

PROTEIN SECONDARY STRUCTURE PREDICTION USING MULTI-INPUT  
CONVOLUTION NEURAL NETWORK

BY

SHAYAN IHSAN JALAL

A THESIS

Submitted in partial fulfillment of the requirements

for the degree of Master of Science

in Computer Science

in the Graduate School of

Troy University


Troy, Alabama, United States

May, 2018


PROTEIN SECONDARY STRUCTURE PREDICTION USING MULTI-INPUT  
CONVOLUTION NEURAL NETWORK

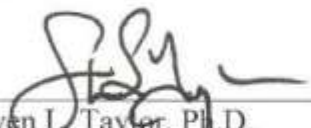
Submitted by Shayan Ihsan Jalal in partial fulfillment of the requirements  
for the degree of Masters of Science  
in computer science  
in the Graduate School of  
Troy University

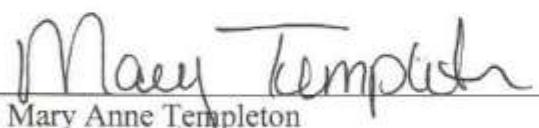
Accepted on behalf of the Faculty of the Graduate School by the thesis committee:

  
\_\_\_\_\_  
Bill Zhong, Ph.D.  
Chair Department of Computer Science  
Troy Campus

  
\_\_\_\_\_  
Alberto Arteta, Ph.D.

  
\_\_\_\_\_  
Long Ma, Ph.D.

  
\_\_\_\_\_  
Steven I. Taylor, Ph.D.  
Dean, College of Arts and Sciences

  
\_\_\_\_\_  
Dr. Mary Anne Templeton  
Associate Provost and  
Dean of the Graduate School

  
\_\_\_\_\_  
Date

## **ABSTRACT**

### **PROTEIN SECONDARY STRUCTURE PREDICTION USING MULTI-INPUT CONVOLUTION NEURAL NETWORK**

Protein secondary structure prediction is a well-known problem in bioinformatics. What makes this problem so hard to solve is there are no clear boundaries between protein features like helix and coil nor sheet and coil states (Yang, Gao, Wang, Heffernan, Hanson, Paliwal, & Zhou, 2016). Moreover, it costs a lot of money and time to do an experiment on each protein in the lab; so predicting protein structures computationally is the only practical solution. The artificial neural network has a good reputation for solving classification problems. Convolutional neural networks are an efficient method to learning sophisticated features; anyway, their performance does not increase monotonically mutually their complexity. In our project, we try several methods to get best results to predict protein secondary structure. First, we pad the sequence and add layers to convolution neural network with only input the primary structure of the protein. Second, we categorize the amino acid and the target structure without padding the sequence and it gives us accuracy 90.93% for SSP and 80.1% for DSSP, which is better by 1% from the best accuracy achieved till now. Moreover, using multi-input convolution neural network improve the accuracy more, by adding protein property with primary structure and it gives us 91.6% for SSP accuracy and 82.2% for DSSP which is 2% better than best result achieved till now.

Copyright by  
Shayan Ihsan Jalal

2018

## **DEDICATION**

“Education is the key to unlocking the world, a passport to freedom.” - Oprah Winfrey

I dedicate this book to all readers who may seek more knowledge. I have a serious reason why I am dedicating this to you: You may looking for very short, specific information to improve your scientific life, and you might find it here.

## **ACKNOWLEDGEMENT**

I would like to thank, first of all, my family; without them I am nothing. I shall spend the rest of my life giving back what I've received. First, I want to dedicate this to my mother Sharmeen Arif; thank you to always be there for me. Also, I want to dedicate this to my father Ihsan Jalal, and my big brother Dr. Rawand; I will always appreciate your giddiness.

I dedicate this work and give special thanks to my best friend and husband Mohamed Shukralla for being there for me throughout the entire master program and my wonderful daughter Mayar. Both of you have been my best cheerleaders.

I would like to thank my advisor: Dr. Bill Zhong, for his giddiness and support through this thesis project. Your guidance was motivational, and your encouragement was never\_ending.

I want to thank Dr. Kumar, Dr. Hovsepian, and Ms. Doris because of how they supported me.

I want to thank my big family and friends.

I want to express gratitude to staff and faculty in the Computer Science Department, the ESL Department, the Writing Center, and the International Office.

Also, I want to give thanks to staff and faculty in HCED Iraq because they trusted me and gave me an opportunity to finish my degree.

## TABLE OF CONTENTS

Abstract.....	iii
Dedication.....	vi
Acknowledgements.....	vii
Table of Contents.....	viii
List of Tables.....	xiii
List of Features.....	xiv
 Chapter 1: Introduction.....	 1- 4
 Chapter 2: Protein Structure and Properties.....	 5-12
Primary Structure.....	5
Secondary Structure.....	6
Tertiary structure.....	7
Quaternary Structure .....	7
Problem Formulations in Relation to Protein Structures.....	8
Amino Acid properties.....	9
First: Amino acid charge .....	9
Second: Amino acid hydrophobicity.....	9
Third: Polar Hydrophilic Amino Acids .....	10
Forth: aromatic amino acids .....	11
A: hydroxyl group .....	11
B: carbonyl group.....	12

C: carboxyl group.....	12
D: sulfhydryl group .....	12
Chapter 3: Problem Formulations and Priuses work.....	13-17
How Secondary structure is defined.....	13
DSSP.....	14
SSP.....	14
SAR.....	14
SAA .....	14
Privies work .....	15
A / Templet base.....	15
B/ Templet Free.....	15
SPINE-X.....	15
SPIDER2.....	16
DeepCNF (raptor x).....	16
MUST-CNN.....	17
Chapter Four: artificial neural networks (ANNs).....	18 - 44
Convolution neural network architecture.....	19
Convolution neural network and protein analyses.....	21
Convolutional networks.....	21
Pooling layer.....	22
Dropout.....	23



Merge.....	24
Fully-connected.....	26
The data.....	27
4protein.....	27
Cb 513 CullPDB.....	27
Methods and Materials.....	28
A/ Protein structure dictation using just the primary structure.....	29
First Method: Protein structure dictation with padding.....	29
Second Method: Protein structure dictation without padding.....	32
B/ Protein structure dictation using primary structure and one protein property.....	34
First Method: Protein structure dictation by merging primary structure and one protein property using symmetric tree.....	34
Second Method: Protein structure dictation by merging primary structure and one protein property using asymmetric tree.....	40
Chapter five: requirements & tools.....	45 - 46
Software Requirements.....	45
Hardware Requirements.....	45
First system.....	46
Second system .....	46

Chapter six: results and discussion.....	47- 56
A/ Protein structure dictation using just the primary structure.....	47
First Method: Protein structure dictation with padding.....	47
Second Method: Protein structure dictation without padding.....	49
B/ Protein structure dictation using primary structure and one protein property.....	51
First Method: Protein structure dictation by merging primary structure and one protein property using symmetric tree.....	51
Second Method: Protein structure dictation by merging primary structure and one protein property using asymmetric tree.....	54
Chapter seven: summary and future work.....	57 - 59
Summery.....	57
Future work.....	59
List of Reference.....	60 - 70
Vitae.....	71 - 72

## LIST OF TABLES

Table 4.1.....	31
The architecture of best model of protein structure dictation with padding	
Table 4.2.....	34
The architecture of best model of protein structure dictation without padding	
Table 4.3.....	37
The architecture of best model of Protein structure dictation using symmetric tree.	
Table 4.4.....	42
The architecture of best model of protein structure dictation using asymmetric tree.	
Table 6.1.....	47
The result of models of protein structure dictation with padding method.	
Table 6.2.....	49
The result of Models of protein structure dictation without padding method.	
Table 6.3.....	51
Accuracy table for Protein structure dictation without padding	
Table 6.4.....	51
The result of models of Protein structure dictation using symmetric tree method	
Table 6.5.....	52
The result of different ways merge layer in symmetric tree	

Table 6.6.....	53
The result of best model of protein structure dictation using symmetric Tree method	
Table 6.6.....	53
The result of small data of protein structure dictation using merge layer method	
Table 6.7.....	54
The accuracy table for symmetric tree	
Table 6.8.....	55
The result of different ways merge layer in asymmetric tree	
Table 6.9.....	55
The result of best model of protein structure dictation using asymmetric tree method.	

## LIST OF FIGURES

Figure 2.1.....	6
Primary structure of a protein (Zvelebil & Baum, 2008).	
Figure 2.2.....	6
The secondary structure of a protein (Zvelebil & Baum, 2008).	
Figure 2.3.....	7
The tertiary structure of a protein (Zvelebil & Baum, 2008).	
Figure 2.4.....	8
Quaternary structure of a protein (Zvelebil & Baum, 2008).	
Chart 6.1.....	48
This chart shows the differences between models accuracy of protein structure dictation with padding.	
Chart 6.2.....	50
This chart shows the differences between the model accuracy between big data and small data.	
Chart 6.3.....	55
This chart shows the accuracy difference between symmetric tree and asymmetric tree	

## CHAPTER ONE: INTRODUCTION

Proteins are made up of polymer sequences differing in lengths and number of polypeptide chains consisting of 20 amino acid residues. In some protein sequences, there are rare amino acids named Selenocysteine (Sec) and pyrrolysine (Pyl) that known as the 21st and 22nd amino acids in the natural code (Zhang & Gladyshev, 2007). So there are 22 amino acids in a protein sequence as we will see in the data we use in this paper. The combination of amino acid and the unique structure of the protein's causes fold, which makes different 3D structure shape and allows it to carry out various life functions such as acting as antibodies, molecular recognition, catalysis, coagulation and other functions (Sillitoe, Lewis & Cuff, 2015; Yang, Gao, Wang, Heffernan, Hanson, Paliwal & Zhou, 2016). From here, make it clear how protein functions require an understanding of its structures. As known, protein divided into four main structures: primary structure, secondary structure, which will be the main focus of this paper, tertiary structure, and quaternary structure.

Over 200 million protein sequences have been huddled in the GenBank, at the same time only 100 000 protein structures have been deposited in Protein Data Bank (Benson, Clark & Karsch-Mizrachi, 2015); predicting protein structures computationally is the only practical solution, as a result of the high cost of protein structure determination (\$100,000 for protein) and low cost of whole-genome sequencing (\$1,000) (Terwilliger, Stuart & Yokoyama, 2009; Mardis, 2006). The most popular subproblem of protein structure prediction is the prediction of protein secondary structure. Moreover, knowing secondary structures provides a good idea about overall structural categories. Secondary structure

plays an important role in predicting how proteins fold and how fast it folds (Plaxco, Simons & Baker, 1998; Zhou & Karplus, 1999; Ozkan, Wu & Chodera, 2007). As a result, the accuracy of protein secondary structure prediction has a direct impact on the accuracy of protein structure prediction (Fischer & Eisenberg, 1996; Wu, Skolnick & Zhang, 2007). However, it is challenging to reliably predict 3D structures from protein sequences without using known structures as templates (Yang, Gao, Wang, Heffernan, Hanson, Paliwal & Zhou, 2016). Thus, the protein structure prediction problem has been divided into smaller subproblems, with the expectation that by splitting it into smaller subproblem, the solution to the bigger problem will be easier. The accuracy of protein structure prediction method depends on how the secondary structure is defined. The most commonly used standard method is the secondary structure assignment methods that include three class Secondary Structure Prediction (SSP) and Dictionary of Secondary Structure of Proteins (DSSP), which automatically assigns the secondary structure into eight states according to hydrogen-bonding patterns (Kabsch & Sander, 1983).

Protein structure prediction began in 1951 when Pauling and Corey predicted helical and sheet conformations for protein polypeptide backbone, and even earlier the first protein structure was determined. Sixty-five years after, powerful new methods push the field of protein prediction into a new era. The highest three-state accuracy without relying on structure templates is now at 89%, a number unthinkable just a few years ago. These improvements came from constantly increasing databases of protein sequences and structures for training, the use of template secondary structure information, and more powerful analytical learning techniques.

By using neural network (NN) a new machine learning method gave a beneficial boost to predicting protein sequence gradually risen from 69.7% by PhD in 1993, 76.5% by PSIPRED in 1999, 80% by Structural Property prediction with Integrated Neural network (SPINE) in 2007, 82% by Structural Property prediction with Integrated deep neural network 2 (SPIDER2) in 2015, to 84% for several test data sets by Deep Convolution Neural Field network (DeepCNF) in 2016, to 89% for A Multilayer Shift-and-Stitch Deep Convolutional Architecture (Rost & Sander, 1993; Jones, 1999, Dor & Zhou, 2007; Heffernan, Paliwal & Lyons, 2015; Wang, Peng, Ma & Xu, 2016; Lin, Lanchantin & Qi, 2016). Moreover, using a convolutional neural network CNN can tag the properties of each amino acid in the full target sequence all at once. CNN has been strongly used in computer vision and natural language processing (Pinheiro & Collobert, 2013; Szegedy, Liu, Jia, Sermanet, Reed, Anguelov, Erhan, Vanhoucke & Rabinovich, 2014; Kim, 2014; Collobert & Weston, 2008). CNN's are also highly parallelizable. Thus, CNN's can achieve a much greater speedup compared to a windowed multilayer perceptron (MLP) approach. The issue when trying to label each position in an input sequence with a CNN is that pooling leads to a decreased output resolution; to solve this issue we omit pooling layer in some point in our project to make sure the protein sequence size not effected through the model we built.

It is easy to verify the sequence of a protein; however, it is difficult to determine the various properties, one as the secondary structure and the others having solvent accessibility. These properties are hypothesized to be almost uniquely determined by the primary structure; nonetheless, it is more computationally deep to determine them on a large scale. By testing convolution neural network we add more layers to it, we can get



better results. However, using the convolution neural network with windowing to predict protein secondary structure has been used before. In this paper, we use the multi-input convolution neural network to improve the secondary structure prediction accuracy by adding some protein properties like water solvent accessibility and the protein's primary structure. Empirically, our model achieves better results on the same dataset been used in Must\_CNN (Lin, Lanchantin, & Qi, 2016), which gave the best result until now.

## **CHAPTER 2: PROTEIN STRUCTURES AND PROPERTIES**

To successfully understand protein classification, first we have to understanding fully what is being classified. Proteins are the roughly versatile macromolecules in living systems and perform crucial functions in most all biological processes. They work as catalysts, they replace and store other molecules a well-known as oxygen, they give mechanical support and immune protection, they generate movement, they transmit nerve impulses, and they control growth and differentiation. The knowledge of geometric model representation for a protein makes heavy use of the biological properties of these protein, thus, it is necessary to know these facts before formulating the problem. Protein chains are counting 22 amino acid residues and herewith, for a polypeptide chain of length  $X$  , the total number of ready willing and able sequences is a typically  $22^x$  . Protein structures are formally isolated into four levels. They are called primary structure, secondary structure, tertiary structure and quaternary structure.

### **Primary Structure**

This structure contains a chain of amino acids. The amino acid sequence information accessible in this stage is suited to test the structure it will fold into as well as its function in nature. For instance, Karp and Geer (2005) describes the case where the change in the sequence of hemoglobin causing sickle cell anemia which is caused by a singular twist in the sequence. Therefore, it is theorized that all amino acid sequences stay in a particular protein is uniquely like the structure of that protein

*GLU – GLU – GLN – VAL – ALA – PRO – PRO – LYS – ASP – PRO ... ..*



Figure 2.1 Primary structure of protein (Zvelebil & Baum, 2008)

## Secondary Structure

This structure is actually the first level of protein folding .In this level construct two conformation called  $\alpha$ -helix and  $\beta$ -sheets. The parts of the polymer chain that does not spin into either conformation becomes loops and turns and this entire structure is held combined by hydrogen bonding between each peptide unit. The 3D structural information accessible in this level besides brings more information about the nature of the protein. For example, a protein especially soluble in water will typically have its hydrophobic residues inside the  $\alpha$ -helix and hydrophilic ones on the outside. The  $\beta$ -sheets or  $\beta$ -strands typically forge alongside one another in the structure and are further held together by hydrogen bonds.

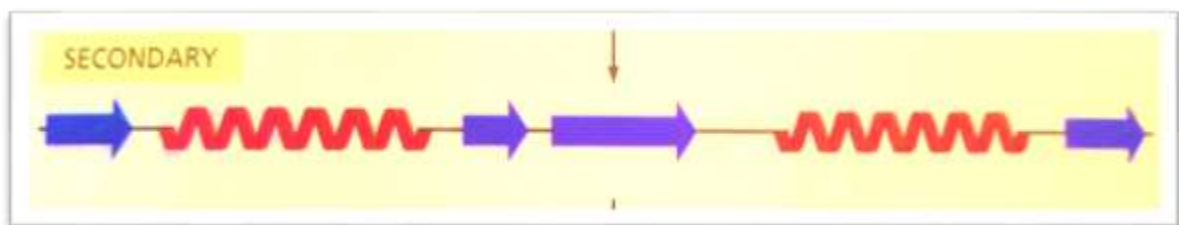


Figure 2.2 Secondary structure of a protein (Zvelebil & Baum, 2008)

## **Tertiary structure**

This structure is a level above the secondary structure which form by further folding and packing together and contains the whole backbone structure of the chain along mutually its secondary conformations. Here the unique final three dimension of protein is form, which can still be determined from its primary and secondary structures.

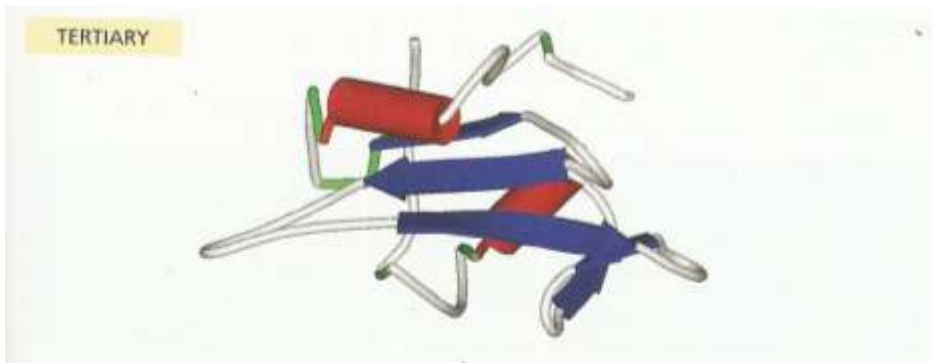


Figure 2.3 Tertiary structure of a protein (Zvelebil & Baum, 2008)

## **Quaternary Structure**

This structure adopted by protein chain and its function. It involves the clustering of several isolated peptide or protein chains into a certain shape. A deviation of bonding interactions including hydrogen bonding, ionic bridges, and disulfide bonds boost the various chains facing a particular geometry. It is impossible to predict the folded structure of protein from its amino acid sequence alone. Solving this problem is one of the challenges face bioinformatics

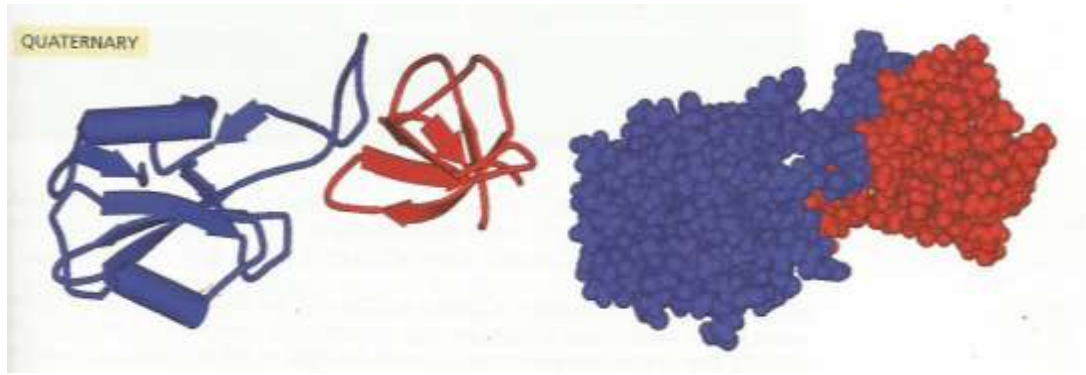


Figure 2.4 Quaternary structure of protein (Zvelebil & Baum, 2008)

## Problem Formulations in Relation to Protein Structures

In computational biology, generally four levels are studied widely and algorithms have been developed to predict the following. The syllabus behind solving one problem is elaborated furthermore in the following chapter.

- Predicting secondary or tertiary structure from primary structure.
- Predicting tertiary structure from secondary structure.
- Predicting the biological function of protein, actually its quaternary structure rather than its physical conformation via information available in the primary, secondary and tertiary structures.
- Designing the primacy structure via information from the secondary and tertiary and quaternary structures.

In our project we work on secondary structure from primary structure first by just using protein amino acid and after word by adding amino acid properties as we discuss below.

## **Amino Acid properties**

Amino acids are the building blocks of peptides and proteins, and interim they generally have commonplace elements of an amine collection, a carboxyl group and a side chain, the various rational groups that form the side chain give each amino acid different physical properties that affect protein production and function. The different amino acids fall into three major classes: hydrophobic, hydrophilic, and charged residues. Here some of these properties:

### **First: Amino acid charge**

Each amino acid have charge property as shown below:

1. Polar and negative charge (aspartic acid and glutamic acid).
2. Polar and positive charge (arginine, lysine, and histidine).
3. Polar and uncharged (asparagine, glutamine, serine, threonine, tyrosine)
4. Nonpolar (alanine, glycine, valine, leucine, isoleucine, proline, phenylalanine, methionine, tryptophan, cysteine)

### **Second: Amino acid hydrophobicity**

Hydrophobic, as the name implies is hydro – water, phobic – fearing. Hydrophobic amino acids have little or no polarity in their side chains. The lack of polarity means they have no way to interact with highly polar water molecules, making them water fearing. Amino acids are grouped according to what their side chains are like. The nine amino acids that have hydrophobic side chains are glycine (Gly), alanine (Ala), valine (Val), leucine (Leu),

isoleucine (Ile), proline (Pro), phenylalanine (Phe), methionine (Met), and tryptophan (Trp) (Aftabuddin & Kundu, 2007).

### **Third: Polar Hydrophilic Amino Acids**

Hydrophilic as the name implies comes from hydro-water and phallic – loving. Polarity comes from a 0.5-1.9 difference in electronegativity between bound atoms. The electronegativity difference is enough to create a slight separation of charge or polarity. And since like attracts like, these partially charged groups will be attracted to oppositely charged or partially charged groups such as water. These groups will twist the polypeptide chain in order to interact with each other and with water (Aftabuddin & Kundu, 2007).

**The accessible surface area (ASA)** or solvent-accessible surface area (SASA) is the surface area of a biomolecule that is accessible to a solvent. Measurement of ASA is usually described in units of square Ångstroms (a standard unit of measurement in molecular biology). ASA was first described by Lee & Richards in 1971 and is sometimes called the Lee-Richards molecular surface. ASA is typically calculated using the 'rolling ball' algorithm developed by Shrake & Rupley in 1973. This algorithm uses a sphere (of solvent) of a particular radius to 'probe' the surface of the molecule (Accessible surface area, 2017).

**Relative accessible surface area or relative solvent accessibility (RSA)** of a protein residue is a measure of residue solvent exposure. It can be calculated by formula (Michael & Gromiha, 2010):

$$ASA = \sum [R / (R^2 - Z_i^2)^{1/2}] \cdot Li \cdot D; D = \Delta Z / 2 + \Delta' Z,$$

**Forth: aromatic amino acids**

Aromatic amino acids [phenylalanine (Phe), tyrosine (Tyr), and tryptophan (Trp)] provide a surface of negative electrostatic potential than can bind to a wide range of cations through a predominantly electrostatic interaction. Aromatic amino acids have unique and important properties. Phenylalanine (Phe), tyrosine (Tyr), and tryptophan (Trp) are generally hydrophobic, but compared with simpler hydrophobic residues, such as leucine or valine, the aromatic amino acids have additional capabilities. Both Tyr and Trp can contribute hydrogen bonds, an important feature. However, there is another reason that aromatic amino acids are substantially overrepresented at protein binding sites and that Trp, in particular, is the most conserved of all amino acids. That is the cation- $\pi$  interaction, a strong, noncovalent binding interaction that contributes to protein secondary structure and to diverse drug-receptor interactions (Dougherty, 2007). There are six functional groups that are most important to the chemistry of life: hydroxyl, carbonyl, carboxyl, sulfhydryl, and phosphate groups:

**A: hydroxyl group**

In a hydroxyl group (-OH), a hydrogen atom forms a polar covalent bond with an oxygen which forms a polar covalent bond to the carbon skeleton. Because of these polar covalent bonds hydroxyl groups improve the solubility of organic molecules. Organic compounds with hydroxyl groups are alcohols and their names typically end in (ol).



**B: carbonyl group**

A carbonyl group ( $=\text{CO}$ ) consists of an oxygen atom joined to the carbon skeleton by a double bond. If the carbonyl group is on the end of the skeleton, the compound is an aldehyde. If not, then the compound is a ketone. Isomers with aldehydes versus ketones have different properties.

**C: carboxyl group**

A carboxyl group ( $-\text{COOH}$ ) consists of a carbon atom with a double bond with an oxygen atom and a single bond to a hydroxyl group. Compounds with carboxyl groups are carboxylic acids. A carboxyl group acts as an acid because the combined electronegativity of the two adjacent oxygen atoms increase the dissociation of hydrogen as an ion ( $\text{H}^+$ ).

**D: sulfhydryl group**

A sulfhydryl group ( $-\text{SH}$ ) consists of a sulfur atom bonded to a hydrogen atom and to the backbone. This group resembles a hydroxyl group in shape. Organic molecules with sulfhydryl groups are thiols. Sulfhydryl groups help stabilize the structure of proteins.

## **CHAPTER THREE: PROBLEM FORMULATIONS AND PRIVUSE WORK**

One of most popular subproblem of protein structure prediction is the prediction of protein secondary structure, or the local conformation of a protein's polypeptide backbone. Solving the secondary structure prediction problem is important because protein tertiary structures are classified facing structural folds according to how secondary structure elements (helices and sheets) are packed and permutated (Yang, Gao, Wang, Heffernan, Hanson, Paliwal & Zhou, 2016). In other words, knowing the structure provides a simulate idea about overall structural categories. Moreover, secondary structure plays an important role in determining how proteins fold and how fast they fold (Plaxco, Simons & Baker, 1998).

There are two main methods used for improving accuracy of protein structure prediction. First, Many recent methods improve accuracy of protein structure prediction by using actual secondary structures of homologous sequences (known as template-based approaches) best of them in this field is Protein Secondary Structure Prediction with Homology Analysis (SSpro) as we will discuss it in later this paper. Second, More methods now days try to improve the accuracy without using alike sequence, and this method we will use in our project as many other project used like: SPIDER2, DeepCNF (raptor x) and MUST-CNN. However, first we need to know how the secondary structure is defined

### **How Secondary structure is defined:**

The accuracy of secondary structure prediction depends on how secondary structure is defined in tow major category:

### **DSSP Dictionary of Secondary Structure of Proteins**

Generally used standard is the secondary structure assignment method Dictionary of Secondary Structure of Proteins (DSSP), which automatically assigns secondary structure facing eight states according to hydrogen-bonding patterns. The class labels are: H = alpha helix, B = residue in isolated beta bridge, E = extended strand, G = 3-helix, I = 5-helix, T = hydrogen bonded turn, S = bend, L = loop (Kabsch & Sander, 1983; Lin, Lanchantin & Qi, 2016).

### **SSP Secondary Structure of Proteins**

DSSP eight states are often besides simplified into three states of helix, sheet and coil which named Secondary Structure of Proteins (SSP). The most widely used convention is that helix is designated as G (310 helix), H (a-helix) and I (p-helix); sheet as B (isolated bridge) and E (extended sheet); whatever other states designated as a coil.

### **SAR Relative solvent accessibility**

Given the most solvent accessible amino acid in the protein has  $x$  °A of accessible surface area, we label other amino acids as solvent accessible if they have greater than  $0.15x$  °A of accessible surface area.

### **SAA Absolute solvent accessibility**

Defined as the amino acid having more than  $0.15$  °A of accessible surface area.

## **Privies work:**

### **A / Templet base**

Protein Secondary Structure Prediction with Homology Analysis SSpro/ACCpro 5: This method improve accuracy by using actual secondary structures of homologous sequences (known as template-based approaches). This method use a combination of machine learning and profiles, and thus must be retrained and assessed periodically as the number of available protein sequences and structures continues to grow. This method achieve results in two category one the order of 80% accuracy, without using any similarity with known proteins. The other category is achieving on the order of 85% accuracy, using sequence similarity alone. However, when sequence-based structural similarity is added, the accuracy of SSpro rises to 92.9% (Magnan & Baldi, 2014).

### **B/ Templet Free**

**SPINE-X:** A neural network was used to predict secondary structure by using PSSM and reside of physiochemical properties of amino acids (PPAA) (Faraggi, Zhang & Yang, 2011). Then, PSSM and PPAA together mutually predicted secondary structures were used to predict solvent accessibility. Afterwards, PSSM and PPAA together with predicted lesser structures and predicted solvent accessibility was used to predict torsion angles. These predicted structural properties were used together with PSSM and PPAA to predict secondary structure for the second time. This rite is extended so that the secondary structure

is predicted for the third time. This iterative manner achieves 82% in 10-fold cross validation for a non-redundant dataset of 2640 proteins (CASP9).

**SPIDER2:** applied deep neural networks to secondary structure prediction. Three layers were used in SPIDER2 (Rost & Sander, 1993). This method used an iterative improvement of secondary structure, backbone torsion angles and solvent accessibility at the same time in three iterations. The method achieves 81.8% for the independent test on 1199 high-resolution proteins ( $<2.0\text{\AA}$ ). When comparing SPIDER2 with SPINE-X, PSIPRED 3.3 and SCORPION in the independent test set of 1199 proteins as well as the CASP11 set, about  $>1\%$  improvement over other methods was observed. Interestingly, SPINE-X is more accurate in predicting helical residues, whereas PSIPRED is more accurate in predicting coil residues. SPIDER2 was not the first method that used deep neural networks for secondary structure prediction but is the most accurate according to reported accuracy (Heffernan & Paliwal & Lyons, 2015).

**DeepCNF (raptor x):** this method is the first approach using deep convolutional neural fields for secondary structure prediction (Wang, Peng, Ma & Xu, 2016). It stacks deep convolutional neural networks mutually with conditional random field model on top as the output layer. The deep convolutional neural networks were used to discover the complex sequence– structure relation of longer sequence separation than a typical deep neural network, whereas the conditional random field model allows detection of barrier of secondary structure among neighboring residues. It uses five hidden layers and an 11-

residue window to achieve the best training. The method was trained on 5600 proteins with 25% sequence identity. The accuracy of test sets ranges from 82.3 to 85.4%.

**MUST-CNN:** A Multilayer Shift-and-Stitch Deep Convolutional Architecture for Sequence-based Protein Structure Prediction convolution neural network. MUST-CNN a Multilayer Shift-and-Stitch Deep Convolutional Architecture: this method is first method uses convolution neural network. It is a 1D classification algorithm which takes protein sequence as input and predicts at amino acid level. Authors have used Torch7 as framework. Model uses three convolutional layers and multitasking approach to tag amino acids in such of the many classes in one go. It describes a multilayer shit-and-stich convolutional neural network architecture for predicting protein properties at amino acid level from the protein sequences. The subject was borrowed form image classification over deep CNN on per position sequence labelling and implemented first time on protein sequence to predict amino acid level properties. Algorithm shifts the amino acid sequence according to pooling in each layer and stich after convolution layers to get the deep embedding for each amino acid. Authors used whole sequence at once, while previous use windowing methods. Deep embedding are fed to multitask linear layers which were vary for individual task. CNN They transformed the input by shifting and padding followed by convolution nonlinearity (ReLU function), max pooling as pooling technique. Output array is passed on a dropout layer (acting as regularization) which is a randomized mask of outputs. Dropout is moved in testing whatever eights are used. The accuracy for DSSP is 76.7% and for SSP is 89.6% (Lin, Lanchantin, & Qi, 2016).

## **CHAPTER FOUR: ARTIFICIAL NEURAL NETWORKS (ANNS)**

Artificial neural networks (ANNs) is fast growing area of research in the application of protein classification. Artificial neural networks are perhaps the most widely used artificial intelligence tools today, it applied for classification and on real-time image and sound processing applications. ANNs contain artificial neurons as basic building blocks, each of which computes a non-linear function of the weighted sum of its inputs. This non-linear function is named the activation function. Then the output of the neuron may be fed to another neurons as input. The neurons in the first layer (called the input layer) work on the input of the network. The output of the network is computed by the output layer. When the problem to solve is a classification task, each class is assigned a different neuron in the output layer, which is activated if the input is classified into the corresponding class.

Convolutional Neural Networks (CNN) is one of methods in deep artificial neural network. It has been used vastly in image recognition domain, through using a deep multilayer convolutional network. It has shown the state-of-the-art on image classification gives very high percentage in the ImageNet Large-Scale Visual Recognition Challenge. CNN has been recently used in natural language processing. Where methods produced to label word properties, such as part of speech or category of a named entity, on text data. Same apply on bioinformatics protein sequence analyses, if we consider each protein chain to be a sentence and each amino acid to be a word, the techniques transfer easily.

Long-short term memory networks have been used very successfully in sequence learning, machine translation, and language modeling (Lin, Lanchantin & Qi, 2016). We note that machine translation is much more general sequence to sequence task where the

input and output sizes are not matched. Language modeling tries to guess future words based on past words, while protein sequences has no innate direction.

### **Convolution neural network architecture**

The building blocks of neural networks are the artificial neurons, they putted as a set of neuron layers. The output of a layer becomes the input of the after layer or the output of the entire network (for the last layer). A layer is a parametric field with learnable parameters. The network is the design of these functions, and itself can be heart of as a parametric field  $f(n)$  with  $n$  as the weight parameter vector that assigns the weights of the inputs of each artificial neuron in the network. Today's neural networks are generally deep neural networks, meaning that they have a larger number of layers than before variations, resulting in an unusually increased learning capacity. If such specifies the non-linear activation functions of the neurons, and the architecture of the network, before the neural network can be trained to perform its categorization task. This learning capacity is the most appealing plot of neural nets. The weights of the neuron inputs, for example, the vector is improved step-by-step, as described below.

Neural networks are usually trained under supervised environment by inputting specific  $x$  values into it and back propagate the error  $\hat{y} - f_n(x)$ . This means that a loss function is applied on the network output  $y = f_n(x)$  and the desired output  $\hat{y}$ , and the parameters of the network are updated with lean descent .The network input  $x$  can be modeled as a random variable, and  $\hat{y} = g(x)$  is a function of  $x$  that needs to be approximated by the network. Let  $\varepsilon(y, \hat{y})$  known a loss function for example  $\varepsilon(y, \hat{y}) = (y - \hat{y})^2$  is our formula, the L2 is loss function. Then the loss of the network can be written as



$$l(n) = E_x \varepsilon(f_n(x), g(x))$$

This formula can then be nearly alternating the expected value if  $x_1, \dots, x_n$  a random sample of inputs, then an approximation of the above formula is

$$\tilde{l}(n) = \frac{1}{n} \sum_{i=0}^n \varepsilon(f_n(x_i), g(x_i))$$

Updating the network weights can be done with stochastic gradient descent (SGD).

Using the update step

$$n_{k+1} = n_k - \gamma \frac{\alpha_l}{\alpha_n} (n_k)$$

The initial value  $n_0$  is initialized randomly. If the learning rate  $\gamma$  is sufficiently small,  $n_k$  will then converge to a place of local minimum.

Stochastic create family is relative to stalling near rap points in the inaccuracy surface, causing slow converge. Furthermore, the degree of the update steps (the differences in  $n$ ) can be too large if  $\gamma$  is too big, and this manage show in divergence. Therefore SGD has over been improved by introducing momentum or per the statistics of the gradients to normalize the update steps and thus yielding the more new methods such as RMSProp, Adagrad or Adam. Classification problems with the inputs possibly corresponding to multiple classes are called multi-label classification problems. When we seek to classify proteins via their amino-acid sequences as inputs and deferent functional or structural classes as outputs, we further need multi-label classification procedures, as a protein may be assigned a well-known or more functional or structural classes.

## Convolution neural network and protein analyses

**Convolutional networks:** are famous for doing image recognition of 2D. Using same approach, we use a 1D convolution for the protein sequence labeling problem. A convolution on sequential data tensor  $X$  of size  $T * n_{in}$  with length  $T$ , kernel size  $k$  and input hidden layer size  $n_{in}$  has output  $Y$  of size  $T * n_{out}$ :

$$Y_{t,i} = \partial(B_i \sum_{j=1}^{n_{in}} \sum_{z=1}^k W_{i,j,k} X_{t+z-1,j})$$

$W$  and  $B$  are the trainable parameters of the convolution kernel, and  $\partial$  is the nonlinearity. There are many different nonlinearity functions, but three of them are most popular: the hyperbolic tangent, rectified linear units (ReLU), and piecewise rectified linear units (PReLU). The hyperbolic tangent is historically the most used in neural networks, because it has nice computational properties that make optimization easy. Both ReLU and PReLU are work very well on deep convolutional networks for object recognition. ReLU is perform better than tanh on the same tasks, and enforces small amounts of sparsity in neural networks (Lin, Z., Lanchantin & Qi, 2016).

The ReLU nonlinearity is defined as

$$relu(x) = \max(0, x)$$

And the PReLU nonlinearity is defined as

$$prelu(x) = \begin{cases} \alpha x & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$$

With a trainable parameter  $\alpha$ .

**Pooling layer:** Its function is to constantly reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and to control overfitting. The pooling strategy tested was max pooling and global max pooling. Max pooling has shown to perform much better than subsampling as a pooling scheme (Lin, Lanchantin & Qi, 2016) and has generally been the preferred pooling strategy for large scale computer vision tasks. Max pooling on a sequence  $Y$  of size  $T \times n$  with a pooling size of  $m$  results in output  $Z$  where:

$$Z_{t,i} = \max_{j=0}^m Y_{m(t-1)+j,i}$$

Global max pooling is same ordinary max pooling layer with pool size equals to the size of the input. For example, if the input of the max pooling layer is 0,1,2,2,5,1,2 global max pooling outputs 5, whereas ordinary max pooling layer with pool size equals to 3 outputs 2,2,5,5,5 assuming stride is 1 (Le, 2015).

Removing pooling layer. Many people dislike the pooling process and picture that we can build model without it. For example, striving for simplicity: The All Convolutional Net proposes to reject the pooling layer in parallel of architecture that only consists of steady CONV layers. To minimize the quantity of the random sample they symbolize using larger stride in CONV layer once in a while. Discarding pooling layers has further been found to be related in training useful generative models, such as variation auto encoders (VAEs) or generative adversarial networks (GANs). It seems likely that future architectures will feature very few to no pooling layers. (CS231n Convolutional Neural Networks for Visual Recognition, n.d.).

**Dropout:** is a technique that prevents overfitting. The phrase “dropout” refers to dropping out units hidden and visible in a neural network. By dropping a unit out, we express temporarily removing it from the network, along with all its incoming and outgoing connections. The choice of which units to drop is random. In the simplest status, each unit is retained mutually a solid probability  $p$  independent of various units, where  $p$  can be chosen for a validation set or can just be fit at 0.5, which seems to be close to optimal for a wide range of networks and tasks. For the input units, anyway, the optimal probability of retention is regularly closer to 1 than to 0.5. Consider a neural network with  $L$  hidden layers. Let  $l \in \{1, \dots, L\}$  index the hidden layers of the network. Let  $Z^{(l)}$  known as the vector of inputs into layer  $l$ ,  $y^{(l)}$  known as the vector of outputs from layer  $l$  ( $y^{(l)} = x$  is the input).  $yW^{(l)}$  And  $b^{(l)}$  are the weights and biases at layer  $l$ . The feed-forward operation of a standard neural network can be described as (for  $l \in \{0, \dots, L-1\}$  and any hidden unit  $i$ ) (Srivastava, & Hinton, & Krizhevsky, & Sutskever, & Salakhutdinov, 2014)

$$z_i^{(l+1)} = w_i^{(l+1)} y^l + b_i^{(l+1)}$$

$$y_i^{(l+1)} = f(z_i^{(l+1)})$$

Where  $f$  is any activation function, for example,  $f(x) = 1/(1 + \exp(-x))$ . With dropout, the feed-forward operation becomes

$$r_i^{(l)} \sim \text{Bernpulli}(j)$$

$$\tilde{y}^{(l)} = r^{(l)} * y^{(l)}$$

$$z_i^{(l+1)} = w_i^{(l+1)} \tilde{y}^l + b_i^{(l+1)}$$

$$y_i^{(l+1)} = f(z_i^{(l+1)})$$

**Merge:** is used to connect two layers or tensors to gather, and it uses with functional API.

First what is sequential API? Sequential API give us the ability to create models layer-by-layer for most problems. This mean you create layers and add it to the model first layer output goes to the second layer as input in sequential mode. Its functionality limited in that it does not let us create models that share layers or have multiple inputs or outputs.

The functional API in Keras is another way to creating models that gives a lot more flexibility, by creating more complex models. It specifically allows us to define multiple input or output models as well as models that share layers. Moreover, it allows us to define a cyclic network graphs.

Merge or share layer as we say it connect layers or tensors, but how it do that? Well it connect the layers weights in many different ways: concatenate, add, multiply, and subtract.

In the concatenate way the two merged layers weights is put beside each other and feed like this to the fallow layer, for example let's say we have two convolution layers each with size 32; when we concatenate them in merge layer it size become 64 and feed to the next layer like this.

In the add way the two merged layers weights are sum mathematically to each other and feed like this to the fallow layer, for example let's say we have two convolution layers each with size 32(of course they must have same size); when we add the weights to each other

in merge layer it size stays 32 but with different weight value and feed to the next layer like this.

In the multiply way the two merged layers weights are multiplied mathematically to each other and feed like this to the fallow layer, for example let's say we have two convolution layers each with size 32 (of course they must have same size); when we add the weights to each other in merge layer it size stays 32 but with different weight value and feed to the next layer like this.

In subtract way the two merged layers weights are subtracted mathematically to each other and feed like this to the fallow layer, for example let's say we have two convolution layers each with size 32 (of course they must have same size); when we add the weights to each other in merge layer it size stays 32 but with different weight value and feed to the next layer like this.

This flexibility in merge layer give us the ability to build different models and connect them. These models may be symmetric or asymmetric, have same input or different inputs as a tree shape, and have same output or different output like tree roots shape.

This is done by specifying where the input comes from when defining each new layer. A bracket notation is used, such that after the layer is created, the layer from which the input to the current layer comes from is specified. Multiple layers can share the output from one layer.

For example, there may be multiple different feature extraction layers from an input, or multiple layers used to interpret the output from a feature extraction layer.

The functional API can also be used to develop more complex models with multiple inputs, possibly with different modalities. It can also be used to develop models that produce multiple outputs.

**Fully-connected:** Neurons in a fully connected layer have full connections to all activations in the previous layer. Therefore, this layer is used mostly at the end of the model. The activations can be computed with a matrix multiplication followed by a bias offset. The only difference between FC and CONV layers is that the neurons in the CONV layer are connected only to a local region in the input, and that many of the neurons in a CONV volume share parameters. However, the neurons in both layers still compute dot products, so their functional form is identical. Therefore, it turns out that it's possible to convert between FC and CONV layers. For any CONV layer there is an FC layer that implements the same forward function. The weight matrix would be a large matrix that is mostly zero except for at certain blocks (due to local connectivity) where the weights in many of the blocks are equal. Any FC layer can be converted to a CONV layer. For example, an FC layer with  $K=4096$  that comes from some input volume of size  $7 \times 7 \times 512$  can be expressed as a CONV layer with  $F=7, P=0, S=1, K=4096$ . In other words, we are setting the filter size to be exactly the size of the input volume, and the output will simply be  $1 \times 1 \times 4096$  since only a single depth column "fits" across the input volume, giving same result as the initial FC layer (CS231n Convolutional Neural Networks for Visual Recognition, n.d.).

## The data

I used two large protein property datasets in my experiments. The train is sixty percent of the data, validation is ten percent and test is thirty percent. The data is as follows:

### 1. 4protein

I got this data from (Lin, Lanchantin, & Qi, 2016) which it got it from (Qi et al, 2012). This data is used in most protein related papers. It has 11794 protein sequence varied in length the shortest sequence has 20 amino acid and the longest sequence is 2060 amino acid. I use a train validation test split where the model is trained on the training set, selected via validation set results, and best results reported by testing on the test set.

### 2. Cb 513 CullPDB

I got this data from (Lin, Lanchantin, & Qi, 2016) which they derived from (Zhou and Troyanskaya, 2014). This data is very important for checking the performance of any model because the CullPDB dataset has sequences with > 25% identity with the CB513 dataset was removed. This data contain 5534 protein sequence varied in length shortest sequence is consist from 12 amino acid and longest is consist from 696 amino acid.

Each amino acid is represented by a number and because we have 22 different amino acid, so we have numbers ranges between 1 and 22 as example below:

Protein sequence : { 21 4 12 12 12 8 13 10 12 12 10 17 7 7 4 6 6 12 4 6 7 12 }

This sequence I use as input to my model. Moreover, the data sets has structure property of SSP, DSSP, SAR (Relative Solvent Accessibility) and SAA (Absolute Solvent



Accessibility). SSP distinguishes between three protein structure (helix, sheet and coil) therefore the data has three classes as shown in the example bellow:

SSP structure: {1 2 2 2 2 2 1 1 1 1 1 1 2 2 2 2 2 1 1 3 1}

While DSSP has eight different classes G (310 helix), H (a-helix) and I (p-helix); sheet as B (isolated bridge) and E (extended sheet); whatever other states designated as a coil. This is its example:

DSSP structure: {1 2 2 2 2 2 3 3 4 1 1 1 2 2 2 2 2 3 3 6 1}

Moreover, SAR distinguish between the protein be water solvent or not depending size of its surface area, while SAA define that the protein is solvent assessable or not. So both have just two classes as example bellow:

SAR structure: {1 2 2 1 1 2 2 1 1 1 2 1 1 1 1 2 1 1 1 2 1}

SAA structure: {1 2 2 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1}

## **Methods and Materials**

We apply several methods and layers of convolution neural network, each layer with different window and filter size on the data described above. We use data in several ways once with padding to the longest protein sequence exist in the data and other without padding. Moreover, and last adding water solvent accessibility to the data to give better accuracy and we did this by using merge layers. Each of these methods had to solve a multilabel classification task. Where each amino acid represent a class, and each structure like helix, sheet and coil are a class for the label. Now let discusses each method in details.

## **A/ Protein structure dictation using just the primary structure**

### **First Method: Protein structure dictation with padding**

In this method we pad each sequence by zero to the longest sequence we have in the data which is 2060. Therefore, each sequence 2060 in length. When training the network, each training protein sequence was fed as the input (x) of the network, along with the classification target (y) of that sequence encoded as a 0-1 vector. Each class was represented as a coordinate in the target. The input sequences were encoded as an arrays of 3 dimensional inputSeq with dimensions [batch size, max length, dims]. Here dims is the one and, batch size means the number of sequences in a mini batch and was set to 10 or 100. Max length was the maximum length of a sequence: sequences shorter than this were padded in the training phase to the max length amino acids in the data. This parameter was set to 2060 in our case.

The deep neural network have a convolutional architecture with 1D spatial pyramid pooling and fully connected layers at the end. We build several models ten models to be exact; starting from three layer model one convolution layer fallowed by max pooling and in the last fully connected layer. After that we add two layers one convolution layer fallowed by max pooling with every model but in some models we add fully connected layer too; to find out which model gives the best results. Here is the models with their layers.

Model one: 1 convolution 1D layer, 1 pooling layer, flatten layer, 1 fully connected layer.

Model two: 2 convolution 1D layer, 2 pooling layer, flatten layer, 1 fully connected layer.

Model three: 3 convolution 1D layer, 2 pooling layer, flatten layer, 1 fully connected layer.

Model four: 1 convolution 1D layer, 1 pooling layer, flatten layer, 3 dropout layer, 3 fully connected layer.

Model five: 4 convolution 1D layer, 3 pooling layer, flatten layer, 1 dropout layer, 2 fully connected layer, 3 Batch normalize layer.

Model six: 4 convolution 1D layer, 3 pooling layer, flatten layer, 2 dropout layer, 2 fully connected layer, 4 Batch normalize layer.

Model seven: 4 convolution 1D layer, 4 pooling layer, flatten layer, 2 dropout layer, 2 fully connected layer, 4 Batch normalize layer.

Model eight: 5 convolution 1D layer, 4 pooling layer, flatten layer, 2 dropout layer, 2 fully connected layer, 3 Batch normalize layer.

Model nine: 5 convolution 1D layer, 4 pooling layer, flatten layer, 2 dropout layer, 3 fully connected layer, 4 Batch normalize layer.

Model ten: 6 convolution 1D layer, 5 pooling layer, flatten layer, 2 dropout layer, 3 fully connected layer, 4 Batch normalize layer.

By building ten models we discover that the model five with eleven layers gives the best result. The full architecture of is shown in Table 4.1. The network had 4 one-dimensional convolution layers with kernel sizes [6, 6, 5, 5] and depths (filter counts) [128, 128, 256, 256], with PReLU (parametric rectified linear unit) activation. We used max pooling with kernel size and stride 2 after each convolutional layer, except the first one. Max pooling was omitted after the first convolution layer.

After checking this method with padding we discover that this method effected so much with padding. For example when we delete the long sequence and make the longest sequence be 998; we see that the accuracy is reduced. So we try the next model without padding.

**Table 4.1:** The architecture of best model of protein structure dictation with padding.

Convolution 1D layer (size=6 depth=128, padding=VALID, activation=prelu)
Convolution 1D layer (size=6 depth=128, padding=VALID, activation=prelu)
max pool1D layer (size=2, stride=2, padding=VALID)
Convolution 1D layer (size=5 depth=256, padding=VALID, activation=prelu)
max pool1D layer (size=2, stride=2, padding=VALID)
Convolution 1D layer (size=5 depth=256, padding=VALID, activation=prelu)
max pool1D layer (size=2, stride=2, padding=VALID)
batch normalize layer (scale=False)
Flatten layer()
fully connected layer (units=1024, activation=prelu)
Dropout layer (p=0.5)
fully connected layer (units=2060, activation=prelu)

## **Second Method: Protein structure dictation without padding**

In this method we just pad the sequence by zero to the longest sequence in each batch after we sort the data from shortest sequence to the longest. Therefore, some of the sequence been padded just few zeros in the end of each batch so we can train it in the model. That is mean, each sequence differ in length from batch to batch but have same size in each batch. When training the network, each training protein sequence was fed as the input (x) of the network, along with the classification target (y) of that sequence encoded as a 0-1 vector. Each class was represented as a coordinate in the target. The input sequences were encoded as an arrays of 3 dimensional inputSeq with dimensions [batch size, none, dims]. Batch size means the number of sequences in a mini batch and was set to 10 or 100. None here mean that the size entered the model is different in size the maximum length of a sequence in each batch: sequences shorter than this were padded to the max length amino acids in the data. We encoded each amino acid as a 23-dimensional vector, where the first 22 components comprised as the amino acid sequence and the last one represent the zero padding. 4 classes for target if it is SSP 3 for the structure and 1 for zero padding, and 9 classes for target if it is DSSP 8 for the structure and 1 for zero padding.

The deep neural network have a convolutional architecture with 1D spatial pyramid pooling and fully connected layers at the end. We build several models ten models to be exact; starting from three layer model one convolution layer fallowed fully connected layer. After that we add layers of convolution in every model but in some models we add fully connected layer too; to find out which model gives the best results. Here is the models with their layers

Model one: 1 convolution 1D layer, 1 dropout layer, 1 fully connected layer.

Model two: 2 convolution 1D layer, 1 dropout layer, 1 fully connected layer.

Model three: 3 convolution 1D layer, 1 dropout layer, 1 fully connected layer.

Model four: 5 convolution 1D layer, 1 dropout layer, 1 Batch normalize, 1 fully connected layer.

Model five: 5 convolution 1D layer, 1 dropout layer, 1 Batch normalize, 2 fully connected layer.

Model six: 6 convolution 1D layer, 2 dropout layer, 2 Batch normalize, 3 fully connected layer.

Model seven: 7 convolution 1D layer, 1 dropout layer, 1 Batch normalize, 3 fully connected layer.

Model eight: 8 convolution 1D layer, 2 dropout layer, 2 Batch normalize, 3 fully connected layer.

Model nine: 8 convolution 1D layer, 3 dropout layer, 2 Batch normalize, 4 fully connected layer.

Model ten: 8 convolution 1D layer, 5 dropout layer, 3 Batch normalize, 6 fully connected layer

By building ten models we discover that the model three with five layers gives the best result. The full architecture of is shown in Table 4.2. The network had 3 one-dimensional convolution layers with kernel sizes [17, 16, 15] and depths (filter counts) [256, 128, 64], with PReLU (parametric rectified linear unit) activation. We used max

pooling with kernel size and stride 2 after each convolutional layer, except the first one. Max pooling was omitted.

After checking this method without padding we discover this method better than previous method and it is stable. Therefore, we use this model as standard and try to add more protein information to make it give better result like we will do in next two methods.

**Table 4.2:** The architecture of best model of protein structure dictation without padding

Convolution 1D layer (size=17 depth=256, padding=SAME, activation=prelu)
Convolution 1D layer (size=16 depth=128, padding=SAME, activation=prelu)
Convolution 1D layer (size=15 depth=64, padding=SAME, activation=prelu)
Dropout layer (p=0.5)
fully connected layer (units=4, activation=prelu)

After checking this method we discover this method gives better results than without adding more information to the model. However, we try share layer to find out if it gives better results.

## **B/ Protein structure dictation using primary structure and one protein property**

### **First Method: Protein structure dictation by merging primary structure and one protein property using symmetric tree**

In this method we use our previous best discovered model (model three with five layers) to build same model for protein primary structure and water solvent accessibility property in a symmetric tree, and we connect both models before fully connected layer using merge layer. Moreover, merge layer has three different ways to connect the inputs

first add or sum the inputs, second concatenate the inputs, and last multiply the inputs. We try all three ways of connection with same data to find out which one gives the best result. Just like the before models we pad the sequence by zero to the longest sequence in each batch after we sort the data from shortest sequence to the longest. Therefore, some of the sequence been padded just few zeros in the end of each batch so we can train it in the model. That is mean, each sequence differ in length from batch to batch but have same size in each batch. When training the network, each training protein sequence was fed as the input (x) of the network, and each training water solvent accessibility property was fed as the input (y) of the network, along with the classification target (z) of that sequence encoded as a 0-1 vector. Each class was represented as a coordinate in the target. The input sequences were encoded as an arrays of 3 dimensional inputSeq with dimensions [batch size, none, dims]. Batch size means the number of sequences in a mini batch and was set to 10 or 100. None here mean that the size entered the model is different in size the maximum length of a sequence in each batch: sequences shorter than this were padded to the max length amino acids in the data. We encoded each amino acid as a 23-dimensional vector, where the first 22 components comprised as the amino acid sequence and the last one represent the zero padding. Moreover, we encoded each the water solvent accessibility property as a 3-dimensional vector, where the first 2 components comprised as the water solvent accessible or not and the last one represent the zero padding. 4 classes for target if it is SSP 3 for the structure and 1 for zero padding, and 9 classes for target if it is DSSP 8 for the structure and 1 for zero padding.

The deep neural network have a convolutional architecture with 1D spatial pyramid pooling and fully connected layers at the end. In this method we did not add any flatten



layer because this layer need fixed size and we have varied size. Moreover, we did not add any max pooling layers because this layer reduce the size and this we do not want in our model. We build several merge models five to be exact, we add convolution layer before or after merge layer to find out which model gives the best results. Here is the models with their layers.

Model one: 3 convolution 1D layer before Merge layer, No convolution layer after, 1 dropout layer, 1 fully connected layer.

Model two: 2 convolution 1D layer before Merge layer, 1 convolution layer after, 1 dropout layer, 1 fully connected layer.

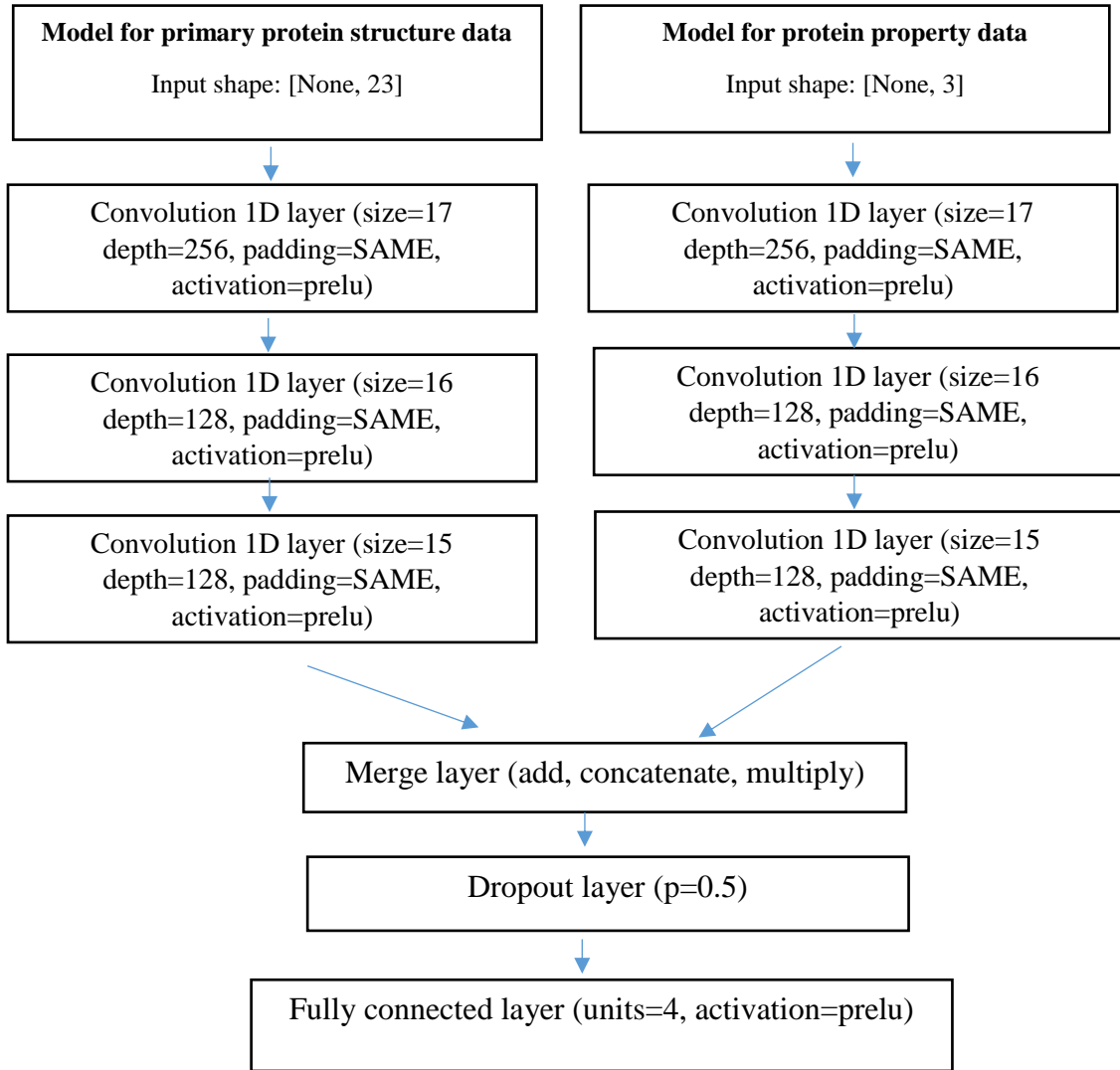
Model three: 1 convolution 1D layer before Merge layer, 2 convolution layer after, 1 dropout layer, 1 fully connected layer.

Model four: 1 convolution 1D layer before Merge layer, 3 convolution layer after, 1 dropout layer, 1 fully connected layer.

Model five: 3 convolution 1D layer before Merge layer, 3 convolution layer after, 1 dropout layer, 1 fully connected layer.

By building five models we discover that the model one with three 3 convolution 1D layers for each data before merge layer, and no convolution layer after it gives the best result. The full architecture of is shown in Table 4.3. The network had 3 one-dimensional convolution layers with kernel sizes [17, 16, 15] and depths (filter counts) [256, 128, 64], with PReLU (parametric rectified linear unit) activation.

**Table 4.3:** The architecture of best model of Protein structure dictation using symmetric tree.



After we find the best model, we try four different merge layer and here it is there output architecture. First, merge layers using add, this layer sum all the inputs come from above layers together of cores they must have same neurons number

Layer (type) Connected to	Output Shape	parameter#
=====		
input_1 (Input Layer)	(None, None, 23)	0

input_2 (Input Layer)	(None, None, 3)	0
conv1d_1 (Conv1D) input_1 [0] [0]	(None, None, 256)	104704
conv1d_4 (Conv1D) input_2 [0] [0]	(None, None, 256)	13312
conv1d_2 (Conv1D) conv1d_1 [0] [0]	(None, None, 128)	524416
conv1d_5 (Conv1D) conv1d_4 [0] [0]	(None, None, 128)	524416
conv1d_3 (Conv1D) conv1d_2 [0] [0]	(None, None, 64)	122944
conv1d_6 (Conv1D) conv1d_5 [0] [0]	(None, None, 64)	122944
add_1 (Add) conv1d_3 [0] [0]	(None, None, 64)	0
conv1d_6 [0] [0]		
dense_1 (Dense) add_1 [0] [0]	(None, None, 9)	585
=====		
Total parameters: 1,413,321		
Trainable parameters: 1,413,321		
Non-trainable parameters: 0		

Second, merge layers using concatenate, this layer it put all the inputs come from above layers beside each other of cores they must have same neurons number and this make the output of this layer doubled in size.

Layer (type) Connected to	Output Shape	parameter#
=====		
input_1 (Input Layer)	(None, None, 23)	0
input_2 (Input Layer)	(None, None, 3)	0
conv1d_1 (Conv1D) input_1 [0] [0]	(None, None, 256)	104704
conv1d_4 (Conv1D) input_2 [0] [0]	(None, None, 256)	13312

conv1d_2 (Conv1D)	(None, None, 128)	524416
conv1d_1 [0] [0]		
conv1d_5 (Conv1D)	(None, None, 128)	524416
conv1d_4 [0] [0]		
conv1d_3 (Conv1D)	(None, None, 64)	122944
conv1d_2 [0] [0]		
conv1d_6 (Conv1D)	(None, None, 64)	122944
conv1d_5 [0] [0]		
concatenate_1 (Concatenate)	(None, None, 128)	0
conv1d_3 [0] [0]		
conv1d_6 [0] [0]		
dense_1 (Dense)	(None, None, 9)	1161
concatenate_1 [0] [0]		

```

=====
Total parameters: 1,413,897
Trainable parameters: 1,413,897
Non-trainable parameters: 0

```

Third, merge layers using multiply, this layer multiply all the inputs come from above layers together of cores they must have same neurons number. As architecture it don't has any different between add architecture for that no need to repeat it. The last one, merge layers using subtract, this layer subtract all the inputs come from above layers together of cores they must have same neurons number. In this way we find out that using add method in merge layer give us the best result fallowed by concatenate, subtract and last multiply.

Here we try different kind of merge model add, concatenate, and multiply. After checking these methods we discover add method gives the best results without adding more information to the model. However, we try to find out if it gives better results.

## **Second Method: Protein structure dictation by merging primary structure and one protein property using asymmetric tree**

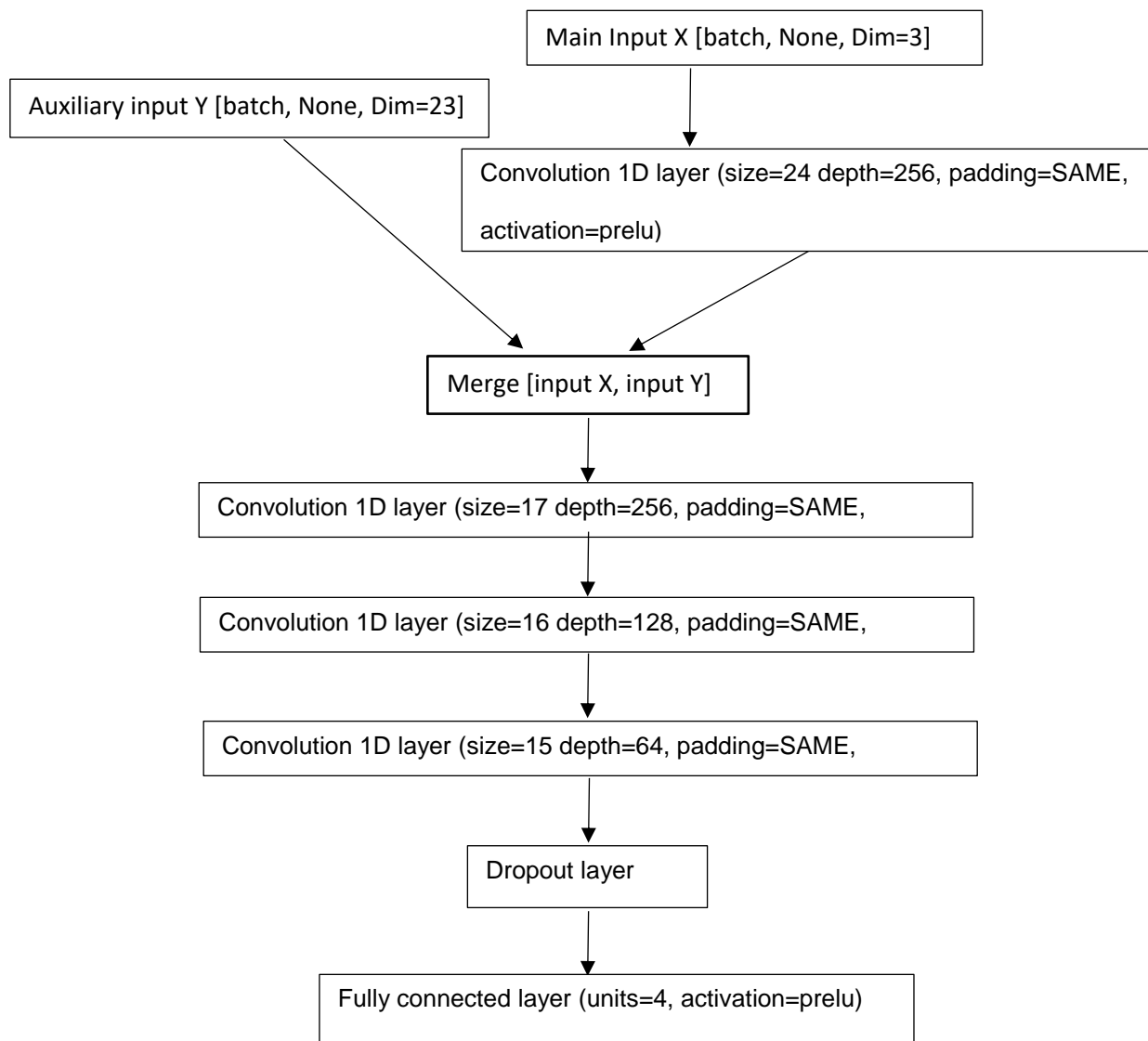
In this method we use our previous best discovered model (model three with five layers) to build same model for protein primary structure and water solvent accessibility property in an asymmetric tree, and we connect both models before fully connected layer using merge layer. Moreover, merge layer has three different ways to connect the inputs first add or sum the inputs, second concatenate the inputs, and last multiply the inputs. We try all three ways of connection with same data to find out which one gives the best result. Just like the before models we pad the sequence by zero to the longest sequence in each batch after we sort the data from shortest sequence to the longest. Therefore, some of the sequence been padded just few zeros in the end of each batch so we can train it in the model. That is mean, each sequence differ in length from batch to batch but have same size in each batch. When training the network, each training protein sequence was fed as the input (x) of the network, and each training water solvent accessibility property was fed as the input (y) of the network, along with the classification target (z) of that sequence encoded as a 0-1 vector. Each class was represented as a coordinate in the target. The input sequences were encoded as an arrays of 3 dimensional inputSeq with dimensions [batch size, none, dims]. Batch size means the number of sequences in a mini batch and was set to 10 or 100. None here mean that the size entered the model is different in size the maximum length of a sequence in each batch: sequences shorter than this were padded to the max length amino acids in the data. We encoded each amino acid as a 23-dimensional vector, where the first 22 components comprised as the amino acid sequence and the last one represent the zero padding. Moreover, we encoded

each the water solvent accessibility property as a 3-dimensional vector, where the first 2 components comprised as the water solvent accessible or not and the last one represent the zero padding. 4 classes for target if it is SSP 3 for the structure and 1 for zero padding, and 9 classes for target if it is DSSP 8 for the structure and 1 for zero padding.

The deep neural network starts with main input (X) which is the protein sequence flowed by a convolutional architecture with 1D then the second input (Y) which is the water solvent property is connected to the main input by share layer. After sharing layer 3 convolution layers are added then fully connected layers at the end. In this method we did not add any flatten layer because this layer need fixed size and we have varied size. Moreover, we did not add any max pooling layers because this layer reduce the size and this we do not want in our model. This is the map of the sharing layer.

The full architecture of is shown in Table 4.4. The network has one 1dimensional convolution layers with kernel sizes [24] which represents the amino acid category and depths (filter counts) [256] then 3 one-dimensional convolution layers with kernel sizes [17, 16, 15] and depths (filter counts) [256, 128, 64], with PReLU (parametric rectified linear unit) activation after the share layer.

**Table 4.4:** The architecture of best model of protein structure dictation using asymmetric tree.



After we find the best model, we try four different merge layer and here it is there output architecture. First, merge layers using add, this layer sum all the inputs come from above layers together of cores they must have same neurons number

Layer (type)	Output Shape	parameter#
Connected to		
input_2 (Input Layer)	(None, None, 3)	0
conv1d_1 (Conv1D)	(None, None, 24)	1248
input_2 [0] [0]		

dropout_1 (Dropout)	(None, None, 24)	0
conv1d_1 [0] [0]		
input_1 (Input Layer)	(None, None, 24)	0
add_1 (Add)	(None, None, 24)	0
dropout_1 [0] [0]		
input_1 [0] [0]		
conv1d_2 (Conv1D)	(None, None, 256)	98560
add_1 [0] [0]		
conv1d_3 (Conv1D)	(None, None, 128)	524416
conv1d_2 [0] [0]		
conv1d_4 (Conv1D)	(None, None, 64)	122944
conv1d_3 [0] [0]		
dropout_2 (Dropout)	(None, None, 64)	0
conv1d_4 [0] [0]		
Main output (Dense)	(None, None, 9)	585
dropout_2 [0] [0]		
=====		
Total parameters: 747,753		
Trainable parameters: 747,753		
Non-trainable parameters: 0		

Second, merge layers using concatenate, this layer it put all the inputs come from above layers beside each other of cores they must have same neurons number and this make the output of this layer doubled in size.

Layer (type)	Output Shape	parameter#
Connected to		
=====		
input_2 (Input Layer)	(None, None, 3)	0
conv1d_1 (Conv1D)	(None, None, 24)	1248
input_2 [0] [0]		
dropout_1 (Dropout)	(None, None, 24)	0
conv1d_1 [0] [0]		
input_1 (Input Layer)	(None, None, 24)	0



```

concatenate_1 (Concatenate)      (None, None, 48)      0
dropout_1 [0] [0]

input_1 [0] [0]

-----
conv1d_2 (Conv1D)                (None, None, 256)     98560
concatenate_1 [0] [0]

-----
conv1d_3 (Conv1D)                (None, None, 128)     524416
conv1d_2 [0] [0]

-----
conv1d_4 (Conv1D)                (None, None, 64)      122944
conv1d_3 [0] [0]

-----
dropout_2 (Dropout)             (None, None, 64)      0
conv1d_4 [0] [0]

-----
Main output (Dense)             (None, None, 9)       585
dropout_2 [0] [0]
=====
Total parameters: 747,753
Trainable parameters: 747,753
Non-trainable parameters: 0

```

Third , merge layers using multiply, this layer multiply all the inputs come from above layers together of cores they must have same neurons number. As architecture it don't has any different between add architecture for that no need to repeat it. Last one, merge layers using subtract, this layer subtract all the inputs come from above layers together of cores they must have same neurons number. As architecture it don't has any different between add architecture for that no need to repeat it In this way we find out that using add method in merge layer give us the best result fallowed by concatenate then subtract and last multiply.

We check these methods we discover add method gives the best results without adding more information to the model. After building symmetric tree and asymmetric tree models we discover that the symmetric model gives better results.

## CHAPTER FIVE: REQUIREMENTS AND TOOLS

### Software Requirements

There are several programming languages for machine learning the most popular are R language and Python. We build our project using Python programming language because it is easy to install, implement, and it has a lot of support and libraries. It would be a surprise if Python can take the data analysis mantle from R, but matrix handling in NumPy may challenge MATLAB and communication tools like IPython are very attractive and a step into the future of reproducibility; SciPy stack for machine learning and data analysis can be used for projects, and frameworks like scikit-learn are mature enough to be used in production systems. Therefore, using Python makes work much easier. Here is a list of software and hardware requirements:

- python 2.7
- Keras
- Tensorflow
- CUDA 8
- Opencv 3

### Hardware Requirements

- RAM bigger better
- GPU

The entire training process has been carried out using the following two system

### **First system**

- Architecture: x86\_64
- Processor: Intel® Xeon® CPU E5-2680, 8-Core w/HT (16 Virtual Cores), 2.7 GHz (3.5 GHz Turbo Boost), 20M Cache, 130w.
- Operating System: Linux 16.04
- Memory: 16GB
- Hard Drive: SSD 600GB, 2.5 SATA-300
- Graphic Card: GF GTX 1070, 8GB, PCIe 3.0 x 16

### **Second system**

- Architecture: x86\_64
- Processor: Intel® Xeon® CPU E5-2680, 8-Core w/HT (16 Virtual Cores), 2.7 GHz (3.5 GHz Turbo Boost), 20M Cache, 130w.
- Operating System: Linux 16.04
- Memory: 16GB
- Hard Drive: SSD 600GB, 2.5 SATA-300
- Graphic Card: TITAN X(Pascal)/PCIe/SSE2

## CHAPTER SIX: RESULTS AND DISCUSSION

The performance of the networks we built is discussed in details below:

### A/ Protein structure dictation using just the primary structure

#### First Method: Protein structure dictation with padding

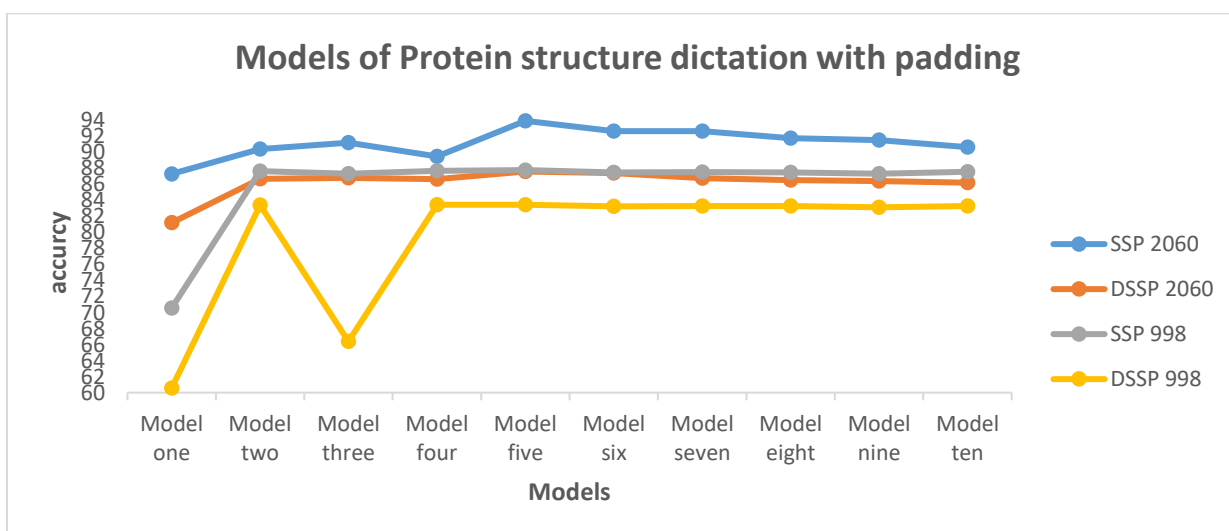
In this method we pad each sequence by zero to the longest sequence we have in the data which is 2060. As we discuss before this method is effected so much with padding; so we made another data which we deleted the long sequences to make the longest sequence we have 998 long; therefor, our model being pad to 998 .However, we build several models ten models to be exact; starting from three layer model one convolution layer fallowed by max pooling and in the last fully connected layer. After that we add two layers one convolution layer fallowed by max pooling with every model but in some models we add fully connected layer too; to find out which model gives the best results. The results of models is shown in table 6.1.

**Table 6.1:** The result of models of protein structure dictation with padding method

Model	Number of layers	Full Padding(2060)		Less Padding(998)	
		SSP	DSSP	SSP	DSSP
Model one	Three layers	87.3	81.2	70.56	60.58
Model two	Five layers	90.4	86.7	87.64	83.43
Model three	Seven layers	91.2	86.79	87.32	66.40
Model four	Nine layers	89.5	86.64	87.67	83.45
Model five	Eleven layers	93.92	87.6	87.78	83.45

Model six	Thirteen layers	92.64	87.4	87.49	83.27
Model seven	fifteen layers	92.64	86.75	87.52	83.29
Model eight	Seventeen layers	91.74	86.53	87.49	83.31
Model nine	Nineteen layers	91.51	86.4	87.033	83.15
Model ten	Twenty one layers	90.64	86.2	87.55	83.29

As we see from above table since model one with three layers the accuracy get race up until model five with eleven layers gives the best results in both situation. It gives 93.92% for SSP data and 87.6% for DSSP data where the sequences are padded to 2060 long. Same model gives the best result when the sequence is padded to 998 long after deleting the long sequences. It gives 87.78 for SSP data and 83.45 for DSSP data. After model five the accuracy drop out rapidly until we stop in model ten because the accuracy did not race up after word. Below is the chart 6.1 shows up the models and their accuracy in both situation with full padding or reduced padding.



**Chart 6.1:** This chart shows the differences between models accuracy of protein structure dictation with padding

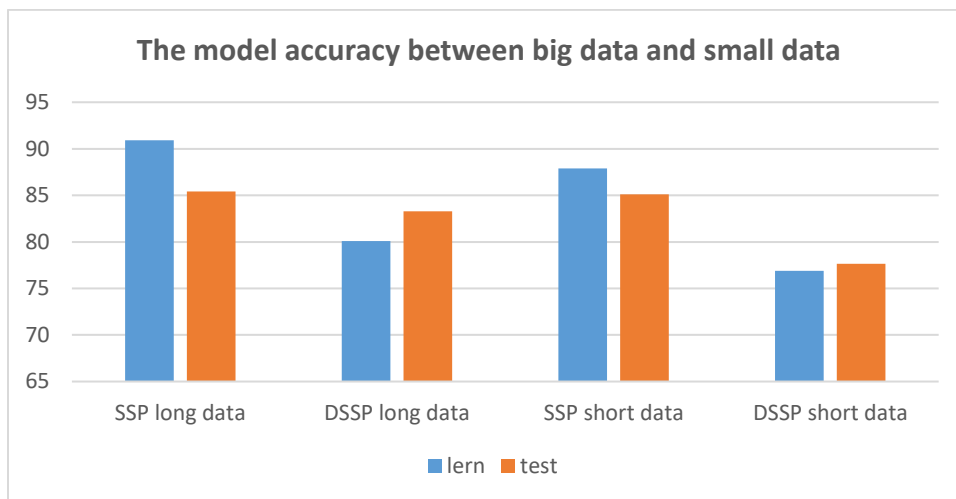
## Second Method: Protein structure dictation without padding

In this method we just pad the sequence by zero to the longest sequence in each batch after we sort the data from shortest sequence to the longest. Therefore, some of the sequence been padded just few zeros in the end of each batch so we can train it in the model, and in the last batch some sequence been padded almost 100 zero. Moreover, we build several models ten models to be exact; starting from three layer model one convolution layer fallowed fully connected layer. After that we add layers of convolution in every model but in some models we add fully connected layer too; to find out which model gives the best results. The results of models is shown in table 6.2.

**Table 6.2:** The result of Models of protein structure dictation without padding method

Model	Number of layers	SSP	DSSP
Model one	Three layers	85.99	74.4
Model two	four layers	85.17	76.1
Model three	five layers	90.93	80.1
Model four	eight layers	86.33	79.3
Model five	nine layers	82.87	79.5
Model six	Thirteen layers	81.09	78.5
Model seven	fourteen layers	79.80	77.6
Model eight	fifteen layers	84.1	76.8
Model nine	seventeen layers	88.2	75.3
Model ten	Twenty one layers	88.1	75.6

As we see from above table since model one with three layers the accuracy get race up until model three with five layers gives the best results in both situation. It gives 90.93% for SSP data and 80.1% for DSSP data. Our model gives better result by 1% from Must\_CNN witch gives 89.6 for SSP, and our model gives better result by 3% comparing to Must\_CNN witch gives 76.7% for DSSP. After model five the accuracy drop out rapidly until we stop in model ten because the accuracy did not race up after word. The above results were tested on big data (4protein). There is small data (Cb 513 CullPDB); this data is very important for checking the performance of any model because the CullPDB dataset has sequences with  $> 25\%$  identity with the CB513 dataset was removed; so we try our model on small data, and it gives 87.88% for SSP and 85.1% for testing. Blow is the chart 6.2 that show the result



**Chart 6.2:** This chart shows the differences between the model accuracy between big data and small data

	precision	recall	f1-score	support
DSSP , SAA				
H	0.87	0.92	0.92	10
E	0.83	0.91	0.86	81
L	0.72	0.83	0.69	60
T	0.51	0.63	0.59	18
S	0.24	0.65	0.34	29
G	0.3	0.72	0.4	1
B	0.39	0.61	0.08	11
I	0	0.01	0	10
SSP, SAA				
C	0.88	0.88	0.85	1
H	0.82	0.9	0.8	20
E	0.86	0.2	0.87	0

**Table 6.3:** accuracy table for Protein structure dictation without padding

## **B/ Protein structure dictation using primary structure and one protein property**

### **First Method: Protein structure dictation by merging primary structure and one protein property using symmetric tree**

In this method we use our previous best discovered model (model three with five layers) to build same model for protein primary structure and water solvent accessibility property in symmetric tree, and we connect both models before fully connected layer using merge layer. We build several merge models five to be exact, we add convolution layer before or after merge layer to find out which model gives the best results; we try our models on DSSP data merge to SAA Absolute solvent accessibility data. The results of models is shown in table 6.3.

**Table 6.4:** The result of models of Protein structure dictation using symmetric tree method

Model	Number of layers	DSSP, SAA
Model one	3 convolution 1D layer before Merge layer, No convolution layer after	82.2



Model two	2 convolution 1D layer before Merge layer, 1 convolution layer after	80.58
Model three	1 convolution 1D layer before Merge layer, 2 convolution layer after	81.43
Model four	1 convolution 1D layer before Merge layer, 3 convolution layer after	80.34
Model five	3 convolution 1D layer before Merge layer, 3 convolution layer after	79.27

As we see from above table since model one with 3 convolution 1D layer before Merge layer, and no convolution layer after the merge layer gives the best results. After we find the best model, we try three different merge layer add, concatenate, and multiply to find out which architecture has best result. We try all architectures with same data DSSP for primary structure and SAR for protein property and here it is the results in table 6.4.

**Table 6.5:** The result of different ways merge layer in symmetric tree

Architecture	Result using DSSP ,SAR
Add	81.07
Concatenate	80.15
Subtract	79.92
Multiply	78.80

As we see from the table above the add architecture has the best result 81.07 followed by concatenate 80.15, and the worst is multiply. After we find the best architecture, we try this architecture on SSP and both protein property SAA and SAR to find out which gives the best result here is the output in table 6.5.

**Table 6.6:** The result of best model of protein structure dictation using symmetric Tree method

amino acid structure/ water solvent accessible	SAA	SAR
SSP	91.6	89.2
DSSP	82.2	81.07

As we see from above table our model gives the best result when we merge it with SAA data in both cases SSP and DSSP. When we merge SSP with SAA we get 91.6 which is better than just try predict protein secondary structure with just primary structure almost by 1%, and same for DSSP it get 82.2% which is better by almost 2%. While, when we merge SSP or DSSP data with SAR it gives 89.2% for SSP which is less than trying amino acid alone, and gives 80.6 for DSSP which is almost the same result when we try alone with primary structure. The above results were tested on big data (4protein). There is small data (Cb 513 CullPDB); this data is very important for checking the performance of any model because the CullPDB dataset has sequences with  $> 25\%$  identity with the CB513 dataset was removed; so we try our model on small data and the table 6.6 has the result.

**Table 6.6:** The result of small data of protein structure dictation using symmetric tree method

amino acid structure/ water solvent accessible	SAA	SAR
SSP	86.73	83.41
DSSP	79.86	79.21

We try another merge method using asymmetric tree layer.

	precision	recall	f1-score	support
DSSP , SAA				
H	0.81	0.92	0.92	10
E	0.81	0.91	0.86	81
L	0.72	0.82	0.69	60
T	0.51	0.65	0.59	18
S	0.2	0.65	0.34	29
G	0.3	0.72	0.48	1
B	0.39	0.61	0.08	11
I	0	0	0	10
SSP, SAA				
C	0.8	0.88	0.85	1
H	0.89	0.94	0.8	20
E	0.86	0.8	0.87	0

**Table 6.7:** The accuracy table for symmetric tree

## **Second Method: Protein structure dictation by merging primary structure and one protein property using asymmetric tree**

In this method we use Merge layer to connect protein data and the water solvent accessibility property data. The deep neural network starts with main input (X) which is the protein sequence, followed by a convolutional architecture with 1D. Then the second input (Y) which is the water solvent property is connected to the main input by merge layer. After merge layer, three convolution layers are added, then fully connected layers at the end. Moreover, we try different merge layers: add, concatenate, subtract, and multiply to find out which architecture has the best result. We try all architectures with the same data: DSSP for primary structure and SAR for protein property, and here are the results in table 6.7.

**Table 6.8:** The result of different ways merge layer in asymmetric tree

Architecture	Result using DSSP ,SAR
Add	79.27
Concatenate	77.22
Subtract	72.43
Multiply	68.83

As we see from the table above the add architecture has the best result 79.27 followed by concatenate 77.22, then subtract 72.43 and the worst is multiply.

We try add model on SSP and another data SAR Relative solvent accessibility which gives the most solvent Accessible here is the results in table 6.8.

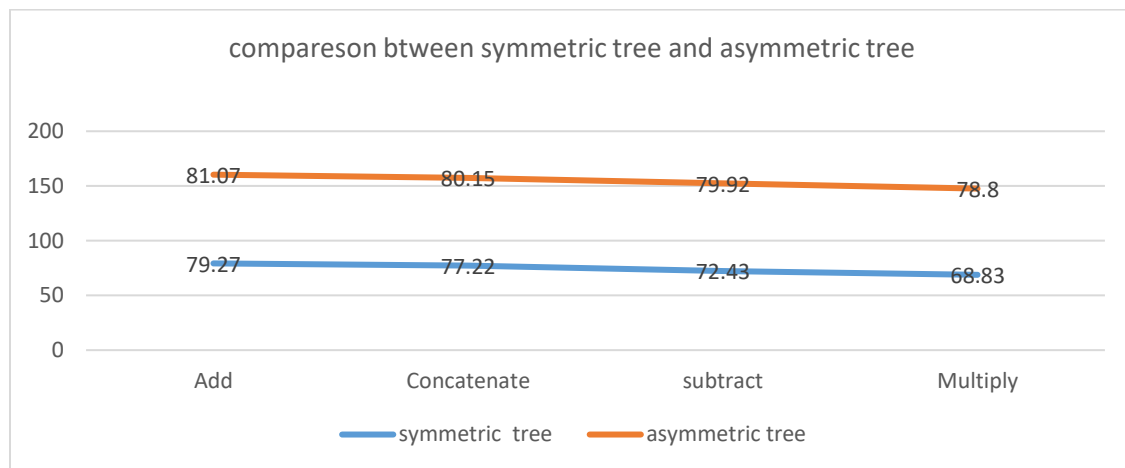
**Table 6.9:** The result of best model of protein structure dictation using asymmetric tree method

amino acid structure/ water solvent accessible	SAA	SAR
SSP	91.2	90.6
DSSP	80.8	79.27

As we see from above table our model gives the best result when we merge it with SAA data in both cases SSP and DSSP. When we merge SSP with SAA we get 91.2% which is better than just try predict protein secondary structure with just amino acid almost by 1%, and same for DSSP it get 80.8% which is almost same. While, when we merge SSP

or DSSP data with SAR it gives 90.6% for SSP which is same as trying amino acid alone, and gives 79.08% for DSSP which is less result when we try alone with amino acid.

When we compare symmetric tree with asymmetric tree we almost get same results, but most time better in symmetric tree. However, defiantly adding more information to amino acid to predict protein structure gives better results. The flowing chart 6.3 show the compaction between merge and sharing layers result



**Chart 6.3:** This chart shows the accuracy difference between symmetric tree and asymmetric tree

## CHAPTER SEVEN: SUMMERY AND FUTURE WORK

### Summery

There are limitation on secondary structure prediction. The unstable definition of three states. Where Ideal helices and sheets do not apply, and there are no clear boundaries between helix and coil nor sheet and coil states (Yang, Gao, Wang, Heffernan, Hanson, Paliwal, & Zhou, 2016). This assignment inconsistency would drag the highest possible accuracy to closely 88–90% . While secondary structure prediction has reached an accuracy (about 89%) in Must\_CNN that is close to the theoretical limit (88%). Can we make the break this long-standing challenge? Recent accuracy improvements have resulted from constantly larger sequence and structural databases, greater fancy deep learning neural networks (Heffernan, Paliwal & Lyons, 2015; Wang, Li, Liu, & Xu, 2016) As the number of protein sequences, along with the number of solved structures, continues to develop exponentially, the secondary structure prediction accuracy is probably to extend its incremental improvement.

Deep learning methods are a powerful complement to classical machine learning tools and other cut and try strategies. Already, these approaches have found manage in an abode of applications in computational science of matter, including controlling genomics and image analysis. Convolutional neural networks are efficient of learning sophisticated features, anyway their performance does not increase monotonically mutually their complexity. We have constructed deep convolution neural networks for protein inconsequential structure illusion from its dominant structure. By comparing previous function, our neural networks outperformed the existing solutions and have attained a better

result 1% higher accuracy comparatively by using primary structure as input, and about 2% better accuracy by adding one information of protein property.

We have tried several methods to find which method gives best result, and in each method we try add more layers to the convolution neural network to find which number of layers gives the best result. The first method we build was padding the protein sequence to the longest sequence we have. However this method was effected so much by the padding, so we try the second method. The second method we did not pad the sequences instead we categorize the amino acids and the target structure. This method gives very good results, about 1% better than other methods used by other paper. The last method we have described a multi\_input convolutional architecture for sequence prediction. We use ideas from the image classification domain to train a deep convolutional network on per-position sequence labeling. In this method we add protein property to the method beside the primary structure of amino acid. More over this method give better result by 2%.

As conclusion, using five layers of convulsion neural network with only input the primary structure of the protein gives accuracy 90.93% for SSP and 80.1% for DSSP. Moreover, using multi\_input convulsion neural network improve the accuracy more; adding protein property with primary structure gives 91.6% for SSP accuracy and 82.2% for DSSP.

## **Future work**

Our project can be develop two ways. First way is by adding Long Short Term Memory (LSTM) it known as Recurrent Neural Network (RNN) layer with convulsion neural network. Recurrent nets are a type of artificial neural network designed to recognize patterns in sequences of data, such as text, genomes, handwriting, the spoken word, or numerical time series data emanating from sensors, stock markets and government agencies. They are arguably the most powerful and useful type of neural network, applicable even to images, which can be decomposed into a series of patches and treated as a sequence. Moreover, we can consider embedding layer too. Embedding layer transfer words in a sentence to vectors, so if think about each protein sequence to be sentence and each amino acid to be a word then this layer may be useful. Second way to improve our work is to add more protein properties like adding amino acid charge information, or polarity, etc. Moreover, collecting more data will have great impact on learning in machine learning.



## REFERENCES

- Aftabuddin, M., & Kundu, S. (2007). Hydrophobic, Hydrophilic, and Charged Amino Acid Networks within Protein. *Biophysical Journal*, Volume 93(2007), 225-231. doi:10.1529/biophysj.106.098004
- Andreeva, A., Howorth, D., Chandonia, J.-M., Brenner, S. E., Hubbard, T. J. P., Chothia, C., & Murzin, A. G. (2008). Data growth and its impact on the SCOP database: new developments. *Nucleic Acids Research*, 36(Database issue), D419–D425. Retrieved from: <http://doi.org/10.1093/nar/gkm993>.
- Angermueller, C., Pärnamaa, T., Parts, L., & Stegle, O. (2016). Deep learning for computational biology. *Molecular Systems Biology*, volume, 878th ser. doi:10.15252/msb.20156651.
- Accessible surface area*. (2017 , September 19). Retrieved from "https://en.wikipedia.org/w/index.php?title=Accessible\_surface\_area&oldid=80136525"2 Wikipedia
- Angermueller, C., Pärnamaa, T., Parts, L., & Stegle, O. (2016, July, 29). Deep learning for computational biology. *Molecular Systems Biology*, Volume 12, Issue 7, 878.doi:10.15252/msb.20156651.

- Ballester, P. J., & Mitchell, J. B. O. (2010). A machine learning approach to predicting protein-ligand binding affinity with applications to molecular docking. *Bioinformatics (Oxford, England)*, 26(9), 1169–1175. Retrieved from: <http://doi.org/10.1093/bioinformatics/btq112>.
- Benson, D. A., Clark, K., Karsch-Mizrachi, I., Lipman, D. J., Ostell, J., & Sayers, E. W. (2015). GenBank. *Nucleic Acids Research*, 43(Database issue), D30–D35. Retrieved from: <http://doi.org/10.1093/nar/gku1216>.
- Berg J., Tymoczko J., Stryer L. (2002). Chapter 3, Protein Structure and Function, *Biochemistry*. 5th edition. New York: W H Freeman. Available from: <https://www.ncbi.nlm.nih.gov/books/NBK21177/>
- Bhattacharya, D., Cao, R., & Cheng, J. (2016). UniCon3D: de novo protein structure prediction using united-residue conformational search via stepwise, probabilistic sampling. *Bioinformatics*, volume 32(18), 2791–2799. Retrieved from: <http://doi.org/10.1093/bioinformatics/btw316>.
- Bodenhofer, U., Bonatesta, E., Horejs-Kainrath, C., & Hochreiter, S. (2015). Msa: an R package for multiple sequence alignment. *Bioinformatics*, 1(2015), 3rd ser. doi:10.1093/bioinformatics/btv494

- Collobert, R., and Weston, J. (2008). *A unified architecture for natural language processing: Deep neural networks with multitask learning*. In Proceedings of the 25th international conference on Machine learning, Helsinki, Finland.
- CS231n Convolutional Neural Networks for Visual Recognition. (n.d.). Retrieved from file:///D:/writing%20thesis/CS231n%20Convolutional%20Neural%20Networks%20for%20Visual%20Recognition.html, karpathy@cs.stanford.edu
- Dor, O., Zhou, Y., (2007) Achieving 80% ten-fold cross-validated accuracy for secondary structure prediction by large-scale training. *Proteins*, volume 66, Issue 4, Pages 838–845. doi: 10.1002/prot.21298.
- Dougherty, D. A. (2007). Cation- $\pi$  Interactions Involving Aromatic Amino Acids. *The Journal of Nutrition*. Retrieved from <http://jn.nutrition.org/content/suppl/2007/05/18/137.6.1504S>
- Faraggi, E., Zhang, T., Yang, Y., Kurgan, L., & Zhou, Y. (2012). SPINE X: Improving protein secondary structure prediction by multi-step learning coupled with prediction of solvent accessible surface area and backbone torsion angles. *Journal of Computational Chemistry*, 33(3), 259–267. <http://doi.org/10.1002/jcc.21968>
- Fischer, D., & Eisenberg, D. (1996). Protein fold recognition using sequence-derived predictions. *Protein Science : A Publication of the Protein Society*, 5(5), 947–955.

Golkov, V., Skwark, M., Golkov, A., Dosovitsky, A., Brox, T., Meiler, J., & Cremers,

D. (2016). *Protein contact prediction from amino acid co-evolution using convolutional networks for graph-valued images*. Paper presented at 30th Conference on Neural Information Processing Systems, Barcelona, Spain.

Heffernan, R., Paliwal, K., Lyons, J., (2015). Improving prediction of secondary structure, local backbone angles, and solvent accessible surface area of proteins by iterative deep learning. *Scientific Reports* 5, Article number: 11476 (2015) doi:10.1038/srep11476

Heffernan, R., Paliwal, K., Lyons, J., Dehzangi, A., Sharma, A., & Wang, J., Zhou, Y. (2015). Improving prediction of secondary structure, local backbone angles, and solvent accessible surface area of proteins by iterative deep learning. *Scientific Reports*. Volume 5, Article number: 11476 (2015). doi:10.1038/srep11476

Jami, M. S., Husain, I. A., Kabashi, N. A., & Abdullah, N. (2011). Multiple Inputs Artificial Neural Network Model for the Prediction of Wastewater Treatment Plant Performance. *Australian Journal of Basic and Applied Sciences*. Volume 6(1): 62-69, 2012 doi:10.2316/P.2011.736-050

Jones, D., (1999) Protein secondary structure prediction based on position-specific scoring matrices, *Journal of Molecular Biology*, volume 292, Issue 2 195–202. volume 292, Issue 2, Pages 195-202. Retrieved from <https://doi.org/10.1006/jmbi.1999.3091>.

Jordan, M., Kleinberg, J., & Scholkopf, B. (2006). *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer Science+Business Media, ISBN: 0-387-310738.

Kabsch, W., Sander, C., (1983) Dictionary of protein structure: pattern recognition of Hydrogen-bonded and geometrical features. *Biopolymers*, 1983, 22:2577–637. doi: 10.1002/bip.360221211.

Kim, Y. (2014) *Convolutional Neural Networks for Sentence Classification*. Retrieved from Cornell university library website: <https://arxiv.org/abs/1408.5882>.

Kuefler, A. R. (2016). *Merging Recurrence and Inception-Like Convolution for Sentiment Analysis*. Retrieved from CS224d: Deep Learning for Natural Language Processing website. <https://cs224d.stanford.edu/reports/akuefler.pdf>

Le, Q. V. (2015). *A Tutorial on Deep Learning Part 2: Autoencoders, Convolutional Neural Networks and Recurrent Neural Networks*. October 20, 2015, retrieved from Google Brain website: <http://www.ai.stanford.edu/~quocle/tutorial2.pdf>

- Li, Z., & Yu, Y. (2016). *Protein Secondary Structure Prediction Using Cascaded Convolutional and Recurrent Neural Networks*. April 25, 2016. Retrieved from Cornell university library website: <https://arxiv.org/abs/1604.07176>.
- Lin, Z., Lanchantin, J., & Qi, Y.(2016, February). *MUST-CNN: A Multilayer Shift-and-Stitch Deep Convolutional Architecture for Sequence-based Protein Structure Prediction*. Paper presented at AAAI'16 Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, Phoenix, Arizona, February 12 - 17, 2016.
- Magnan, C. N., & Baldi, P. (2014). SSpro/ACCpro 5: almost perfect prediction of protein secondary structure and relative solvent accessibility using profiles, machine learning and structural similarity. *Bioinformatics*, volume 30(18), 2592–2597. Retrieved from: <http://doi.org/10.1093/bioinformatics/btu352>.
- Michael, M., & Gromiha (2010) Protein Structure Analysis, *Protein Bioinformatics*, Retrieved from: <https://www.sciencedirect.com/topics/biochemistry-genetics-and-molecular-biology/accessible-surface-area>
- Nguyen, N. G., Tran, V. A., Ngo, D. L., Phan, D., Lumbanraja, F. R., Faisal, M. R., Satou, K. (2016). DNA Sequence Classification by Convolutional Neural Network. *Biomedical Science and Engineering*, volume 9(2016), 280-286. Retrieved from: <http://dx.doi.org/10.4236/jbise.2016.95021>.

Nowak, A., & Bruna, J. (2017). *Divide and Conquer Networks*. Retrieved from ARXIV website. <https://arxiv.org/abs/1611.02401>

Nowak, A., & Bruna, J. (2017, April). *Divide and Conquer with Neural Networks*. Paper presented in International Conference on Learning Representations Toulon, France. Place of publication: [https://openreview.net/forum?id=Hy3\\_KuYxg](https://openreview.net/forum?id=Hy3_KuYxg).

Ozkan, S., Wu, G., Chodera, J., (2007). Protein folding by zipping and assembly. *Proceedings of the national academy of science of United States of America PNAS*, 104:11987–92. July 17, 2007 vol. 104 no. 29 11987–11992 BI. Retrieved from: <http://www.pnas.org/content/104/29/11987.full>

Pan, X., & Shen, H. (2016). RNA-protein binding motifs mining with a new hybrid deep learning based cross-domain knowledge integration approach. *BMC Bioinformatics*, (2017) 18:136. DOI 10.1186/s12859-017-1561-8

Pinheiro, P. H. O., and Collobert, R. (2013). *Recurrent Convolutional Neural Networks for Scene Parsing*. Retrieved from Cornell university library website: <https://arxiv.org/abs/1306.2795>.

Plaxco, K., Simons, K., Baker, D., (1998) Contact order, transition state placement and the refolding rates of single domain proteins, *Journal of Molecular Biology*, volume 1998;277:985–94. DOI:10.1006/jmbi.1998.1645

Ramsey, D. C., Scherrer, M. P., Zhou, T., & Wilke, C. O. (2011). The Relationship Between Relative Solvent Accessibility and Evolutionary Rate in Protein Evolution. *Genetics*, volume 188(2), 479–488. Retrieved from: <http://doi.org/10.1534/genetics.111.128025>.

Rost, B., & Sander, C. (1993). Improved prediction of protein secondary structure by use of sequence profiles and neural networks. *Proceedings of the National Academy of Sciences of the United States of America*, 90(16), 7558–7562.

Sillitoe, I., Lewis, T. E., Cuff, A., Das, S., Ashford, P., Dawson, N. L., Orengo, C. A. (2015). CATH: comprehensive structural and functional annotations for genome sequences. *Nucleic Acids Research*, 43(Database issue), D376–D381. Retrieved from: <http://doi.org/10.1093/nar/gku947>.

Spencer, M., Eickholt, J., & Cheng, J. (2015). *A Deep Learning Network Approach to ab initio Protein Secondary Structure Prediction*. Paper presented at IEEE/ACM Trans Comput Biol Bioinform, 12(1), 103-112. doi:10.1109/TCBB.2014.2343960.

Spencer, M., Eickholt, J., & Cheng, J. (2015). *A Deep Learning Network Approach to ab initio Protein Secondary Structure Prediction*. Paper presented at IEEE/ACM Trans Comput Biol Bioinform, 12(1), 103-112. doi:10.1109/TCBB.2014.2343960.



Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014).

Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15, 1929-1958. Retrieved June, 2014.

Sun,Y., Zhu, L., Wang,G., Zhao,F.,(2017) .Multi-Input Convolutional Neural Network for Flower Grading, *Journal of Electrical and Computer Engineering*, vol. 2017, Article ID 9240407, 8 pages, 2017. doi:10.1155/2017/9240407.

Szalkai, B., & Grolmusz, V. (2017). *Near Perfect Protein Multi-Label Classification with Deep Neural Networks*. March 30, 2017. Retrieved from Cornell university library website: <https://arxiv.org/abs/1703.10663>.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2014). *Going Deeper with Convolutions*. Retrieved from Cornell university library website: <https://arxiv.org/abs/1409.4842>

Terwilliger, T. C., Stuart, D., & Yokoyama, S. (2009). Lessons from Structural Genomics. *Annual Review of Biophysics*, 38, 371–383. Retrieved from: <http://doi.org/10.1146/annurev.biophys.050708.133740>.

- Ullah, Abu Dayem, and Kathleen Steinhöfel. (2010, January). *A hybrid approach to protein folding problem integrating constraint programming with local search*. Paper presented at The Eighth Asia Pacific Bioinformatics Conference, Bangalore, India, Place of publication: [www.biomedcentral.com/1471-2105/11/S1/S39](http://www.biomedcentral.com/1471-2105/11/S1/S39).
- Wang, S., Li, W., Liu, S., & Xu, J. (2016). RaptorX-Property: a web server for protein structure property prediction. *Nucleic Acids Research*, 44(W1):W430-5. doi: 10.1093/nar/gkw306.
- Wang, S., Peng, J., Ma, J., & Xu, J. (2016). Protein Secondary Structure Prediction Using Deep Convolutional Neural Fields. *Scientific Reports*, 6, 18962. Retrieved from: <https://www.nature.com/articles/srep18962.pdf>.
- Wu, S., Skolnick, J., & Zhang, Y. (2007). *Ab initio* modeling of small proteins by iterative TASSER simulations. *BMC Biology*, 5, 17. <http://doi.org/10.1186/1741-7007-5-17>.
- Yang, Y., Gao, J., Wang, J., Heffernan, R., Hanson, J., Paliwal, K., & Zhou, Y. (2016). Sixty-five years of the long march in protein secondary structure prediction: the final stretch?. *Briefings in Bioinformatics*, 1-13. doi:10.1093/bib/bbw129.
- Zeng, H., Edwards, M. D., Liu, G., & Gifford, D. K. (2016). Convolutional neural network architectures for predicting DNA–protein binding. *Bioinformatics*, 32(12), i121–i127. <http://doi.org/10.1093/bioinformatics/btw255>

Zhang, Y., & Gladyshev, V. N. (2007). High content of proteins containing 21st and 22<sup>nd</sup> amino acids, selenocysteine and pyrrolysine, in a symbiotic deltaproteobacterium of gutless worm *Olavius algarvensis*. *Nucleic Acids Research*, 35(2007), 4952–4963. doi:10.1093/nar/gkm514.

Zhou, Y., Karplus, M., (1999) Interpreting the folding kinetics of helical proteins. *Nature* 401:400–3. Retrieved from:  
<https://www.nature.com/nature/journal/v401/n6751/full/401400a0.html>

Zvelebil, M., & Baum, J. O. (2008). *Understanding bioinformatics*. New York, NY: Garland science, Taylor& Francis Group. ISBN: 0-8153-4024-9

## VITAE

Name of Author: Shayan Ihsan Jalal

Address in USA: 601 Botts Ave,

Troy, AL,36081

Address in Iraq: 101 Chuar Qurna main street, Rania, AL Sulaymani

Kurdistan region, Iraq

Telephone Number: (334) 372-4686

### EDUCATION

***Master degree in computer science*** – Troy University, Troy, Alabama, 2017.

Major: Computer Network & Security

***Bachelor degree in computer science*** – Kirkuk University, Kirkuk, Iraq, 2007.

Major: General

### EXPERIENCE

- In 2009 and 2010 work as computer principle lecturer in oil engineering department /engineering college /Kirkuk university /Kirkuk/ Iraq.
- In 2010 and 2011 work as head of programming unit in Computer and Internet Center /Kirkuk university /Kirkuk/ Iraq.
- In 2011 until 2015 work as lecturer in computer department / faculty of basic education/ Raparin university/ Al Sulaymani/ Kurdistan region/ Iraq.

## AWARDS

- IT Essentials: PC Hardware and Software form Cisco/ Networking Academy / Kirkuk University/ Kirkuk/ Iraq in 2010.
- C++ programing language Kirkuk University/ Kirkuk/ Iraq in 2010.
- IC3 degree from Kirkuk University/ Kirkuk/ Iraq in 2010.
- Control database with visual basic from Kirkuk University/ Kirkuk/ Iraq in 2010.
- Computer Skills from Computer and Internet Center/ Kirkuk University / Kirkuk/ Iraq in 2009.