optimization algorithms used during the neural network learning processes. These algorithms minimize the difference between the network predicted outputs and the target outputs related to the inputs. The use of neural networks in protein structure prediction is discussed in Section 3.3.

3.1 Strengths of Neural Networks

Neural networks have many strengths. Neural networks themselves are highly nonlinear, and this inherent nonlinearity makes them appropriate for analysing and generating data from the nonlinear real world. In addition, neural networks are well suited for continuous-valued inputs and outputs [19]. Some neural networks generally learn the relationships between inputs and outputs by modifying the weights involved in the networks. After training, the learned knowledge is stored in these weights. The neural networks are then able to produce correct outputs for new inputs that have not

appeared in the training set [19]. It follows that the performance of neural networks depends on the quality of the training set. If the networks perform on the appropriate training set, the result will be with a low error rate and high accuracy. To some extent, the performance of networks may be improved by increasing the number of examples in the training set [22]. Algorithms involved in neural networks are inherently parallel, which helps reduce the processing time [23].

The neural network is good at working on continuous-valued data and establishing the relationship between its inputs and outputs. Thus, it would be an appropriate approach to predict proteins' 3D shapes from the complex relationship with their amino acid sequences.

3.2 Learning under Class Imbalance

Class imbalance refers to the uneven distributed data, in which the number of training examples in the minority class is much smaller than those in the majority class [24]. A previous study, which is done by Murphy et al., showed that the traditional feed-forward neural network has difficulties when learning against imbalanced datasets [25]. Because of the large number of training examples in the majority class, the neural network tends to ignore the minority class. In order to solve this problem, a learning algorithm for the class imbalance may be generally divided into two categories, the resampling based and cost-effective based [24].

The resampling-based approach aims to solve the class imbalanced problem by modifying the prior probability of the minority and majority class in the training data set. Under-sampling refers to the process of reducing the number of examples in the majority class [26]. It extracts a smaller set of the majority class, but preserves all the examples in the minority class. Due to the extraction of examples in the majority class, this approach may lose some informative examples from the discarded examples. The under-sampling approach is suitable for large-scale applications with the tremendous number of examples in the majority class [24]. Over-sampling refers to the process of

increasing the number of examples in the minority class [26]. Compared with the under-sampling approach, the advantage of this approach is that no information would be lost, since all the examples in the training set are employed. The training examples in the minority class are over-represented and the increase of examples in the training set leads to the longer training time [24].

The cost-effective based approach is implemented by modifying the classifier. It was introduced by Berardi and Zhang [27] by assigning different costs to errors in different classes. This approach improved the prediction accuracy on the minority class by adding a cost function to its learning processes. During the learning processes, the cost function assigns larger cost to the minority classes [27]. The cost-effective approach is usually tailored for a specific problem, which may not applicable to a wide range of problems. However, it is better to design an approach for general learning problems with imbalanced data [26].

Recall that the neural network training in our experiments is based on the observation that, generally, proteins having similar amino acid sequences also have similar 3D shapes. For a given query protein, there may exist a large number of proteins having similar 3D shapes to it, or only a few. Even if there is a large number of protein having similar 3D shapes with the query protein, their amino acid sequences may present a wide variety in their proportion of constituent amino acids. Thus, in our experiments, the majority class has a larger number of proteins having similar amino acid sequences and 3D shapes, while the minority class has very few proteins having similar amino acid sequences and 3D shapes.

3.3 Artificial Neural Network - Supervised Training Algorithms

There are many different neural network models and algorithms. In this work, we use the multi-layer feed-forward neural network for supervised learning. Supervised learning infers the function from the supervised training data, and each example in the training data consists of an input and a target output. The supervised learning algorithm

analyses the relationship between the inputs and target outputs in the training set, and produces an inferred function. The inferred function is then used by the network to predict the output for a given input.

Recall that the aim of this research is to predict 3D shapes of proteins by neural networks. The prediction is based on the learned relationship between known amino acid sequences and 3D shapes, which is a process of supervised learning.

A commonly used algorithm for the multi-layer feed-forward neural network is the backpropagation algorithm. Its learning process may be divided into two sub-processes. First, the data feeds forward from the input layer to the output layer. Second, the error is back-propagated from the output layer to the input layer to optimize the difference in between the actual output and the target output. Together with the backpropagation algorithm, some other optimization methods may be used as the learning method. Therefore, we employs five learning methods, namely, the Gradient descent BP algorithm, the Gradient descent BP algorithm with momentum, the Conjugate gradient descent algorithm with Fletcher-Reeves updates, the BFGS Quasi-Newton algorithm and the Levenberg-Marquadt (LM) algorithm. The structure of multilayer feed-forward neural networks and the feed-forward process is presented in Section 3.2.1, and the optimization processes are introduced in Section 3.2.2.

3.2.1 A Multilayer Feed-forward Neural Network

A Multilayer feed-forward neural network is composed of several simple, highly interconnected computing units, called neurons, which are organized in layers. Recall that each neuron performs a simple task of information processing, i.e. converting received inputs into processed outputs. The neurons are connected through linking edges. The knowledge learned between network inputs and outputs is generated and stored as edges weights and biases, according to the strength of the relationships between different nodes. Although the information is distributed at each node, overall the neural network performs well and efficiently [28].

The architecture of a three-layer feed-forward neural network is shown in Figure 4. The neurons in this network are organized into three layers (i.e. input layer, hidden layer and output layer), and the neurons in each layer are fully connected to neurons in the adjacent layer. In the feed-forward network, the data propagates in one direction, from the input layer to the output layer, without feedback from the output layer [28].

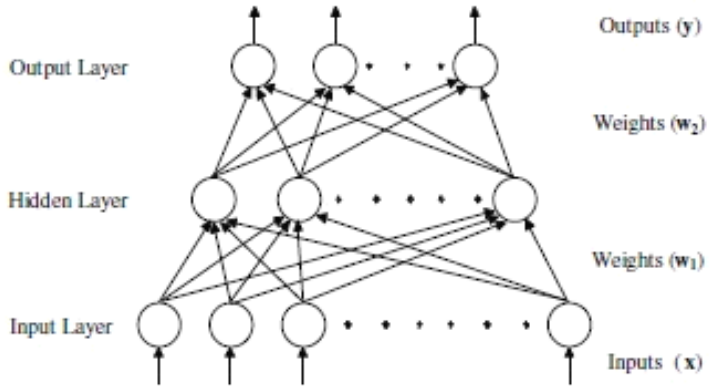


Figure 4. Multi-layer feed-forward neural networks [28]

Neurons in the input layer are passive, since they simply receive data and pass it on to the neurons in the hidden layer, without any data transformation. Neurons in the hidden layer are fully connected with neurons in both the input and output layers, and are critical for neural networks to learn the mapping relationships between inputs and outputs. After receiving information from neurons in the input layer, the hidden neurons process the information using the transfer function, and then propagate the processed information to the output layer for further processing to generate the outputs. Although neural networks may have more than one hidden layer, most applications only use one [28].

Thus, the multi-layer feed-forward neural network architecture is represented by the number of layers, the number of nodes in each layer, and the transfer function used in each layer, since the nodes in the same layer use the same transfer function. However, there is no widely accepted procedure to determine the architecture of an MLP, such as the number of hidden layers and the number of neurons in each layer. Therefore, it is a

complex process to construct a neural network [28]. In general, the number of hidden layers and number of neurons in each depends on the types of transfer functions and learning algorithms and the problems that need to be solved, and is usually determined empirically [23]. Due to the complexity of establishing a neural network, the cost of overly large neural networks may be high, particularly when the model structure is large and the input has a high number of dimensions [20].

In order to understand how a neural network operates, we first need to know how the neurons in the hidden layers and output layers process data; this mechanism is shown in Figure 5. Information is processed in two steps at each neuron in the hidden and output layers. In the first step, inputs, denoted by $\{x_1, x_2, ..., x_d\}$, from different links are multiplied by the weights, denoted by $\{\omega_1, \omega_2, ..., \omega_d\}$, related to each corresponding edge, and then combined to form a weighted sum. In the second step, the neuron uses the transfer function to convert the sum into the output. In many cases there is an additional step between the formation of the weighted sum and the transformation, as some networks may add a bias onto the weighted sum before it is transformed by the transfer function. The bias is a constant, which helps to improve the flexibility of the network [28].

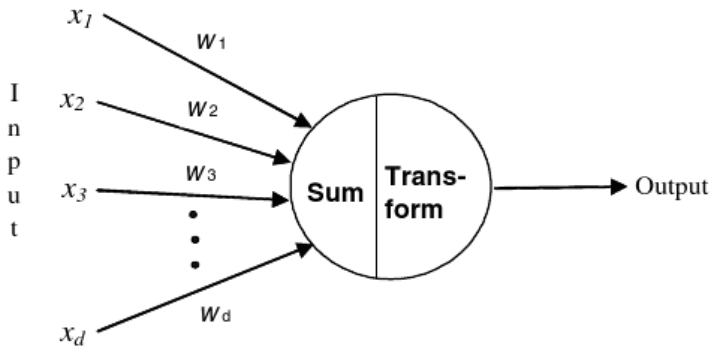


Figure 5. Information processing in a single neuron [20]

There are many options for the transfer function, but only a few are commonly used in practice. In general, the transfer function is bounded and non-decreasing [20].

The logistic function is the most widely used transfer function, particularly in the hidden layers, because it is simple, bounded, nonlinear and monotonically increasing [20].

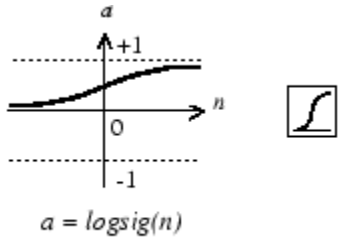


Figure 6. The Sigmoid (logistic) Function

If the logistic function is used in the hidden layer and the linear function is used in the output layer, the neural network structure for a feed-forward neural network may be expressed as:

$$y = b_2 + \sum_{j=1}^{q} w_{1j} f\left(\sum_{i=1}^{p} w_{ij} x_i + b_1\right) \quad \backslash* \text{ MERGEFORMAT (1)}$$

where y is the predicted output and $\{x_i, i = 1, 2, ..., p\}$ are the input tuples to the network, p is the number of nodes in the input layer, q is the number of nodes in the hidden layer, $\{w_{ij}, i = 0, 1, ..., p; j = 1, 2, ..., q\}$ represents the weights located between the input and hidden layers, $\{w_{1j}, i = 0, 1, ..., p; j = 1, 2, ..., q\}$ represents the weights between the hidden and output layers, b_1 and b_2 are biases in the hidden and output layers, respectively, and f is the logistic function defined above [20].

After determining the neural network topology, we train the neural network with the training set, which is composed of the input tuples and the known output tuples. Then the input values are weighted and summed at the hidden layer, and the weighted sum is transformed through a specific transfer function to form the inputs to the output layer,

or the inputs to another hidden layer. The same operation is conducted on the next layer until the network generates its outputs. The actual outputs produced by the network are compared with the desired (target) outputs to determine how closely they match. This is measured by a score function, the mean squared errors (MSE), denoted by E [19].

The goal of training a neural network is to optimize the score function, namely the mean squared error (MSE), to achieve its global minimal. During the optimizing processes, the set of weights and biases keeps updating. Thus, the network training process is essentially a nonlinear optimization problem.

3.2.2 Optimization Algorithms

Recall that backpropagation (BP) is a process that propagates the errors backward from network outputs to inputs. During this operation, weights and biases involved in the networks are updated to minimize the distance between the actual predicted values and the desired values. Essentially, the aim is to optimize the score function. Recall that, in this research we experiment with five optimization algorithms. These are the gradient descent BP, the gradient descent with momentum BP, the conjugate gradient algorithm with Fletcher-Reeves updates, the BFGS Quasi-Newton algorithm, and the Levenberg-Marquadt (LM) algorithm. The first three methods are based on the gradient of the score function, and the remaining two rely on the Hessian Matrix of the score function. In this section we present these methods in detail, and then discuss the advantages and disadvantages of each.

Recall that the optimization operation aims to optimize the score function, and we use the MSE function as the score function in our experiments. In the MSE function, the estimate value obtained from neural networks is composed by a function of the parameter θ , which is within the model and needs to be updated to minimize the error function. In a neural network, the parameters will be the weights and biases related to the network. We typically use iterative improvement search methods for this

optimization process, and local information to guide the local search. The iterative local optimization algorithm may be broken down into three steps [20]:

1. **Initialize:** Choose the initial values for the vector of parameter θ .In general these are chosen randomly;
2. **Iterate:** Starting from $i=0$, let $\theta^{i+1}=\theta^{i}+\lambda^{i}\vec{\nu}^{i}$, where $\vec{\nu}$ is the direction for the next step, and λ represents the step size. In our research, $\vec{\nu}$ is a direction in which the score function is minimized;
3. **Convergence:** Repeat step 2 until the score function achieves a local minimum.

Essentially, all the following optimization algorithms provide the required information to determine $\vec{\nu}$, the direction for the next step [20]. To find the local minimum, we calculate the gradient and the Hessian of the targeted function when the function is twice continuously differentiable [29].

Gradient Descent BP

The gradient descent BP, the most basic BP algorithm, is known as the steepest descent. Among all directions we may move, the steepest descent direction $-\nabla_{\theta}E_{(\theta)}$ is the direction along which the target function (the score function MSE) decreases most rapidly [29].

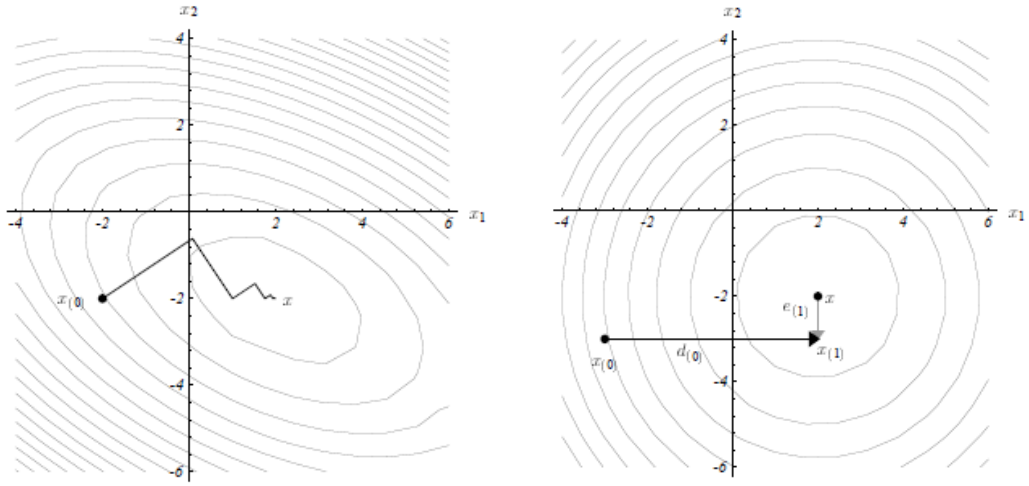
The advantage of this method is that it requires only the calculation of the gradient $\nabla_{\theta}E(\theta)$, but not the second derivates (Hessian) [29]. If the chosen learning rate is small enough, the gradient descent is guaranteed to be convergent [29].

Gradient Descent with momentum BP

This method is based on the gradient descent BP, but it accelerates its convergence by adding a momentum. This is because there is always a trade-off when choosing the learning rate: if it is too small the convergence will take too long, while if it is too large we will step too far and miss the minimum.

When the momentum term equal to 0, this optimization algorithm is equivalent to the gradient descent BP, and when the momentum term is larger than 0, the algorithm will accelerate the convergence in the current direction [20].

Conjugate Gradient Descent Algorithm with Fletcher-Reeves Updates



(a) **(b)**

Figure 7. Search direction of Gradient Descent Method (a) and Conjugate Gradient Method (b) [29]

Both methods described above optimize the score function in the direction with the steepest descent gradient. As it is shown in Figure 7 (a), the gradient descent method takes the right angle after each step [29]. However, the search direction that is taken by the conjugate gradient descent method would be conjugate to all previous search directions. Thus, the conjugate gradient method converges faster than the gradient descent method.

The conjugate gradient algorithm was originally designed for optimizing linear

functions of the form $Ax = b$, which is equivalent to minimize a convex quadratic function $\phi(x) = \frac{1}{2}x^T Ax - b^T x$ where A represents a n×n matrix, which is symmetric, positive definite and real, and b is a n-dimensional vector. Then, we may calculate the gradient of this function $\phi(x)$.

The searching directions $\{p_0, p_1, ..., p_l\}$ are said to be conjugate with respect to a symmetric positive definite matrix A, if $p_i^T Ap_j = 0$, for all $i \neq j$.This property enables us to find the minimum of the function $\phi(x)$ by minimizing the function along each individual direction in the conjugate set.

In order to extend this method to more general convex functions or even general nonlinear functions, we need to use the gradient $g(\theta^i)$ of the nonlinear function E and perform a nonlinear search to investigate an approximate minimum of the nonlinear function E along $p(\theta^i)$. We use a variable $\beta(\theta^i)$, which is a scalar function, to ensure that both $p(\theta^i)$ and $p(\theta^{i-1})$ are conjugate. This variable $\beta(\theta^i)$ is defined by the Fletcher-Reeves formula:

$$\beta(\theta^i) = \frac{g^T(\theta^i)g(\theta^i)}{g^T(\theta^{i-1})g(\theta^{i-1})} \tag{2}$$

The conjugate gradient methods are generally more effective than the steepest descent methods, and are just as easy to compute. Although these networks cannot converge as quickly as the Quasi-Newton method which is presented below, they do not require storage for matrices [28].

BFGS Quasi-Newton Algorithm

In addition to the steepest descent search direction, another important search technique

is the Newton-Raphson method, which is derived from the second-order Taylor series. The Newton-Raphson method for univariate optimization is based on the first and second derivative information. The Newton method is equivalent to the Newton-Raphson for multivariate optimization, and it is based on the second derivative of E, $\nabla^2 E_\theta$, which is known as the Hessian matrix.

Methods using the Newton method, which searches along the direction $-\nabla^2 E_\theta^{-1} \nabla E_\theta$, have a fast rate of local convergence, but the computation of the Hessian matrix may be a cumbersome and expensive process.

The Quasi-Newton search direction is an effective alternative, as the fast rate of convergence is maintained and computation of the Hessian matrix is not required. Instead of the Hessian matrix, $\nabla^2 E_{(\theta)}$, it uses an approximation, B_i, which is updated after each step. In order for the Hessian approximation to simulate the properties of the true Hessian matrix, the approximation needs to satisfy the secant equation.

The Broyden-Fletcher-Goldfarb-Shanno (BFGS) Quasi-Newton algorithm is based on this Quasi-Newton search direction. The BFGS formula is used to update the Hessian approximation B_i, which is formulated as [29]:

(BFGS)
$$B_{i+1} = B_i - \frac{B_i s_i s_i^T B_i}{s_i^T B_i s_i} + \frac{y_i y_i^T}{y_i^T s_i} \tag{3}$$

Levenberg-Marquadt (LM) Algorithm

As the Quasi-Newton algorithm does, the LM algorithm aims to approach the second derivatives without calculating the Hessian matrix. When the score function is formed by a sum of squares, we may represent the Hessian ($\nabla^2 f(x)$) by a Jacobian matrix, and the Hessian matrix may be approximately expressed as $H = J^T J$.

This technique involves the steepest descent method and the quadratic rules. It uses the steepest descent to approach a minimum, then switches to the quadratic rule. The gradient may be defined as:

$$g(\theta^i) = J^T e \tag{4}$$

where e is the network error vector, which is equal to the difference between the network actual and desired outputs ($e = d_m - y_m$). The LM algorithm is defined as:

$$\theta^{i+1} = \theta^i - \lambda \left[H + \mu I \right]^{-1} g(\theta^i) \tag{5}$$

where I represents the identity matrix. When the variable equals zero, this equation is used for the BFGS Quasi-Newton BP. When it becomes very large, it may be considered a gradient descent BP with small step sizes. This optimization method outperforms gradient descent and conjugate gradient methods for medium sized problems [29].

Discussion

Currently, there are no algorithms that guarantee a global solution for general nonlinear optimization problems, such as those we encountered in the neural network training processes. It is inevitable that all algorithms in nonlinear optimization will suffer from local optima problems. When the true global optimum is not available, we must use the currently available optimization methods to obtain the best local optima [28].

The most widely used optimization method is a gradient steepest descent technique, which is known as the gradient descent BP algorithm. As it is the most basic BP algorithm, it has problems of slow convergence, inefficiency and a lack of robustness.

Recall that there is a learning rate involved with this method, and it is very sensitive to the choice of learning rate. Larger learning rates may lead to network oscillation in the weight space, while smaller ones may take more time for the learning process [28].

The common modification to the gradient descent BP to reduce oscillation in the weight space is to add a momentum parameter in the weight updating equation. This method is called the gradient descent momentum BP. The momentum parameter is proportional to the latest weight change, and it will penalize networks with large weights [28].

In addition to these standard BP algorithms, there are several other optimization methods for neural network training. As second-order methods, the BFGS Quasi-Newton and Levenberg-Marquadt methods are more efficient due to their faster convergence, robustness and ability to locate good local optima [28]. However, as stated above, these two algorithms require more computation during each step, particularly the BFGS Quasi-Newton BP. Thus, they may take more time when they are applied to networks with more parameters.

The conjugate gradient descent BP with Fletcher-Reeves updates is well suited to large nonlinear optimization problems. This is because only the objective function and the gradient need to be evaluated, during each training iteration. It also requires less storage, as no matrix operations are performed [29].

It follows that there are always trade-offs in between the quality of information, and the resources required to calculate the information, such as time and memory. No one method is absolutely superior to the others, so we must select the appropriate method according the specific task [20].

3.3 Neural Networks used in the Protein Secondary Structure Prediction

Neural networks have been used in protein prediction for about twenty years, and this prior work focussed on predicting the secondary structures of proteins. In order to evaluate the accuracy of these methods, they all employ the three state accuracy, or so-

called Q3 scores. Here, the Q3 scores refer to the proportion of correctly predicted residues in the three sub-structures, the helix, the strand and the coil [30].

The earliest use of neural networks for protein secondary structure prediction was in 1988, and it was introduced by Qian and Sejnowski. They employed a fully connected multi-layer feed-forward neural network. The neural network has only one hidden layer with 40 nodes. The input of the neural network used a sliding window consisting of 13 consecutive amino acids. The neural network aims to predict the secondary structure of the amino acid, which resides in the middle of the window. The output layer has 3 nodes, each representing a class of the protein secondary structure. This neural network was trained with the standard back-propagation algorithm, and the Q3 scores achieved 64.3% [31] [32] [33]. The PHD (Profile network from HeiDelberg) and PHDpsi are two popular methods used for protein secondary structure prediction. They share the same basic neural network. The PHD method combines database searching and prediction from multiple alignment predictions by neural networks. The multiple alignment prediction refers to the process to explore whether the sequences have an evolutionary relationship. The multiple sequence alignment is converted into profiles, and used as the input of a neural network. This method has Q3 scores of over 70%. The more recent PHDpsi method exploits this improvement, by introducing increased protein data sizes. As well, it uses the PSI-BLAST to find the more distantly related sequences, which makes the multiple alignments more reliable [34] [35]. The PSI-BLAST (Position-Specific Iterated BLAST) aids users to find the distant evolutionary relationships between amino acid sequences [36].

Another method is known as PSIPRED. Similar to PHDpsi, it uses the PSI-BLAST to invoke more sequence information to the given query sequence, and the position-specific sequence profiles are directly used as the neural network input. The first hidden layer of the neural network uses a size 15 window to slide the sequence profile, and produces an initial prediction. The output layer filters the raw prediction results and makes the final prediction. The widespread use of this method is due to its high

Q3 score, which is 76.6%, ease of use, and high overall prediction performance [32] [34] [37].

The idea behind the PROF (king) method is to use as much relevant information as possible. Thus, it follows a complicated scheme that it uses multi-staged classifiers as the inputs to a three-layer neural network. The SSpro method introduces long-range interactions into the prediction, and employs a bi-directional recurrent neural network (BRNN). The long-range interaction refers to the two contacted amino acids whose sequence separation is larger than 4 amino acids [38]. In addition to this basic sliding window, the SSpro method employs two other windows which slide in from opposite sides at each basic window sliding position. The Q3 score of this method is 76.3%. The Porter is a development of the SSpro toolkit. It improves the SSpro Q3 scores to 80% by using more extensive training set and improving incorporation of long-range distance information [34].

In addition to those methods exclusively based on neural networks, several methods partly based on neural networks were developed as well, such as PSSP, SAM-T and YASPIN. The PSSP method combines neural networks with the K-Nearest-Neighbour (KNN), which is a learning method to classify objects based on closest training examples in the feature space. Both the SAM-T and YASPIN methods combine neural networks with Hidden Markov Models (HMM), which is a statistical model, to predict the secondary structures [39] [40] [41].

According to the literature review, as far as we are aware, this is the first work that aims to predict proteins' 3D shapes directly from their amino acid sequences, through the use of neural networks. Although the amino acid sequences are involved in the above secondary structure prediction methods, they are used to establish relationships between subsequences and proteins' local structures. Thus, these methods usually use sliding windows to go through amino acid sequences to establish the relationship, without representing the amino acid sequences. However, in our thesis, we aim to predict the proteins' 3D shapes, which concern the global structures. Thus, we need an

algorithm to represent the amino acid sequences, which compose the protein. As well, we need to represent the 3D shape, before the neural network training.

3.4 Summary

In this chapter, we introduce artificial neural networks, which are defined by their architectures, parameters involved in the neural networks and optimization algorithms. We focus our attention on supervised learning. Here, the neural network is trained by known inputs and outputs, and the prediction for unknown inputs is based on the learned relationships. The learning processes aim to minimize the difference between the actual output and the desired output, and the difference is measured by the score function. Thus, the learning process is essentially a process of optimization. We detailed the five optimization algorithms we used in this research. Together with the introduction on the neural network operations, we presented earlier work on the use of neural network on the prediction of protein secondary structure prediction. As far as our understanding goes, this thesis represents the first study that aims to predict proteins' 3D shapes from their amino acid sequences, as will be discussed next.

PART II

PROTEIN STRUCTURE PREDICTION

Chapter 4

Methodology

In this chapter, we discuss the methodology we followed. First, the process of experimentation is introduced. Next, the experiment will be detailed by discussing the data pre-processing steps and the neural network design. Further, we will present the data we used in our experiments.

4.1 **Overview of Protein 3D Shape Prediction Neural Networks**

Recall that the aim of this research is to predict the 3D shape of a query protein directly from its amino acid sequence, using neural networks. The neural networks work well on continuous-valued data and establish the complex relationship between its inputs and outputs, such as the relationship existed between amino acid sequences and 3D shapes. The prediction is based on the observation that proteins with similar amino acid sequences usually have similar 3D shapes [6] [7] In reality, for the purposes of predicting the 3D shape of an unknown protein, the neural networks should be trained by many known proteins to learn the relationships existed between amino acid sequences and 3D shapes

As stated earlier, in order for neural networks to learn the relationship between amino acid sequences and 3D shapes, both amino acid sequences and 3D shapes must be represented by descriptors. That is, this research concerns a supervised learning, or classification, setting. These descriptors should be independent of the length of amino acid sequences or the complexity of 3D shapes. To this end, we proceed to create amino acid sequence descriptors and 3D shape descriptors for known proteins. Before the amino acid sequence descriptors may be used as inputs of neural networks and 3D shape descriptors may be used as outputs, they both need to be normalized. Since the entries for both descriptors are widely distributed, the neural networks may treat the large entries with more importance, which may cause biases during the training processes. Thus, we need to normalize them into small ranges, in order to reduce the biases and improve the accuracy of prediction [21]. Once the neural network has been trained, it will be used to predict the 3D shape descriptors of a query protein, according to its amino acid sequence descriptors. The predicted 3D shape descriptors are used to search and retrieve the known protein which has the closest 3D shapes with the query protein. Finally, we will evaluate the accuracy of prediction by verifying whether the query protein and the retrieved protein belong to the same protein family, since proteins in the same family generally share similar 3D shapes [6] [7].

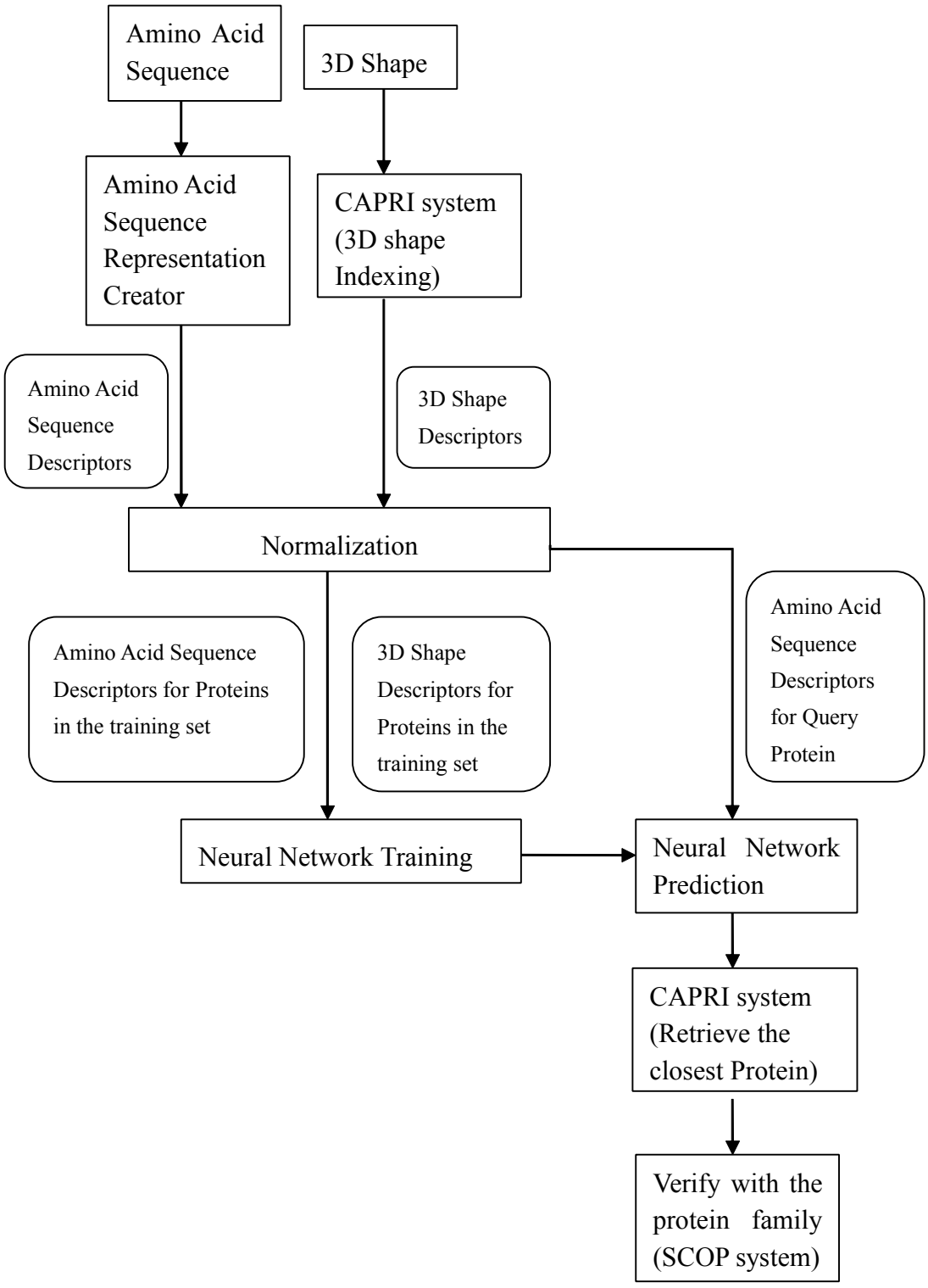


Figure 8. Experimental Design

The methodology is outlined in Figure 8 and Algorithm 1, and it will be explained in detail in the following sections.

Algorithm of Experimental Methodology

Input: Amino acid sequences and 3D shapes of proteins

Output: The performance of the prediction

1. Create the amino acid sequence descriptors
2. Create the 3D shape descriptors
3. Normalize the amino acid sequence descriptors and 3D shape descriptors
4. Train the neural network by the normalized amino acid sequence descriptors and 3D shape descriptors
5. Predict 3D shape descriptors of the query protein based on its amino acid sequence descriptors
6. Using the predicted 3D shape descriptors to search retrieve the known proteins that have the closest 3D shapes with the query protein
7. Verify whether the retrieved protein is from the same protein family with the query protein

Algorithm 1. Algorithm for the Experimental Methodology

4.2 Data Pre-processing

This section details our amino acid sequence representation creator and introduces the CAPRI system, which is used for protein 3D shape indexing and retrieval.

4.2.1 Amino Acid Sequence Representation Creator

The calculation of amino acid sequence descriptors is performed by the Amino Acid Sequence Representation Creator (AASRC) algorithm, implemented by means of a Java program. The AASRC algorithm first reads the amino acid sequence files, and stores the protein name and the number of constructing amino acid sequences into corresponding variables. It then sorts the constructing amino acid sequences into arrays. Recall that the amino acid sequence descriptors should be independent from the

length of the sequence. Although the proteins having similar 3D shapes tend to have similar amino acid sequences, it does not mean that they are identical. Indeed, the similarity here refers to the fact that those proteins share a high number of similar subsequences. Due to mutation and evolution, uncorrelated outliers may exist in between similar subsequences. As well, similar subsequences may have, for a given position in the subsequence, distinct amino acids, but those amino acids may be highly similar from their chemical properties. Thus, we need a metric that may describe the similarity between two amino acids from their physicochemical point of view. The BLOSUM Matrix, as introduced in Section 2.1.2, reflects the similarity between two amino acids from a physico-chemical perspective. We calculate the correlation between two amino acids at a specific function distance based on the BLOSUM Matrix. The function distance is in terms of the number of amino acids.

In order to calculate amino acid sequence descriptors, we first consider the case of a protein composed of only one sequence. The calculation is based on five function distances. For a given amino acid within the sequence, we compute the similarity between that particular amino acid and the next one. The similarity is evaluated by searching corresponding scores allocated to these two amino acids in the BLOSUM Matrix. After processing the entire amino acid sequence, a distribution (normalized histogram) of each similarity (the frequency of two amino acids having the same similarity) is calculated. As seen from the BLOSUM Matrix there are 15 different measures of similarity, so the distribution of similarity for amino acid sequences with separation of zero amino acid would be on 15 channels. The calculation is repeated, but this time with a separation of one amino acid. Subsequently, it is repeated with a separation of two amino acids, a separation of three amino acids and a separation of four amino acids. The distributions of similarity on different function distances are concatenated to be used as the descriptor.

When a protein has more than one sequence, we perform the same calculation on each sequence. After finishing the calculations on all sequences, we obtain the similarity

distribution at different function distances for each sequence, and then calculate the distribution for this protein. That is, for this protein the frequencies of a certain similarity measure at a certain function distance calculated for sequences will be aggregated, and those frequencies are concatenated to form the descriptor. This calculation process is summarized in Algorithm 2.

Consider the protein 1sea, Tumor Suppressor P53, as an example of this calculation process. Protein 1sae is composed of four amino acid sequences, as shown in the Figure 9.

```
>1SAE:A|PDBID|CHAIN|SEQUENCE
KKKPLDGEYFTLQIRGRERFEMFRELNEALELKDAQAGKEPG
>1SAE:B|PDBID|CHAIN|SEQUENCE
KKKPLDGEYFTLQIRGRERFEMFRELNEALELKDAQAGKEPG
>1SAE:C|PDBID|CHAIN|SEQUENCE
KKKPLDGEYFTLQIRGRERFEMFRELNEALELKDAQAGKEPG
>1SAE:D|PDBID|CHAIN|SEQUENCE
KKKPLDGEYFTLQIRGRERFEMFRELNEALELKDAQAGKEPG
```

Figure 9. Amino acid sequences of protein 1sae

Let us consider the first amino acid sequence. We focus our attention on the first ten amino acids in its first sequence in order to outline the calculation, namely:

KKKPLDGEYF

For the separation of zero amino acid, we consider the BLOSUM Matrix for the correlation between an amino acid and the one beside it on the right, such as the first amino acid ‘K’ and the second amino acid ‘K’.

As observed from the BLOSUM Matrix, there are 15 measures of the similarity. We proceed to calculate how many times a certain similarity measure appears. For example, when the similarity measures equal to -2, the frequency will be 2, which is obtained by ‘G’ and ‘E’, and ‘E’ and ‘Y’. The frequency of similarity measures based

on the results obtained is shown in Table 2.

Table 1. Similarity between amino acids with a separation of zero amino acid

An amino acid	An amino acid next to it	Similarity
K	K	5
K	K	5
K	P	-1
P	L	-3
L	D	-4
D	G	-1
G	E	-2
E	Y	-2
Y	F	3

Table 2. Frequency of Similarity between amino acids with a separation of zero amino acid

Level	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	11
Frequency	1	1	2	2	0	0	0	1	0	2	0	0	0	0	0

Table 3. Frequency of similarity for the first sequence of protein 1sea

Separation	Level	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	11
0	Frequency	1	10	11	9	6	1	0	1	0	2	0	0	0	0	0
1	Frequency	3	7	8	8	5	2	2	0	2	3	0	0	0	0	0
2	Frequency	0	7	10	10	5	2	1	0	0	3	1	0	0	0	0
3	Frequency	2	8	5	12	4	1	2	0	1	2	1	0	0	0	0
4	Frequency	1	11	7	10	2	2	3	0	0	1	0	0	0	0	0

When we consider a separation of one amino acid, the calculation is performed on an amino acid and the second after (i.e. with one amino acid between them, such as the first amino acid ‘K’ and the third amino acid ‘K’). For a separation of two amino acids, the separation will be three away, namely the first amino acid ‘K’ and the forth amino acid ‘P’. The same holds for the separation of three amino acids and the separation of four amino acids.

In the case where we perform the calculation only on the first amino acid sequence of protein 1sae, we obtain the frequency of similarity on each distance sequence, as shown in Table 3.

Algorithm to Calculate the Amino Acid Sequence Descriptors

Input: The amino acid sequences of a protein

Output: The amino acid sequence descriptors

1. Read the first amino acid sequence of the protein
2. Calculate the similarity between a given amino acid and the next one, according to the BLOSUM Matrix
3. After processing step 2 on the whole sequence, derive the distribution of the similarities
4. Repeat the step 2,3 with a separation of one amino acid
5. Repeat the step 2,3 with a separation of two amino acid
6. Repeat the step 2,3 with a separation of three amino acid
7. Repeat the step 2,3 with a separation of four amino acid
8. Repeat the step 2-8 on the other sequences
9. Derive the total frequency of a certain similarity measure at a certain distance separation
10. Concatenate the distributions

Algorithm 2. Calculation of the Amino Acid Sequence Descriptors

Next, the computation of amino acid sequence descriptors for protein 1sea is performed on each sequence. We then obtain the similarity distribution between two amino acids for different function distance on each sequence. The results are shown in the Table 4.

Recall that, for proteins having more than one sequence, we produce the frequencies obtained from the different sequences. That is, for the different sequences we calculate the total frequency at the same separation (in terms of amino acids) and for the same similarity measure. For example, for the similarity measure -4 with a separation of zero amino acid, we aggregate the frequencies of similarity on the four sequences, which results in 4 (=1+1+1+1). After calculation we obtain the results shown in Table 5,

which are the amino acid sequence descriptors of protein 1sae. The frequencies on these five function distances are concatenated to be used as the amino acid sequence descriptors. This process of calculation is described in Algorithm 2.

Table 4. Frequency of Similarity of Each Sequence in Protein 1sae

	Separation	Level	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	11
First Sequence	0	Freq.	1	10	11	9	6	1	0	1	0	2	0	0	0	0	0
	1	Freq.	3	7	8	8	5	2	2	0	2	3	0	0	0	0	0
	2	Freq.	0	7	10	10	5	2	1	0	0	3	1	0	0	0	0
	3	Freq.	2	8	5	12	4	1	2	0	1	2	1	0	0	0	0
	4	Freq.	1	11	7	10	2	2	3	0	0	1	0	0	0	0	0
Second Sequence	0	Freq.	1	10	11	9	6	1	0	1	0	2	0	0	0	0	0
	1	Freq.	3	7	8	8	5	2	2	0	2	3	0	0	0	0	0
	2	Freq.	0	7	10	10	5	2	1	0	0	3	1	0	0	0	0
	3	Freq.	2	8	5	12	4	1	2	0	1	2	1	0	0	0	0
	4	Freq.	1	11	17	10	2	2	3	0	0	1	0	0	0	0	0
Third Sequence	0	Freq.	1	10	11	9	6	1	0	1	0	2	0	0	0	0	0
	1	Freq.	3	7	8	8	5	2	2	0	2	3	0	0	0	0	0
	2	Freq.	0	7	10	10	5	2	1	0	0	3	1	0	0	0	0
	3	Freq.	2	8	5	12	4	1	2	0	1	2	1	0	0	0	0
	4	Freq.	1	11	7	10	2	2	3	0	0	1	0	0	0	0	0
Fourth Sequence	0	Freq.	1	10	11	9	6	1	0	1	0	2	0	0	0	0	0
	1	Freq.	3	7	8	8	5	2	2	0	2	3	0	0	0	0	0
	2	Freq.	0	7	10	10	5	2	1	0	0	3	1	0	0	0	0
	3	Freq.	2	8	5	12	4	1	2	0	1	2	1	0	0	0	0
	4	Freq.	1	11	7	10	2	2	3	0	0	1	0	0	0	0	0

Thus, regardless of how many amino acid sequences a protein is composed of, its sequences are described by 75 values (=15*5). The amino acid sequence descriptors are on 75 channels. The pseudocode of the amino acid descriptor algorithm is presented in Algorithm 3.

Table 5. Frequency of Similarity of Protein 1sea

Separation	Level	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	11
0	Frequency	4	40	44	36	24	4	0	4	0	8	0	0	0	0	0
1	Frequency	12	28	32	32	20	8	8	0	8	12	0	0	0	0	0
2	Frequency	0	28	40	40	20	8	4	0	0	12	4	0	0	0	0
3	Frequency	8	32	20	48	16	4	8	0	4	8	4	0	0	0	0
4	Frequency	4	44	28	40	8	8	12	0	0	4	0	0	0	0	0

Algorithm Calculate Amino Acid Sequence Descriptor

Input: A proteinFile from the ProteinDataBankDatabases

Output: The AminoAcidSequenceDescriptor of proteinFile

1. read (*proteinFile*);

2. **for each** *protein* in *proteinFile* **do**

Chain.**add**(separateProteinIntoChains(*Protein*));

num.**add**(computeNumberOfChains(*Protein*));

3. **for each** protein in proteinFile **do**

for each sequence in protein **do**

for each function distance **do**

for each amino acid in the sequence **do**

fDCal=computeCorrelationBetweenTwoAminoAcids (*Chain*);

fDistance= computeDistributionofCorrelation(*fDCal*);

functionalDistanceFile=computeTotalDistributionofCorrelation (*fDistance*);

4. write(*functionalDistanceFile*);

5. Return *AminoAcidSequenceDescriptor*

End **CalculateAminoAcidSequenceDescriptor**

Algorithm 3. Pseudocode for the Amino Acid Sequence Representation Creator

4.2.2 CAPRI System

The CAPRI system is used for both protein 3D shape indexing and retrieval. It creates 3D signatures of protein structure, and identifies proteins having closest structures.

3D Protein Structure Index

The CAPRI system employs a global approach to create 3D signatures of protein structures, which helps reduce processing time and increases discrimination [42]. In order to depict a protein's arbitrary location and pose in space, the signature or index is translation, scale and rotation invariant [43]. The algorithm may be summarized as follows and shown in Algorithm 4 [42].

First, the surface of the protein 3D shape is tessellated with triangular simplices, and then the barycentre and the tensor of inertia, a symmetrical matrix, of the protein are calculated. A rotation invariant frame for the protein 3D shape may then be constituted by the Eigen vectors; that is to say, the Eigen vectors are invariant, even though the original 3D shape of the protein rotates or translates in the space. The axes of the reference frame are identified by their corresponding Eigen values. For example, the first axis corresponds to the highest Eigen value while the second axis corresponds to the second highest Eigen value [42]. Once the reference framework has been produced, the probability density functions associated with the radial distribution of the triangular simplices, the angular distribution of the triangular simplices relative to the Eigen vector corresponding to the largest Eigen value, and the angular distribution of the triangular simplices relative to the Eigen vector corresponding to the second largest Eigen value are calculated. These three distributions are concatenated together to constitute the descriptors of the 3D protein shapes. The radial distribution refers to the distribution of the norm of the vector, which starts from the barycentre of the 3D shape and ends at the barycentre of each triangular simplex. The angular distributions refer to the distribution of the angles between the previous vectors and the given reference axis of the Eigen frame [42].

Algorithm Calculate3DIndexDescriptor

Input: A proteinFile from the ProteinDataBankDatabases

Output: The *3DIndexDescriptor* of *proteinFile*

1. read(*proteinFile*);
2. *triangulatedProtein*=triangulate(*proteinFile*);
3. **for each** *triangle* in *triangulatedProtein* **do**

    triCentres.**add**(computeTriCentre(*triangle*));

    *weight*s.**add**(computeWeight(*triangle*));
4. *barycentre*=computeBarycentre(*triCentres*);
5. *tensorInertia*=computeTensorInertia(*barycentre*,

    triCentres, *weights*);
6. *principalComponents*=jacobi(*tensorInertia*);
7. sortByPrincipalValues(*principalComponents*);
8. **for each** *triCentre* **in** *triCentres* **do**

    cords.**add**(computeCord(*barycentre*, *triCentre, principalComponents*));
9. **for each** *cord* **in** *cords* **do**

    angles.**add**(computeAngles(*cord*,

        principalComponents));

        radii.**add**(computeRadii(*cord*));
10. *histogramsAngles*=computeHistogramsAngles(*angles*);
11. *histogramRadii*=computeHistogramsRadii(*radii*);
12. normalisedAnglesHistograms(*histogramsAngles*);
13. normalisedRadiiHistogram(*histogramRadii*);
14. write(*histogramsAngles*);
15. write(*histogramRadii*);
16. Return *3DIndexDescriptor*

Algorithm 4. Pseudo code of the algorithm for the calculation of the shape index or descriptor associated with the topology and the envelopes [42]

Measuring Signature Similarity

Similarity searches are intended to find the proteins which have the most similar structures to a given query protein. The search process is an iterative or spiral approach. An index may be viewed as a point in a N-dimensional space, where N depends on the size of the descriptor. The prototype chose by the user may be viewed as a point in the space, and then the search engine will determine which points are the closest to the prototype. In the CAPRI system, the distance between two protein descriptors is measured by using the Euclidian distance [43].

$$d = \sqrt{\sum_{i=1}^{N} (x_i - y_i)} \tag{6}$$

For a given representation, the CAPRI system may conduct the search and rank all the protein structures, in a fraction of a second [42].

Evaluations of the CAPRI system have shown that it may locate similar protein structures between families, with the results validated by the SCOP system [43]. Overall, the CAPRI system performs similarity search very quickly, provides high precision, and retrieves the most pertinent results with a minimal number of outliers [42] [43].

For this research, the 3D shape descriptors predicted by the neural network will be used by the CAPRI system to retrieve the proteins with closest 3D shapes.

4.2.3 Rescaling

Before the amino acid sequence descriptors and 3D shape descriptors are presented to the network, they must be rescaled, since they are distributed in wide ranges. The widely distributed descriptors could bias the learning process of the neural network, because networks learn the relationships between their inputs and outputs only by their values [21]. That is to say, even though the values used to compose the descriptor are

on the same level of importance, the network may consider the large value with higher level of importance. Thus, in order to reduce the bias caused by those values and increase the accuracy of neural networks, it is necessary to normalize inputs and outputs of the neural network.

Rescaling refers to the process by which attribute data are scaled into a small specified range, such as -1.0 to 1.0, or 0.0 to 1.0 [21]. Processing each feature into the scale helps the network to train faster. It is particularly useful for modelling applications, where inputs or outputs are generally on widely different scale. In addition to transform the data into a more suitable range for network training, normalization also helps reduce the chances that the network will be trapped in local optima [44]. Many methods are used for normalization. In this project, we used the min-max normalization, because it precisely preserves all relationships in the data [44].

Min-max normalization preserves the relationships of the original data values, by performing a linear transformation of the original data. Assume an attribute A, whose minimum and maximum values are presented by $\min_A$ and $\max_A$. The min-max normalization will map a value, v, of A to v', which is in the range [$new\_\min_A$, $new\_\max_A$]. This mapping computation is done by the equation 32 [21]:

$$v' = \frac{v - \min_A}{\max_A - \min_A}(new\_\max_A - new\_\min_A) + new\_\min_A \qquad (7)$$

For this project, the amino acid sequence descriptors and 3D shape descriptors were normalized into the range [-1, 1] by using the 'mapminmax' function, which is provided in Matlab.

4.3 Neural Network Modelling

In this research, we explore the performance of neural networks using different architectures and optimization algorithms. In the following section we present the

designs of neural network architectures, and then we discuss the algorithms used for neural network training. We use the Matlab Neural Network Toolbox for neural network modelling and training. All the experiments were performed using a workstation with Dual Core with 2 processors, 4 GB RAM and 500 GB disk space.

4.3.1 Different Architectures of Neural Networks

As indicated earlier, it follows that the performance of neural networks is highly influenced by their structures. In this section we explain the different architectures of neural networks that we explored for protein shape prediction.

Input Nodes: The amino acid sequence descriptor of a protein is described by 75 values. As we want to set each value to a node in the input layer of the neural network, there are 75 nodes in the input layer.

Output Node: The 3D shape descriptor of a protein is formed of 120 entries. Thus, as with the amino acid sequence descriptor, there are 120 nodes in the output layer of the neural network.

The neural networks may have a varying number of hidden layers. When a network has one hidden layer, we explore the influences caused by the number of nodes in that layer, in order to learn whether an increase in the number of nodes will improve the performance of the network. We have two situations for networks with one hidden layer: the network has 18 nodes in its hidden layer, or it has 40 nodes in its hidden layer. These numbers were experimentally determined. When the neural network has 18 nodes in its hidden layer, it would have 3648 weights while when the neural network has 40 hidden nodes, it would have 7960 weights.

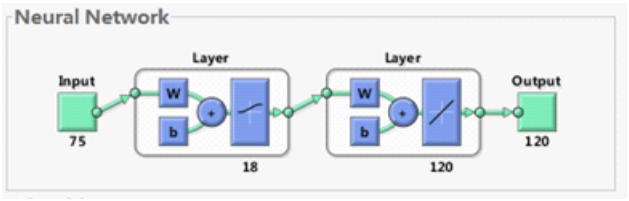


Figure 10. Neural Networks with One Hidden Layer with 18 Nodes

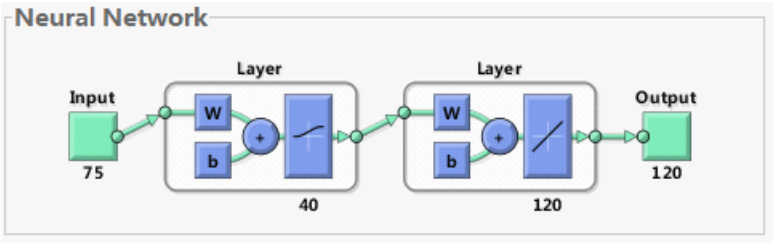


Figure 11. Neural Networks with One Hidden Layer with 40 Nodes

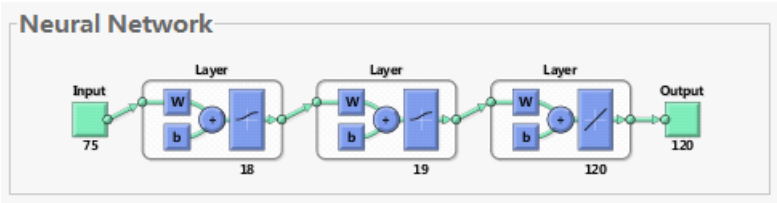


Figure 12. Neural Networks with Two Hidden Layers

The other scenario we explore is where the neural network has two hidden layers, with 18 nodes in the first hidden layer and 19 nodes in the second for which correspond 4129 weights. Again, these values were experimentally obtained.

Recall that the behaviour of neural networks is related to weights and biases, and the transfer functions. Nodes in neural networks are linked by edges, and each edge between two nodes is related to a weight and a bias. To improve the performance of the networks, the weights and biases are updated during each training epoch. In this research, random initial weights and biases were used.

4.3.2 Different Optimization Algorithms used by Neural Networks

Recall that our experiments explore five optimization algorithms. They are the Gradient Descent BP, the Gradient Descent BP with momentum, the conjugate Gradient algorithm with Fletcher-Reeves, the BFGS Quasi-Newton algorithm and Levenberg-Marquadt (LM) algorithm.

Table 6. Datasets

	Number of Families	Smallest Family Size	Biggest Family Size
Dataset 1: Two Class Problem with Class Imbalance	2	7	79
Dataset 2: Multi-class Problem with Few Training Examples per Class	11	1	8
Dataset 3a:Varying Family Sizes and Similar Amino Acid Sequences	23	1	64
Dataset 3b:Varying Family Sizes and Dissimilar Amino Acid Sequences	9	2	35

We conduct three sets of experiments, in order to explore the performance of the various neural networks in three different scenarios. We first aim to explore the performance of neural networks when training on only two protein families with a large difference in sizes, i.e. where we have one large and one small family Our aim here is to validate whether we are able to correctly find the 3D shapes of the minority (small) family members in an imbalanced setting. Here, the members of the two families have different amino acid sequences and 3D shapes. We are thus especially interested in learning whether the neural networks are able to learn the 3D shapes of the minority class (the small family). Second, we explore the performance of neural networks on proteins that are only from small classes, i.e. where the class member size is between one and eight. Proteins in the same class are from the same protein family. That is to say, here the neural networks have fewer examples to learn the relationship between amino acid sequences and 3D shapes, and there are a number of different classes to learn. Thus, we are interested in determining whether the neural network succeeds when we have a number of such examples. Thirdly, the performance of neural

networks is tested on proteins with diverse family sizes. Also, we study the cases where proteins having amino acid sequences presenting a lesser degree of similarity, even though they belong to the same family. The following observation is noteworthy. In this study, we are interested in finding the relationships between amino acid sequences and 3D shapes of proteins. The family membership is thus *unknown during training*, but we use it in our validation stage. Together with the above three research questions, the performance of the five optimization algorithms on this work is also explored.

4.4 The Datasets and Research Tasks

Table 7. Two families used in Dataset 1

Family Name	Number of Proteins	Query Protein	Proteins
P22 Tailspike	7	1tyv	1tyv, 1tsp, 1qrb, 1qa1, 1qq1, 1clw, 1qa2
Phage Lysozyme	79	142l	142l, 143l, 1l89, 1l85, 1ovj, 2l78, 1l90, 1l84, 1dyf, 1l91, 1owy, 1cuq, 231l, 1dya, 1d3f, 1l74, 139l, 1l77, 1l69, 248l, 1l66, 1l96, 155l, 1d2y 1cv0, 1lyh, 196l, 215l, 1l67, 159l, 1l94, 1l79, 1l64, 165l, 112l, 219l, 1lyf, 182l, 1li6, 1l68, 1qtv, 1l59, 229l, 228l, 1l63, 2b70, 1l57, 115l, 110l, 1ovk, 147l, 185l, 1l65, 1l95, 1qsb, 1l81, 1l29, 157l, 243l, 162l, 1b6i, 190l, 1llh, 221l, 224l, 1c62, 1ctw, 1owz, 120l, 1cu5, 1qsq, 158l, 1lye, 1c66, 1c61, 1l26, 1l54, 138l, 141l, 1cx7

In order to explore the neural network performance under the three situations we stated earlier, we employ three datasets for the neural networks training. Recall that proteins in the same class belong to the same protein family. The first dataset is formed of proteins from two classes, each with a different number of members. The second dataset is composed of proteins with fewer members per class. Here, for each pair of relationship between amino acid sequences and their corresponding 3D shapes, there are on average four members for neural network training. The third dataset contains

200 proteins. The amino acid sequences of some proteins present a wider variability, despite the fact that they correspond to very similar 3D shapes. This dataset also includes imbalanced data, in that we include large, medium and small sized families.

Table 8. Dataset 2: Multi-class problem with few training examples

Family Name	Number of proteins	Query Protein	Proteins
Bacterial AB5 toxins, B-subunits	4	1bcp	1bcp, 1prt, 1prt, 1pto
Iron-dependent repressor protein	1	1bi3	1bi3
Carbonic anhydrase	4	1fsn	1fsn, 1fsq, 1fr7, 1fsr
Fluorescent proteins	4	1ggx	1ggx, 1zgo, 1zgp, 1g7k
Heme-dependent catalases	4	1gwe	1gwe, 1hbz, 1gwf, 1gwh
Ferritin	1	1h96	1h96
Globins	8	1lfq	1lfq, 1lfv, 1hco, 2hco, 1r1y, 1gzx, 1ye2, 1y7z
Ligand-gated protein channel	4	1qfg	1qfg, 1fi1, 1qjq, 1qff
P22 tailspike protein	4	1tyv	1tyv, 1tsp, 1qrb, 1qa1
Catalase-peroxidase KatG	4	2b2s	2b2s, 2b2r, 2dv2, 2b2q
Phage Lysozyme	4	142l	142l, 143l, 1l89, 1l85

4.4.1 Task 1: Two class problem with class imbalance

Recall that our first research question is to study how the neural networks perform when trained against two classes, i.e. where we have two sets of amino acid sequences and 3D shapes with very different sizes and properties. In this dataset, the smaller family has seven members, while the larger is constituted of 79 members.

4.4.2 Task 2: Multi-class problem with few training examples per class

To explore whether neural networks perform well on proteins with less training examples, we experimented on the dataset composed of multiple classes with very few training examples. In this case, the 10 classes correspond to families and the 32

training examples are the proteins that are within each class. In most classes, there are 4 protein members.

4.4.3 Task 3: Diversity in terms of amino acid sequences and class membership

In order to explore the performance of neural networks on more diverse proteins, we conduct experiments with a third dataset, which is composed by 200 proteins. When proteins are from the same protein family, their amino acid sequences may present similarities or dissimilarities within each other. The similarity between amino acid sequences is presented by similar frequency proportion of composing amino acids, and we determine it by using the angular distance, which will be introduced next. Recall that proteins in the same family generally have similar 3D shapes [6] [7]. Thus, proteins with similar 3D shapes may be divided into two groups, namely proteins with similar amino acid sequences for similar 3D shapes, and proteins with relatively dissimilar amino acid sequences for very similar 3D shapes. This dataset is demonstrated in the Table 9 and 10.

To determine whether proteins are similar, we use the angular distance. Although it is called the angular distance, it is a distance measure, and it is widely used for Text Document cluster analysis. It employs the direction of the two pattern vectors [26] to represent the cosine angle between the unit vectors. Thus, the value lies between -1 and 1 same as the range of the cosine angle. The higher value of this function means that these two objects are very similar to one another. Although the angle is measured, it gives the linear distance between data points. The similarity and distance measure between x_i and x_j are given below:

$$s_{x_i,x_j} = \frac{\langle x_i, x_j \rangle}{\|x_i\| \|x_j\|} \tag{8}$$

$$d_{x_i,x_j} = 1 - \frac{\langle x_i, x_j \rangle}{\|x_i\| \|x_j\|} \tag{9}$$

Since the angular distance does not depend on the length ($s_{x_i,x_j} = s_{\alpha x_i,x_j}, \forall \alpha \succ 0$ [26]), the documents with similar term frequency proportion but different totals are considered as similar. Thus, even though amino acid sequences have different lengths, sequences composed by similar frequency proportion of amino acids are treated as similar.

Table 9. Dataset 3a: Varying family sizes and similar amino acid sequences

Family Name	Number of Proteins	Proteins
DNA polymerase processivity factor(Bacteriophage T4)	1	1czd
DNA polymerase processivity factor(Bacteriophage RB69)	2	1b8h, 1b77
RUNT domain	1	1e50
WD40-repeat	1	2ce8
Zinc finger based beta-beta-alpha motif	1	1sna
Nucleoside phosphorylase/phoshoribosyltransferase N-terminal domain	1	1o17
GroES	2	2c7c, 2c7d
SH2 domain	1	1a81
HtrA-like serine proteases	1	1y8t
Recombination protein RecR	1	1vdd
Complement control module/SCR domain	3	1g40, 1g44, 1y8e,
Ferritin	2	1h96,1lb3
Sermidine synthase	1	1uir
Glycosyl transferases group 1	1	2iv7
2,3-Bisphosphoglycerate-independent phosphoglycerate mutase, substrate-binding domain	1	1ejj
Tyrosyl-DNA phosphodiesterase TDP1	1	1q32
V set domains(antibody variable domain-like)	3	1rum, 1ncw, 1rup
P53 tetramerization domain	8	1sak, 1sal, 1sai, 1sah, 1sag, 1olg, 1saj, 1saf
Calmodulin-like(Human)	1	1y0v
Calmodulin-like(Cow)	5	1xfw, 1xfu, 1xfv, 1xfz, 1xfy
PBP5 C-terminal domain-like	5	1z6f, 1nzo, 1hd8, 1nj4, 1nzu
GABA-aminotransferase-like	8	2bi3, 2bi1, 2bi2, 2bhx, 2bi5, 2bi9,2bia, 1w23
Phage Lysozyme	64	142l, 143l, 1l89, 1l85, 1ovj, 2l78, 1l90, 1l84, 1dyf, 1l91, 1owy, 1lcuq, 231l, 1dya, 1d3f, 1l74, 139l, 1l77, 1l69, 248l, 1l96, 1l66, 155l, 1d2y, 1cvo, 1lyh, 196l, 2115l, 1l67, 159l, 1l94, 1l79, 1l64, 165l, 112l, 219l, 1lyf, 182l, 1li6, 1l68, 1qtv, 1l59, 229l, 228l, 1l63, 2b70, 1l57, 115l, 110l, 1ovk, 147l, 185l, 1l65, 1l95, 1qsb, 1l81, 1l29, 157l, 243l, 162l, 1b6i, 190l, 1llh, 221l

Dataset 3a: Proteins with similar 3D shapes and similar amino acid sequences

In this dataset, proteins having a high similarity of 3D shapes are associated with a high similarity of their amino acid sequences. For each pair of relationship between amino acid sequences descriptors and 3D shape descriptors, there are a different number of proteins used as training examples, ranging from 1 to 64.

Table 10. Dataset 3b: Varying family sizes and dissimilar amino acid sequences

Family Name	Number of Proteins	Family Members
Bacterial AB5 toxin, B-subunits	3	1bcp, 1prt
		1pto
Fatty acid binding protein-like	2	2hmb
		2fw4
Carbonic anhydrase	12	1fsn, 1fsq, 1fr7, 1fsr, 1xev, 1fr4, 1fqm
		1zsa, 1if6, 2h4n, 1if8
		1ugg
Fluorescent proteins	4	1ggx, 1zgo, 1zgp
		1g7k
Heme-dependent catalases	4	1gwe, 1gwf, 1gwh
		1hbz
Globins	35	1y7c
		1xzv, 1xz5, 1xz7
		1r1y, 1gzx, 1ye2, 1y7z, 2sn2, 1y0w,1y5f, 1dxt, 1y4v, 1xxt, 1dxv, 1y0t, 1y7g, 1y22, 1y4b, 1yih, 1y4g, 1y7d, 1rq3, 1dke, 2d60, 1y35, 1y83, 1lfq, 1lfv, 2hco, 1lfy, 1lft, 2hbd, 2hbc
Ligand-gated protein channel	7	1qfg, 1qjq, 1qff, 1qkc, 2fcp
		1fil, 1fcp
P22 tailspike protein	12	1tsp, 1qrb, 1qq1, 1qrc
		1tyv, 1qa1, 1clw, 1qa3, 1qa2, 1tyx, 1tyu, 1tyw
Catalase-peroxidase KatG	7	2dv2, 2dv1, 1x7u
		2b2s, 2b2r, 2b2q, 2b2o

Database 3b: Proteins with similar 3D shapes and dissimilar amino acid sequences

In this group, proteins in the same family have similar 3D shapes, but they are associated with a lesser similarity in between their amino acid sequences. For example, in the protein family Bacterial AB5 toxin, B-subunits, proteins 1bcp, 1prt and 1pto present similar 3D shapes. Protein 1bcp and 1prt share similar amino acid sequences,

but protein 1pto has a dissimilar amino acid sequences, when compared to the others.

4.5 Summary

This chapter detailed the methods used by neural networks to predict protein structures. The first step is to pre-process the data, both the amino acid sequences and 3D shapes, so that it may be analysed by neural networks. An algorithm to compute the amino acid sequence descriptors was described, and the CAPRI system which will be applied to calculate the 3D shape descriptors was detailed. We also discussed the need for normalization. The next step se introduced is network modelling. During modelling, we explore different network architectures and optimization algorithms. The amino acid sequence descriptors calculated from the previous step serves as the input of the neural network. The 3D shape descriptors are used as the output of the network. Our experiments, as will be discussed in the next chapter, are performed on networks having one or two hidden layers. The results of the evaluation of the proposed protein structure prediction neural networks are highlighted in the next chapter.

Chapter 5

Experimental Evaluation

This chapter provides the results of a comparative study of the performance of our methodology when aiming to predict the 3D shapes of proteins. We also introduce the evaluation method used to assess the accuracy of prediction when using different neural network architectures and optimization algorithms.

5.1 Criteria of Evaluation

In order to evaluate the performance of neural networks in this research, researchers often use the network error rates (MSE), which we have introduced in Section 3.2.1, and the time required to train neural networks. The time is measured in seconds. Since the purpose of the research is to study the usefulness of neural networks to predict the 3D protein shapes, it is not enough to evaluate our experiments only by these performance measures. We also need to verify the accuracy of the predicted 3D shape descriptors, in terms of family membership. To this end, we also verify whether the predicted 3D shape descriptors of query proteins are accurate enough to allow the CAPRI system to retrieve their family members with the SCOP system, which has been introduced in Section 2.1.3.

With the purpose of estimating the generalization and reliability of our prediction, we employ a standard ten-fold cross validation method, in which the data is randomly partitioned into ten approximately equal parts [21]. During the neural networks training procedure, each partition is used as a testing set and the remainder is used as training sets. That is, one of the ten partitions is withheld for testing and the remaining nine are used for training [45], and the process is repeated ten times. The error rate (MSE) is calculated and the time is recorded, during each training process. Finally, an overall error rate is determined by averaging the ten error rates, and the average time is calculated.

Using a single ten-fold cross-validation is not adequate to obtain a reliable error rate. Even though the experiment is executed ten times with the same learning algorithm on the same dataset, the results may differ due to variations in the random division of the folds. In order to obtain an accurate error rate, the standard procedure is to run the tenfold cross-validation itself ten times, which is known as the ten tenfold cross-validation, thereby invoking the learning algorithm one hundred times [45]. After execution of the ten tenfold cross-validation, the ten resulting error rates are averaged to yield a final error rate. .

5.2 Experimental Results

Recall that, during our experiments, we are addressing three different research questions. That is, we aim to explore the performance of neural networks on proteins from two families with different sizes, families with small sets of member proteins, and a data set containing proteins from diverse families. The datasets used for these experiments were introduced in the Section 4.4 and more related information, such as the query proteins, may be found in the Appendix A.

Recall that we do not use the family membership during training. Rather, this information is used as validation. That is, after the training and prediction phases, the predicted 3D shapes descriptors associated with the query proteins are used to search the CAPRI system in order to retrieve the closest proteins, in terms of 3D shape, from a database composed by 45,000 known protein structures. We employ the SCOP system in order to determine the families to which the retrieved protein structures belong. This information is then used to evaluate the performance of the neural network prediction.

5.2.1 Dataset 1: Two class problem with imbalance

In this experiment, we used a neural network that has only one hidden layer with 18 nodes. This number of hidden nodes was set by inspection. The neural network was trained using the gradient descent BP algorithm on the dataset composed by proteins from the P22 Tailspike and Phage Lysozyme families, with 7 and 79 members, respectively.

The training completed in 34 seconds, and the error rate trained by the network training process is 0.619. Once the network has been trained, it is able to predict the 3D shape descriptors of the query protein. The retrieved first 12 proteins are shown in Figures 13 and 14. Figure 13 shows the results retrieved by the predicted 3D shape descriptors of the query protein 1tyv, and Figure 14 shows the results retrieved by the

query protein 142l.

By verifying the retrieved proteins with the SCOP system, we find that the first retrieved protein 1qrc belongs to the same family with the query protein 1tyv, and the first retrieved protein 155l is in the same family with the query protein 142l. Therefore, the neural networks have properly predicted the 3D shape descriptors for 1tyv and 142l.

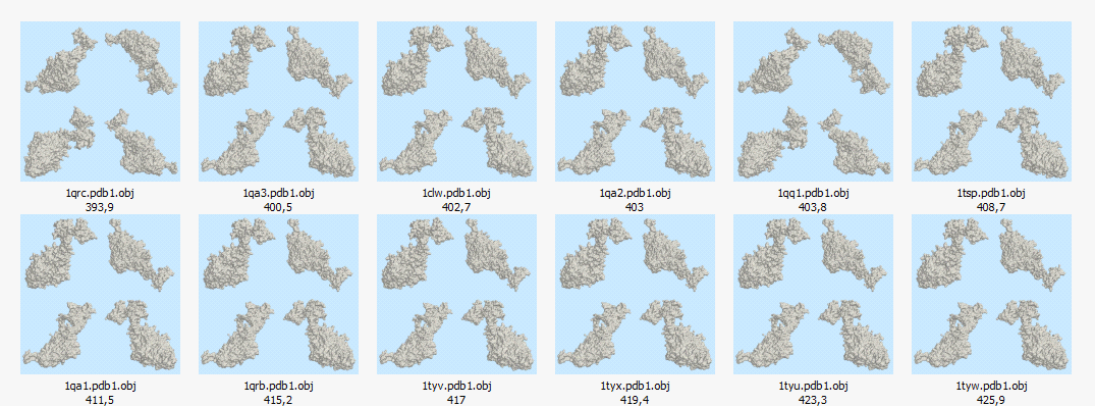


Figure 13. Results retrieved by the predicted 3D shape descriptors of 1tyv

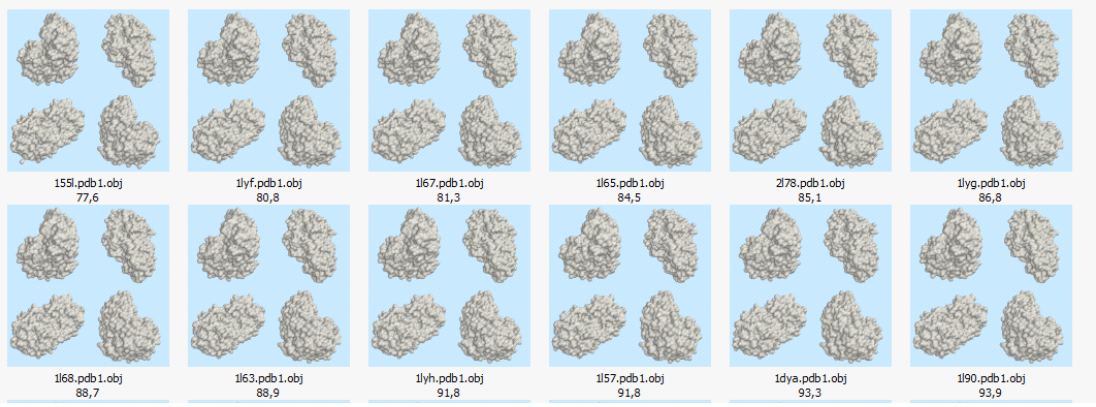


Figure 14. Results retrieved by the predicted 3D shape descriptors of 142l

Recall that, in order to explore the accuracy of predicted 3D shape descriptors, we use the SCOP system to verify whether there are retrieved proteins from the same family with the query protein. This is, indeed the case here and we can conclude that the results obtained from neural networks which are trained by the gradient descent BP algorithm are accurate for the CAPRI system to retrieve similar proteins. Thus, the

neural network performs well on the proteins from two classes with imbalanced sizes.

5.2.2 Dataset 2: Performance of neural networks on multiple classes with few examples

In some cases, especially for large families, several proteins may have similar 3D shapes. This fact provides more examples for neural network to learn the relationships, i.e. by using the similarities to refine the supervised search. However, there are a large number of proteins, who do not have a number of other proteins with similar 3D shapes. In this experiment, we address this issue. Our experiments were implemented on neural networks having three (3) different topologies, and the neural networks were trained with our previously introduced five (5) optimization algorithms. First, we explore the neural network that has only one hidden layer with 18 nodes. By observing the performance of neural networks, shown in Table 11, we find the error rates resulting from neural networks using different optimization algorithms are almost the same, except for the Levenberg-Marquadt (LM) algorithm, which performs better than the others. Though the time required by the Levenberg-Marquadt (LM) algorithm for network training is much shorter than with the BFGS Quasi-Newton algorithm, it still takes significantly longer time than the other three algorithms. In the other three, the gradient descent BP algorithm with momentum works fastest, and the gradient descent BP algorithm is the slowest.

Verifying the first twelve (12) retrieved proteins with SCOP System, we find that the proteins from the same family with the query protein can be retrieved by using the predicted descriptors, regardless of which optimization algorithm the neural networks use. However, there is an exception to this: when the network is trained using the Levenberg-Marquadt (LM) algorithm, the predicted descriptors of the query protein 1r1y from the family Globins fail to retrieve its family members. This failure is a false alarm. Here, a false alarm refers to the situation where the predicted values obtained from neural networks produced wrong results [46]. In our research, a false alarm

refers to the situation that the CAPRI system retrieves proteins from the wrong families by using predicted descriptors.

Although the error rate resulting from networks using the Levenberg-Marquadt (LM) algorithm is the lowest, the networks failed one of the 12 cases for verification. Thus, the gradient descent BP algorithm with momentum and the conjugate gradient algorithm with Fletcher-Reeves updates perform better, when networks have one hidden layer with 18 nodes and are trained on the proteins from multiple classes with few training examples.

Table 11. Performance of networks with one hidden layer and 18 nodes on Dataset 2

	Gradient Descent BP	Gradient Descent BP with momentum	Conjugate Gradient algorithm with Fletcher-Reeves Updates	BFGS Quasi-Newton algorithm	Levenberg-Marquadt (LM) algorithm
MSE	0.775	0.682	0.699	0.327	0.0117
Time(s)	29	26	58	1248	354

Next, the experiments are conducted on neural networks having one hidden layer with 40 nodes. When the neural networks are trained on the dataset composed by proteins from small classes, they cannot be optimized by the BFGS Quasi-Newton algorithm or the Levenberg-Marquadt (LM) algorithm. This is because the large volume of matrices operations involved in these two algorithms causes memory problem. When the network is optimized by the conjugate gradient algorithm with Fletcher-Reeves updates, the error rates produced during the training process is lower than the ones produced by networks which are trained by the other two algorithms, but it takes more time. Comparing with the Gradient descent BP algorithm with momentum and the conjugate gradient algorithm with Fletcher-Reeves updates, neural networks trained by the Gradient descent BP algorithm work fastest and produce the second lowest error rates. According to the results of experiments on these neural networks, for the predicted 3D shape descriptor of each query protein, there are retrieved proteins within

the first twelve (12) results belonging to the same family with the query protein. Thus, when the neural network having 40 nodes in its hidden layers is trained on dataset consisting of proteins with few training examples per class, the Gradient descent BP and Conjugate gradient algorithm with Fletcher-Reeves updates are proven to be the appropriate optimization algorithm.

Table 12. Performance of networks with one hidden layer and 40 nodes on Dataset composed by proteins from small families

	Gradient Descent BP	Gradient Descent BP with momentum	Conjugate Gradient algorithm with Fletcher-Reeves Updates	BFGS Quasi-Newton algorithm	Levenberg-Marquadt (LM) algorithm
MSE	0.546	0.594	0.0199	Fail to converge	Fail to converge
Time(s)	87	29	68		

When neural networks have two hidden layers, with 18 nodes in the first hidden layer and 19 nodes in the second, the error rates trained by different algorithms are almost the same, except for those trained by the Levenberg-Marquadt(LM) algorithm. Within the other four algorithms, the Conjugate gradient algorithm with Fletcher-Reeves update performs with the lowest error rates. As for the time consumed by the training process, the neural networks trained by the Gradient descent BP and the Gradient descent BP with momentum perform the fastest, and the Conjugate gradient descent algorithm with Fletcher-Reeves updates performs second fastest. Networks trained by the BFGS Quasi-Newton algorithm are the slowest, without much improvement their performance (error rate). According to the results of experiments on these networks, for the 3D shape descriptors of each query protein, there are proteins within the first twelve (12) retrieved results belonging to the same family with the query protein. Overall, combining the performance during training processes and verification results, the conjugate gradient algorithm with Fletcher-Reeves updates works better than the other algorithms, when the neural networks having two hidden layers, with 18 nodes in

the first hidden layer and 19 nodes in the second.

Table 13. Performance of networks with two hidden layers on Dataset composed by proteins from small families

	Gradient Descent BP	Gradient Descent BP with momentum	Conjugate Gradient algorithm with Fletcher-Reeves Updates	BFGS Quasi-Newton algorithm	Levenberg-Marquadt (LM) algorithm
MSE	0.566	0.654	0.424	0.334	0.0152
Time(s)	28	28	77	1827	496

Discussion

For the training process, the best predicting results, in terms of the mean square error and time consumed, are obtained by the Gradient descent BP with momentum when the network has one hidden layer and the Conjugate gradient descent algorithm with Fletcher-Reeves updates when network has two hidden layers. Due to a large volume of calculation, the BFGS Quasi-Newton BP and the Levenberg-Marquadt (LM) algorithm cause memory problem, when the network has one hidden layer with 40 nodes. Although networks using the Levenberg-Marquadt (LM) algorithm produce a better performance, in terms of the mean square of error, the time it takes for training is generally more than ten times longer than the time took by gradient methods.

5.2.3 Dataset 3: Multiple diverse classes and Dissimilar Amino Acid Sequences

Even when proteins share a very similar 3D shape, their amino sequence may present some variability. For proteins having very similar 3D shapes, but dissimilar amino acid sequences, it may difficult for neural networks to build the relationship between their amino acid sequence and 3D shapes. Thus, we use this dataset to explore the performance of neural networks in this case.

In order to explore generalization of our neural networks, we again employ tenfold cross-validation and perform it on the third dataset, which was introduced in Section

4.4.3. First, we implemented experiments on neural networks, which are trained by 5 different optimization algorithms. By analysing the performance of those neural networks, we find which optimization algorithm is more appropriate for this work, and implement the ten tenfold cross-validation on it. Since there are 240 protein 3D shapes contained in the dataset, 10% of them will be used as the testing set, which is 24 protein 3D shapes, according to the tenfold cross-validation. Those 24 protein 3D shapes are randomly selected by Matlab. In order to evaluate the accuracy of prediction, we verify the first eight (8) proteins retrieved through the CAPRI system by using the predicted 3D shape descriptors to determine whether there are retrieved proteins from the same family with the query protein. The above-mentioned numbers and parameters were set experimentally.

After running tenfold cross-validation on those diverse proteins with different neural network architectures and optimization algorithms, we obtained the average error rates and time produced by different optimization algorithms, shown in Table 14.

As seen by the results, the LM algorithm again has memory problem, due to the large volume of calculations. The same happens to the BFGS Quasi-Newton algorithm when it is operated on the neural network having one hidden layer with 40 nodes. When the BFGS Quasi-Newton algorithm is used for network training, the time consumed is much longer than the others, but without improvements in the error rates (MSE). As seen in Table 15, this algorithm does not have an advantage regarding the number of proteins retrieved by predicted descriptors. That is, the 3D shape descriptors predicted by networks using the BFGS Quasi-Newton algorithm are not as accurate as those predicted by the other three algorithms. Therefore, from this point on the comparison of neural network performance between different optimization algorithms will be based the other three algorithms, namely, the gradient descent BP algorithm, the gradient descent BP with momentum algorithm and the conjugate gradient algorithm with Fletcher-Reeves updates.

Table 14. Evaluation on Neural Networks using Different Optimization Algorithms

	Gradient Descent BP		Gradient Descent with momentum BP		Conjugate Gradient BP with Fletcher-Reeves Updates		BFGS Quasi-Newton BP		Levenberg-Marquadt (LM) BP	
	MSE	Time (s)	MSE	Time (s)	MSE	Time (s)	MSE	Time (s)	MSE	Time
18	0.140	51	0.094	134	0.156	5	0.239	1.26e+04	Fail to converge	
40	0.194	46	1.113	46	0.114	11	Fail to converge			
18&19	0.168	31	0.303	120	0.153	6	0.235	1.79e+04		
Average	0.167	42	0.503	100	0.141	7				

Among these three optimization algorithms, the gradient descent BP with momentum has the highest error rate, and it requires much more time to train the neural network than the other two algorithms. As shown by the poor performance in error rates, the number of proteins with predicted 3D shape descriptors that may retrieve their family members is the lowest.

Table 15. Percentage of Proteins whose Family Members may be found by their Predicted Descriptors

	Gradient Descent BP	Gradient Descent with momentum BP	Conjugate Gradient BP with Fletcher-Reeves Updates	BFGS Quasi-Newton BP
18	75%	38%	75%	0%
40	71%	54%	75%	Fail to converge
18&19	50%	46%	71%	42%
Average	65%	46%	74%	

The error rates from networks using the gradient descent BP algorithm and the conjugate gradient algorithm with Fletcher- Reeves updates are almost the same; those obtained using the conjugate gradient algorithm with Fletcher-Reeves updates are slightly lower. Thus, neural networks trained by the conjugate gradient algorithm with Fletcher- Reeves updates perform better than those trained by the gradient descent BP algorithm. According to the results from verification, the prediction is more accurate, when the conjugate gradient algorithm with Fletcher-Reeves updates is used. It not only performs slightly better than the gradient BP algorithm, it also shows a significant

advantage in the time consumed. That is, it may train networks far more quickly than the gradient BP algorithm.

Table 16. Ten tenfold cross-validation on neural networks trained by conjugate gradient descent algorithm with Fletcher-Reeves updates

	One hidden layer with 18 nodes	One hidden layer with 40 nodes	Two hidden layers with 18 and 19 nodes
Average MSE	0.049	0.021	0.041
Average Time(s)	584	641	596
Average Percentage of proteins whose family members are found	63%	83%	58%

Analysis of the performance of networks using different BP algorithms shows that the conjugate gradient BP with Fletcher-Reeves updates algorithm works best. Subsequently, we apply the ten tenfold cross-validation to neural networks optimized by this algorithm on dataset composed by diverse proteins.

According to both the average error rates and percentage of proteins whose family members are found during verification demonstrated in Table 16, neural networks with 40 nodes perform better than the other architecture. In terms of time consumed during the training process, networks with 40 nodes in one hidden layer work faster than those with 18 nodes, but slower than networks with two hidden layers, and 18 and 19 nodes respectively.

Comparing the results of the network with 18 nodes in its hidden layer, and the network with two hidden layers, the time required for training is reduced with a higher number of hidden layers, but performance does not improve. Although the network with 40 nodes in one hidden layer takes a little more time than the one with two hidden layers, its performance is much better. Thus, in this research the neural network with one hidden layer performs better than the network with two hidden layers, and networks with 40 nodes in their hidden layers perform better than those with 18 nodes. Although false alarms appear under all three neural network architectures, neural

networks having one hidden layer with 40 nodes has less false alarms. Since the single layer neural networks with 40 nodes are involved with more weights, it helps to reduce the time used by the optimization and improve the results obtained from the optimization.

As stated earlier, this diverse protein dataset contains imbalanced data, in that the sizes of the families therein vary. In order to assess the influence caused by the data itself, we compare the percentage of successfully retrieved proteins for each protein family, under specific network architecture. The percentage is equal to the number of query proteins whose family members are existed in the results retrieved by their predicted descriptors, divided by the total number of proteins having similar 3D shapes with them in the training set. This percentage is shown in Appendix A, Tables 1 and 2.

According to the results in the tables, it is easier for neural networks to learn the relationship between amino acid sequence descriptors (inputs) and 3D shape descriptors (outputs) when the proteins have more training examples. For example, for the query protein from family Phage lysozyme, there are 63 training examples and all of them have very similar amino acid sequences and 3D shapes. Regardless of which architecture the neural network uses, the predicted descriptors succeeded in retrieving the known proteins in which there are proteins belonging to the same family with the query protein. For proteins having similar 3D shapes, but dissimilar amino acid sequences, such as the query proteins from the family Globins and P22 tailspike protein, the 3D shape descriptors may be predicted accurately due to the large number of learning examples for the networks. The prediction accuracy is improved by expanding the architectures of networks. However, a protein without any proteins having similar 3D shape and similar amino acid sequences with it, such as the protein from family DNA polymerase processivity factor (Bacteriophage T4), generates inferior verification results, as may be expected.

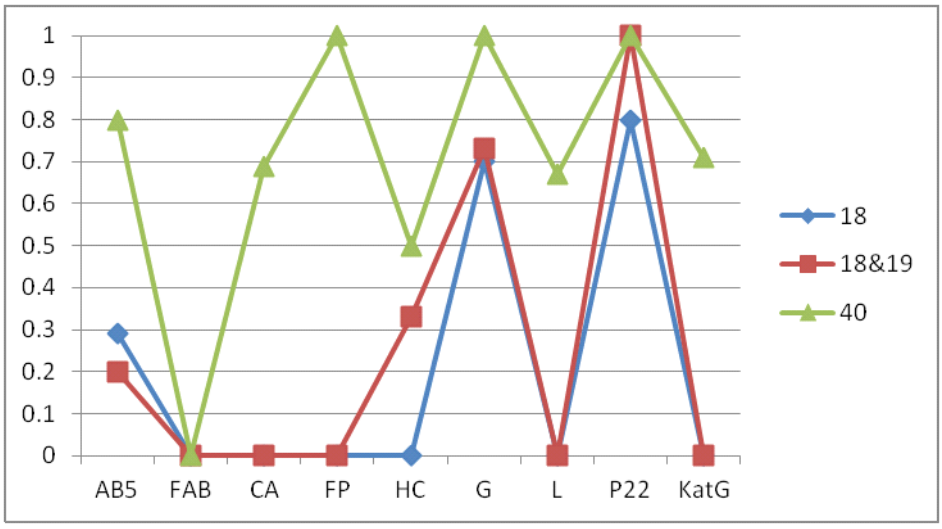


Figure 15. Line Chart: Percentage of Proteins Whose Family Members are Retrieved (Proteins having similar 3D shapes and dissimilar amino acid sequence)

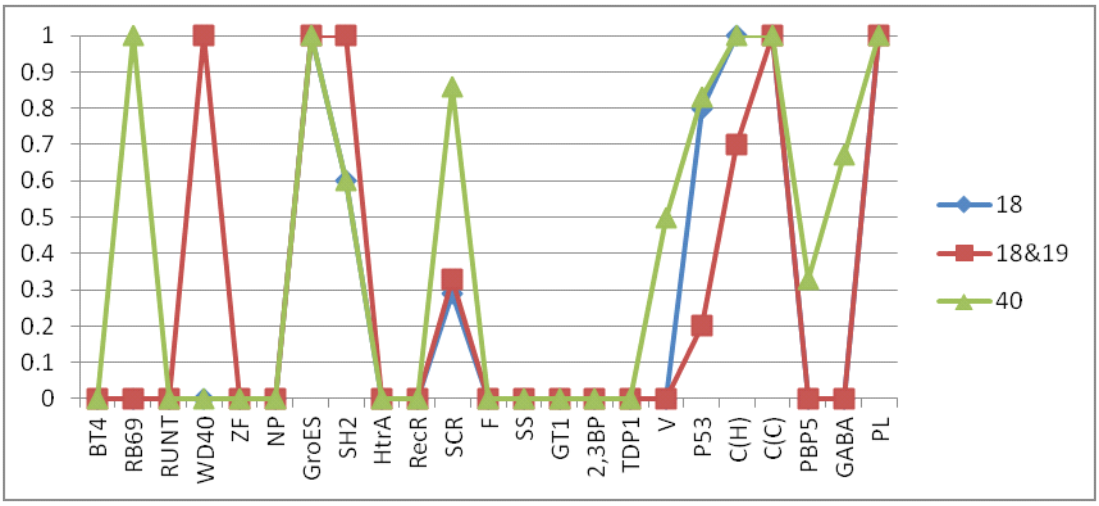


Figure 16. Line Chart: Percentage of Proteins Whose Family Members are Retrieved (Proteins having similar 3D shapes and similar amino acid sequence)

In order to indicate the network performance in line chart, we use the family code to represent that family, instead of the family name. The corresponding family code with each family can be found in the Appendix A Table 1 and 2. After depicting the percentage of proteins whose family members are retrieved in a line chart (Figure 15 and 16), we may see that the neural network with 40 nodes in its hidden layer works best, and that the network with two hidden layers works better than the one with one

hidden layer and 18 nodes.

This allows us to determine whether the accuracy of the predicted descriptors may be improved by increasing the number of nodes in the hidden layers to 55 and 70 respectively. When we increase the nodes in the hidden layer, the verification performance on proteins from families Catalse-peroxidase KatG (1z6f) and GABA-amino transferase-like (2bi3) is dramatically improved, but performance on the other families is not. When there are 70 nodes in the hidden layer, the accuracy of descriptors obtained by neural networks for the family Phage lysozyme (142l) is stable, which is 100%.

Discussion

The aim of this experiment was to explore the performance of our neural networks on proteins from families with varying sizes and properties. Next, in order to evaluate the neural network performance we analyse the results from the perspective of the data properties. We will analyse the neural networks performance on proteins having similar 3D shapes and similar amino acid sequences, and then continue the discussion with proteins having similar 3D shapes, but not so similar amino acid sequences.

In Figure 16 and also Appendix B Table 1, proteins have both similar 3D shapes and similar amino acid sequences. When there is only one training example, neural network training and predicting accurate descriptors is difficult. Even when we increase the number of hidden layers, or the number of nodes in the hidden layer, there is no improvement in the accuracy of predicted descriptors. When the family has a large number of examples, such as query proteins from family Phage Lysozyme, it is easy for the neural networks to learn the relationships between the amino acid sequence descriptors and the 3D shape descriptors, and the trained neural networks may predict the 3D shape descriptors for the query proteins with 100% accuracy. When there are fewer examples, such as query proteins in the Ferritin family, it is difficult for

the networks to learn the relationships. Our experiments showed that, in general, the accuracy of the prediction descriptors is improved, or the false alarm is reduced, by increasing the number of nodes in the hidden layer of the networks when networks have one hidden layer. Further, networks with one hidden layer outperform those with two hidden layers. Differences in the network training data influence the performance of network training. When there are more data examples from which the networks may learn the relationships between inputs and outputs, the predicted descriptors are more accurate, and the prediction produces less false alarms.

Using the predicted 3D shape descriptors to locate the closest proteins depends on whether there are enough examples for the required network training, and whether the proteins having very similar 3D shapes present a similarity in their amino acid sequences. This has been shown in situations with families of proteins having similar 3D shapes, but dissimilar amino acid sequences, as seen in Figure 15 and Appendix B Table 2. When there are more family members, the prediction of 3D shape descriptors is more accurate, and fewer false alarms are produced. For example, the Bacterial AB5 toxin, B-subunits families have 3 training examples, while the P22 tailspike protein has 12 training examples. The predicted descriptors for proteins from the latter are more accurate than those from the former. In addition to the number of examples that may be used for network training, prediction performance also depends on the diversity of proteins with similar 3D shapes. Even though the Ligand-gated protein channel and Catalase-peroxidase KatG have same number of training examples, the predicted descriptors for proteins from Ligand-gated protein channel have higher accuracy, because proteins in the Catalase-peroxidase KatG are more diverse, in terms of amino acid sequences. In the Ligand-gated protein channel, two proteins share similar amino acid sequences and the other five have similar sequences, while in the Catalase-peroxidase KatG, three of the seven proteins have similar sequences and the other four have similar sequences. When the networks perform on proteins having similar 3D shapes and dissimilar amino acid sequences, networks having one hidden layer

generally outperform those with two hidden layers, and the performance may be further improved by increasing the number of nodes in the hidden layer.

Overall, training is difficult when there are only a few members of a family that the neural networks may learn from. Although increasing the nodes in the hidden layers of the neural network may improve network training, the more imbalanced the dataset is, the less significant the improvement will be. When the query protein has a large number of training examples, and those training examples share similar 3D shapes and similar amino acid sequences, the predicted descriptors are highly accurate to find its family members in the retrieved known proteins. When the query protein having a small number of training examples, or its other family members have similar 3D shapes but dissimilar amino acid sequences, the accuracy for prediction is from 50% to 80%, and the performance improves by increasing the number of nodes in the hidden layer.

5.3 Summary of Results

According to the results, the accuracy of the predicted 3D shape descriptors of the query protein is influenced by two factors, namely, the design of the neural network and the nature of the data, as may be expected.

In terms of neural network design, we explored the performance of neural networks using different optimization algorithms and architectures. According to the error rates and time we obtained during network training, the conjugate gradient algorithm with Fletcher-Reeves updates is the best learning algorithm for the prediction of 3D shape descriptors. We have shown that the conjugate gradient algorithm with Fletcher-Reeves updates is the method with the fastest convergence. This is most likely because it optimizes the objective function along the mutually orthonormal direction. Although the Levenberg-Marquadt (LM) algorithm shows a better error rate than the conjugate gradient algorithm with Fletcher-Reeves updates, it is still not an effective approach for networks with fewer weights. This is because it takes a longer time, due to the

calculation of the Hessian matrix. After the tenfold cross-validation, we found that networks with one hidden layer containing more neurons perform better than networks with two hidden layers.

When proteins have similar 3D shapes, neural networks may predict the 3D shape descriptors of the query protein accurately. Networks with one hidden layer with more than 40 nodes perform better than networks with two layers. However, the performance depends on the number of examples available for the networks to learn the relationships from, and the diversity of the amino acid sequences. Networks are unable to predict accurate 3D shape descriptors for query proteins with less training examples, such as the proteins in Bacterial AB5 toxin, B-subunits. It is also difficult for neural networks to establish the required relationships to predict accurate descriptors if the proteins as contained in the training examples have diverse amino acid sequences. Predicted descriptor accuracy may be improved by increasing the number of neurons in the hidden layer when networks have only one hidden layer. When proteins have similar 3D shapes, the neural networks may predict 3D shape descriptors with 100% accuracy if the proteins share similar amino acid sequences. It is difficult for neural networks to predict accurate descriptors for members having similar 3D shapes but dissimilar amino acid sequences; the accuracy is from 50% to 80%. Conclusion

In this chapter, we presented an evaluation of the performance of the neural networks used to learn the 3D shape descriptors. We also compared the performance of networks on query proteins from different family sizes and with a variety of properties. Our results suggest that out of the five algorithms we explored, the conjugate gradient BP with Fletcher-Reeves updates was found to be the best algorithm for the prediction of 3D shape, in terms of the accuracy of the predicted descriptors. The neural networks perform well on the 3D shape descriptors prediction, when the query protein has a large amount of family members and they share similar amino acid sequences. It is difficult for networks to learn the relationship between amino acid sequence descriptors and 3D shape descriptors when the protein has a small number of training

examples or the training examples have dissimilar amino acid sequences. However, the performances may be improved and the false alarms may be reduced by increasing the number of nodes in the hidden layer, when there is only one hidden layer.

Chapter 6

Conclusion

Knowledge about the 3D shapes of proteins is important, because it improves biologists' and pharmacologists' understanding of protein functions and interactions. Accurate prediction of 3D shapes may enhance protein function analysis and drug design. Proteins are composed of amino acid sequences folding together, and there is a

link between the amino acid sequences and the 3D shapes. Although there are a large number of proteins whose amino acid sequences are known, the number with known 3D shapes is relatively small. This is due to the fact that the 3D shapes are more complex than the sequences. There is a need to reduce the discrepancy in the number of proteins with known sequences and those with known 3D shapes. That is, we need to find a more efficient approach to predict the 3D shapes.

In this research, we propose a novel approach to predict protein 3D shapes from the amino acid sequences using neural networks. We create an algorithm to compute a descriptor for the amino acid sequence and then use our descriptors to train a neural network to predict proteins' 3D shapes.

6.1 Thesis Contributions

Methods to predict the structures of proteins directly from their amino acid sequences have been proposed previously in the research community, but all are limited to predicting the secondary structure. Our main contribution makes it possible to use the amino acid sequences to predict 3D shapes by using a feed-forward neural network. Our analyses show that this new approach produces promising results.

In order for neural networks to analyse and learn the relationships between amino acid sequences and 3D shapes, we need to transform them into a form that is independent from their underlying sequences or 3D shapes. We designed an algorithm to create amino acid sequence descriptors, based on the probability density functions associated with the correlation between pairs of amino acids, and used them as inputs to the neural network. We used the 3D shape descriptors, which are provided by the CAPRI system, as the outputs. The neural network is trained by a group of proteins with similar amino acid sequences with the query protein. Once the network is trained, it may predict the 3D shape descriptors of the query protein from its amino acid sequence descriptors. The CAPRI system then utilizes the predicted descriptors to locate the known proteins with the closest structures.

In this research, we explored five different optimization algorithms, namely, the gradient descent BP, the gradient descent BP w/momentum, the conjugate gradient descent algorithm with Fletcher-Reeves updates, BFGS Quasi-Newton algorithm and Levenberg-Marquadt (LM) algorithm. The latter two algorithms perform poorly, which means the target function is difficult to converge. Both these two algorithms are based on an approximation of the Hessian function. The calculation of the Hessian during each one of the iterations takes a significant amount of time, particularly for neural networks with a large number of inputs. The conjugate gradient descent BP with Fletcher-Reeves updates converges far faster, likely because the optimization is explored with mutually orthogonal directions.

The influences that different network topologies and different protein data have on prediction accuracy are also explored. Neural networks with one hidden layer and more nodes perform better than networks with two hidden layers. When a neural network is trained with proteins from families with similar amino acid sequences and similar 3D shapes, the predicted 3D shape descriptors are generally 100% accurate and allows for the CAPRI system to retrieve the family members of the query protein. However, when a network is trained by proteins from families with dissimilar amino acid sequences the accuracy of prediction ranges from 50 to 100%. The accuracy may be improved by increasing the number of hidden nodes, or the number of proteins used for network learning.

6.2 Future Work

To this point, our research on protein 3D shape prediction does not allow for the direct reconstruction of the shapes based on descriptors predicted by neural networks. Therefore, this will be our future research direction. We intend to use spherical harmonics to index the protein 3D shapes to provide information, and pose invariance to enable the direct reconstruction.

Spherical harmonics, functions which arise in physics and mathematics [47], are the

angular portion of a set of solutions to Laplace's equation [47]. Spherical polar coordinates, with coordinates θ and ϕ locating a point in space, are used to investigate problems in three dimensions. Spherical harmonics plays an important role in the descriptions of 3D shapes in computer graphics [44]. The useful property of spherical harmonics to describe the rotations, spherical averaging procedures and smooth surface representation on a sphere, makes it widely used in protein crystallography, molecular docking, protein surface display routines and other applications.

The spherical harmonics, $Y_{lm}(\theta,\phi)$, are single-valued and infinitely differentiable complex functions of two variables θ and ϕ , which are indexed by two integers, l and m . In quantum physics, l is the angular quantum number, which represents the number of local minima of the function and a spatial frequency, and m is the azimuthal quantum number [47].

Due to the complete set of orthonormal functions formed by spherical harmonics, the vector space formed is analogue to unit basis vectors. Thus, in the same way that a vector is described by its projections on to each axis, the shape may then be described by coefficients of a real spherical harmonics expansion [47]. Any function of θ and ϕ may be expanded as follows:

$$f(\theta,\phi)=\sum_{l=0}^{\infty}\sum_{m=-l}^{l}c_{lm}Y_{lm}(\theta,\phi) \tag{10}$$

The expansion coefficients, c_{lm} , may be calculated by multiplying the function with the complex spherical harmonics and then integrating with the solid angle Ω , and it is shown below (inner product):

$$c_{lm}=\int_{S_2} f(\theta,\phi)Y_{lm}^{*}(\theta,\phi)\,d\Omega \tag{11}$$

As these coefficients are unique, they may be used directly to describe the shape. Through this indexing method, we may construct the shape of a protein directly from its 3D shape descriptors.

In addition to construct the 3D shapes of proteins directly from their predicted descriptors, we may also expand our experiments on other types of artificial neural networks, such as the Bayesian neural networks, or even explore the predicting performance on an ensemble networks. Besides the BLOSUM matrix we used for amino acid sequence descriptors calculation, there are many other metric for two amino acids comparison from a physicochemical point of view [48]. The experiments may be performed on the amino acid sequence descriptors calculated through other measures, such as the PAM (Point Accepted Mutation) matrix, which is a substitution matrix that describes the expected evolutionary change at amino acid level. Within our experiments, some predicted 3D shape descriptors make the CAPRI system find the wrong family, which may be seen as false alarms. We may explore those false alarms in our future work as well.

The prediction of proteins' 3D shapes is based on the learned relationship between known amino acid sequences and known 3D shapes. Recall that there is a gap between the number of known amino acid sequence and the number of known 3D shapes. If the query protein's amino acid sequence and 3D shape present a brand new relationship, it is difficult for neural networks to predict an accurate 3D shape, since neural networks have not been trained to learn this relationship. Thus, the limitation of this work is that it is difficult for neural networks to predict accurate 3D shapes for the query protein, when the relationship between amino acid sequences and 3D shapes of the query protein has not been presented within the known proteins and learned by neural networks.

Bibliography

[1] A.Polanski & M. Kimmel, Bioinformatics, New York: Springer, 2010.

[2] R.J.Morris, R.J.Najmanovich, A.Kahraman & J.M. Thornton, "Real Spherical Harmonic Expansion Coefficients as 3D Shape Descriptor for Protein Binding Pocket and Ligand Comparisons," *Bioinformatics,* vol. 21, no. 10, pp. 2347-2355, 2010.

[3] T. Lengauer & R. Zimmer, "Protein Structure Prediction Methods for Drug Design," *Briefings in Bioinformatics,* vol. 1, no. 3, pp. 275-288, 2000.

[4] T. Schwvede & M.C. Peitsch, Computational Structural Biology: Methods and Applications, London: World Scientific, 2008.

[5] Y. Zhang, "Protein Structure Prediction: when is it useful," *Current Opinion on Structural Biology,* vol. 19, no. 2, pp. 146-155, 2009.

[6] G. S. M. John, C. Rose & S. Takeuchi, "Understanding Tools and Technique in Protein Structure Prediction," in *System and Computational Biology-Bioinformatics and Computational Modeling*, InTech, 2011, pp. 185-212.

[7] H. Rangwala & G. Karypis, "Introducation to Protein Structure," in *Introduction to Protein Structure Prediction*, John Wiley & Sons, Inc., 2010, pp. 1-15.

[8] S.H. Park & K.H. Ryu, "Effective Filtering for Structural Similarity Search in Protein 3D Structural Database," in *Database and Expert Systems Applications: Lecture Notes in Computer Science*, New York, Springer, 2004, pp. 761-770.

[9] Y.S. Chiang, T.I. Gelfand. A.E. Kister, I.M. Gelfand, "New Classification of Supersecondary Structures of Sandwich-like Protein Uncovers Strict Patterns of Strand Assemblage," in *Proteins*, Wiley-Liss Inc., 2007, pp. 915-921.

[10] Ryan Andrew, "The Core Matrix," 5 Apr 2011. [Online].
Available: http://thecorematrix.wordpress.com/2011/04/05/all-about-protein/.
[Accessed 1 Nov 2012].

[11] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N.

Shindyalov & P. E. Bourne, “The Protein Data Bank,” in *Nucleic Acids Res*, Oxford University Press, 2000, pp. 235-242.

[12] “RSCB Protein Data Bank,” [Online]. Available: http://www.rcsb.org/pdb/home/home.do.[Accessed 11 September 2012].

[13] A. Tramontano, Protein Strucure Prediction Concepts and Applications, Wiley-VCH, 2006.

[14] L. Wei, R.B. Altman & J.T. Chang, “Using the Radial Distributions of Physical Features to Compare Amino Acid Environments and Align Amino Acid Sequences,” *Pac. Symp. Biocomput,* vol. 2, pp. 465-476, 1997.

[15] A. G. Murzin, S. E. Brenner, T. J. P. Hubbard & C. Chothia, “Structural Classification of Proteins,” MRC Laboratory of Molecular Biology and Centre for Protein Engineering, June 2009. [Online]. Available: http://scop.mrc-lmb.cam.ac.uk/scop/intro.html. [Accessed 11 April 2012].

[16] A. Andreeva, D. Howorth, J. Chandonia, S.E. Brenner, T.J. P. Hubbard, C. Chothia & A.G.Murzin, “Data growth and its impact on the SCOP database: new developments,” *nudeic Acids Rec,* vol. 36, pp. 419-425, 2008.

[17] G. Rennie, “The Art of Protein Structure Prediction,” U.S Department of Energy Research News, 12 2004. [Online]. Available: http://www.eurekalert.org/features/doe/2004-12/ddoe-tao122204.php. [Accessed 17 July 2012].

[18]] M. A. Marti-Renom, A. C. Stuart, A. Fiser, R. Sanchez, F. Melo & A. Sali, “Comparative Protein Structure Modeling of Genes and Genomes,” *Annual Review of Biophysics and Biomolecular Strucure,* vol. 29, pp. 291-325, 2000.

[19] M. Kantardzic, Data Mining: Concepts, Models, Methods and Algorithms, NJ: Wiley-IEEE Press, 2011.

[20] D. Hand, H. Mannila & P. Smyth, Principles of Data Mining, MIT Press, 2001.

[21] J. Han & M. Kamber, Data Mining: concepts and techniques, San Francisco: Morgan Kaufmann Publisher, 2006.

[22] D. Taniar, Data Mining and Knowledge Technologies, IGI Pub, 2008.

[23] M. H. Dunham, Data Mining: Introductory and Advanced Topics, Prentice/ Pearson Education, 2003.

[24] G. H. Nguyen, A. Bouzerdoum & S. L. Phung, “A Surpervised Learning Approach

for Imbalanced Data Sets," in *ICPR 2008 19th International Conference on Pattern Recognition*, 2008.

[25] Y. L. Murphey, H. Guo & L. Feldkamp, "Neural Learning form Imbalanced Data," *Applied Intelligence, Special issues on neural networks and applications,* vol. 21, pp. 117-128, 2004.

[26] K. Yoon & S. Kwek, "A Data Reduction Approach for Resolving the Imbalanced Data Issue in Functional Genomics," *Neural Comput. and Applic.,* vol. 16, pp. 295-306, 2007.

[27] V. L. Berardi & G. P. Zhang, "The Effect of Misclassification Costs on Neural Network Classifiers," *Desicion Sciences,* vol. 30, no. 3, pp. 659-682, 1999.

[28] O. Mainmn & L. Rokach, Data Mining and Knowledge Discovery Handbook, New York: Springer, 2010.

[29] J. Nocedal & S. J. Wright, Numerical Optimization, New York: Springer, 2000.

[30] H. Cheng, T. Z. Sen, A. Kloczkowski, D. Margaritis & R. L. Jernigan, "Prediction of Protein Secondary Structure by Mining Structural Fragment Database," *Polymer,* vol. 46, no. 12, pp. 4314-4321, 2005.

[31] S. B. Deris, R. B. M. Illias, S. B. Senafi, S. O. Abdalla & S. N. V. Arjunan, "Univerisit Teknologi Malaysia," 2007. [Online].
Available: http://eprints.utm.my/4265/1/74017.pdf. [Accessed 2 September 2012].

[32] H. Bordoloi & K. K. Sarma, "Protein Structure Prediction Using Multiple Artificial Neural Network Classifier," *Studies in Computational Intelligence,* vol. 395, pp. 137-146, 2012.

[33] H. K. Ho, L. Zhang, K. Ramamohanarao & S. Martin, "A survey of machine learning methods for secondary and supersecondary protein structure prediction," *Methods in Molecular Biology,* vol. 932, pp. 87-106, 2013.

[34] W. Pirovano & J. Heringa, "Protein Secondary Strucutre Prediction," in *Data Mining Technique for The Life Science* , Springer, 2010, pp. 327-348.

[35] P. D. Lena, P. Fariselli, L. Margara, Marco Vassura & R. Casadio, "Divide and Conquer Strategies for Protein Structure Prediction," *Mathematical Approaches to Polymer Sequence Analysis and Related Problems,* pp. 23-46, 2011.

[36] European Bioinformatics Institute, 2012. [Online].

Available: http://www.ebi.ac.uk/Tools/sss/psiblast/.

[37] L. J. McGuffin, K. Bryson & D. T. Jones, "The PSIPRED protein structure prediction server," *Bioinformatics,* vol. 16, no. 4, pp. 404-405, 2000.

[38] J. Chen & N. S. Chaudhari, "Statistical Analysis of Long-Range Interactions in Proteins," in *BIOCOMP*, Las Vegas, 2006.

[39] H. Zeng, L. Zhou, L. Li & Y.Wu, "An improved prediction of protein secondary strucutures based on a multi-mold integrated neural network," in *Natrual Computation (ICNC)*, Chongqing, 2012.

[40] T.J. Koswatta, P. Samaraweera & V. A. Sumanasinghe, "A simple comparison between specific protein secondary structure prediction tools," *Tropical Agrichultural Research,* vol. 23, pp. 91-98, 2012.

[41] H. Zhang, T. Zhang, K. Chen, K. D. Kedaisetti, M. J. Mizianty, Q. Bao, W. Stach & L. Kurgan, "Critical assessment of high-throughput standalon methods for secondary strucutre prediction," *Briefings in Bioinformatics,* vol. 12, no. 6, pp. 672-688, 2011.

[42] E. Paquet & H.L. Viktor, "Finding Protein Family Similarities in Real Time Through Multiple 3D and 2D Representations, Indexing and Exhaustive Searching," in *ISTICC, ACM SIGMIS International Conference on Knowledge Discovery and Information Retrieval-KDIR*, Madeira, Portugal, 2009.

[43] E. Paquet & H. L. Viktor, "CAPRI-Content-based Analysis of Protein Structure for Retrieval and Indexing," in *The 33rd Very Large Databases (VLDB2007) Conference, Workshop on Data Mining in Bioinformatics*, 2007.

[44] T. Jayalakshmi & A. Santhakumaran, "Statistical Normalization and Backpropogation for Classification," *International Journal of Computer Theory and Engineering,* vol. 3, no. 1, pp. 1793-8201, 2011.

[45] I. H. Witten & E. Frank, Data Mining: practical machine learning tools and technique with Java, San Francisco: Morgan Kaufmann Publisher, 2000.

[46] N. Thirathon, "Extension and Alarm Analysis of Neural Network Prediction of Separation Distance in AMASS Safety Logic," 23 May 2003. [Online]. Available: http://web.mit.edu/profit/htdocs/thesis/NattavudeThirathon.pdf. [Accessed 15 12 2012].

[47] V. Venkatraman, L. Sael & D. Kihara, "Protential for Protein Surface Shape

Analysis Using Spherical Harmonics and 3D Zernike Descriptors," *Cell Biochemistry and Biophysics,* vol. 54, pp. 23-32, 2009.

[48] M. S. Venkatarajam & W. Braun, "New Quantitative Descritors of Amino Acids based on Multidimensional Scaling of A Large Number of Physical-chemical Properties," *J Mol Model,* vol. 7, pp. 445-453, 2001.

[49] A.Shepherd, "Protein Secondary Structure Prediction with Neural Network A Tutorial," [Online].
Available: http://www.biochem.ucl.ac.uk/~shepherd/sspred_tutorial/ss-pred-old.html. [Accessed 1 May 2012].

Appendix A. Experimental Results

Family Name	Family Code	Size	One hidden layer with 18 nodes	Two hidden layers with 18 and 19 nodes	One hidden layer with 40 nodes	One hidden layer with 55 nodes	One hidden layer with 70 nodes
DNA polymerase processivity factor(Bacteriophage T4)	BT4	1	NAN	0%	0%	0%	0%
DNA polymerase processivity factor(Bacteriophage RB69)	RB69	2	NAN	0%	100%	100%	100%
RUNT domain	RUNT	1	0%	0%	0%	0%	50%
WD40-repeat	WD40	1	0%	100%	0%	NAN	0%
Zinc finger based beta-beta-alpha motif	ZF	1	0%	NAN	NAN		NAN
Nucleoside phosphorylase/phoshoribosyltransferase N-terminal domain	NP	1	0%	0%	NAN	0%	0%
GroES	GroES	2	100%	100%	100%	NAN	100%
SH2 domain	SH2	1	60%	100%	60%	100%	100%
HtrA-like serine proteases	HtrA	1	0%	0%	0%	0%	0%
Recombination protein RecR	RecR	1	NAN	0%	0%	0%	NAN
Complement control module/SCR domain	SCR	3	29%	33%	86%	0%	
Ferritin	F	2	NAN	0%	0%		
Sermidine synthase	SS	1	NAN	NAN	0%		
Glycosyl transferases group 1	GT1	1	0%	0%	NAN		
2,3-Bisphosphoglycerate-independent phosphoglycerate mutase, substrate-binding domain	2,3BP	1	NAN	0%	0%		
Tyrosyl-DNA phosphodiesterase TDP1	TDP1	1	0%	NAN	0%		NAN
V set domains(antibody variable domain-like)	V	3	0%	0%	50%		
P53 tetramerization domain	P53	8	80%	20%	83%		
Calmodulin-like (Human)	C(H)	1	100%	70%	100%		
Calmodulin-like (Cow)	C(C)	5	100%	100%	100%		
PBP5 C-terminal domain-like	PBP5	5	0%	0%	33%		100%
GABA-aminotransferase-like	GABA	8	0%	0%	67%		100%
Phage Lysozyme	PL	64	100%	100%	100%		100%

Table 1 Percentage of proteins whose family members are retrieved

(Proteins in the same family have similar amino acid sequences)

Note: 'NAN' in the tables means that there is no protein in the file randomly selected into the testing set..

Family Name	Family Code	Size	One hidden layer with 18 nodes	Two hidden layers with 18 and 19 nodes	One hidden layer with 40 nodes	One hidden layer with 55 nodes	One hidden layer with 70 nodes
Bacterial AB5 toxin, B-subunits	AB5	3	29%	20%	80%		
Fatty acid binding protein-like	FAB	2	0%	0%	0%		
Carbonic anhydrate	CA	12	0%	0%	69%		
Fluorescent proteins	FP	4	0%	0%	100%		
Heme-dependent catalases	HC	4	0%	33%	50%		
Globins	G	34	70%	73%	100%	93%	82%
Ligand-gated protein channel	L	7	0%	0%	67%		
P22 tailpike protein	P22	12	80%	100%	100%		
Catalase-peroxidase KatG	KatG	7	0%	0%	71%	70%	50%

Table 2 Percentage of proteins whose family members are retrieved
(Proteins in the same family have similar amino acid sequences)

Note: 'NAN' in the tables means that there is no protein in the file randomly selected into the testing set.