

Modern Science and Arts University

Faculty of Computer Science

Graduation Project

Prediction of adaptor proteins using PSSM profiles and
recurrent neural networks.

Under Supervision
of:

Dr. Ahmed Farouk

Submitted By:

Name: Ahmed Samir Ahmed Bakrey

ID: 152059

ABSTRACT

The primary role of adaptor proteins is to transport and transmitting a signal. There are often intracellular signaling proteins of modular design, each domain of which may have a specific binding affinity and interact by forming complexes with other molecules inside the cell. A large number of studies have discovered the adaptor proteins to be associated with various human illnesses. Therefore, the process of discovering the function of adaptor proteins is one of bioinformatics and computational biology's most critical challenges. Only a few computational biology studies have attempted to determine protein functions, and almost all of them used position-specific scoring matrix (PSSM) profiles in the process. But unfortunately, the neural networks failed to develop optimal processing sequence information. Our approach is based on deep learning using a recursive neural network architecture to avoid the problem of the neural networks had previously encountered. So, this research claims to have developed a new solution by including RNNs (recurrent neural networks) and PSSM (Position-Specific Scoring Matrix) techniques to solve this problem.

Seems to have worked previously with state-of-the-art strategies that had proven effective, our methodology makes significant improvements in all of the common measurement metrics. Our results depend on the six measurements for our models such as accuracy, sensitivity, loss, the area under the curve (AUC), Matthew's correlation coefficient (MCC), and specificity of both validation and training the model.

This study opens the path for Recurrent Neural Network (RNN) and Position-Specific Scoring Matrix (PSSM) structures to be employed in computational biology and bioinformatics. The research approach we promote for is applicable to scientists who aim to improve the prediction results of protein function prediction problems.

Table of Contents:

ABSTRACT	1
CHAPTER 1: INTRODUCTION.....	6
1.1. INTRODUCTION.....	7
1.2. OBJECTIVE	9
1.3. MOTIVATION	9
1.4. PROBLEM STATEMENT	10
1.5. THESIS LAYOUT	11
CHAPTER 2: BACKGROUND AND LITERATURE REVIEW	12
2.1. BACKGROUND INFORMATION.....	13
2.1.1 Cell.....	13
2.1.2 DNA	13
2.1.3 Protein	15
2.1.4 Background on the protein adaptor.....	18
2.1.5 Virtual Screening.....	20
2.1.6 Deep Learning in Bioinformatics.....	21
2.1.7 The significance and background of protein adaptor research	25
2.2 PREVIOUS WORK.....	26
2.2.1 Classification of signaling proteins based on molecular star graph descriptors using Machine Learning models.....	26
2.2.1.1 Strategy and Structure.....	26
2.2.1.2 Data.....	27
2.2.1.3 Method Evaluation.....	27
2.2.1.4 Results	28
2.2.2 Identification of Clathrin proteins by incorporating hyperparameter optimization in deep learning and PSSM profiles	29
2.2.2.1 Strategy and Structure.....	29
2.2.2.2 Data.....	31
2.2.2.3 Method Evaluation.....	32
2.2.2.4 Results	32
CHAPTER 3: MATERIAL AND METHODS.....	34
3.1. MATERIALS	35
3.1.1. Data	35
3.1.2. Tools	37
3.1.3. Environment.....	38
3.2.1. System architecture Overview.....	39
CHAPTER 4: SYSTEM IMPLEMENTATION	45
4.1. INTRODUCTION.....	46
4.2. SYSTEM DEVELOPMENT	46
4.2.1. Building an experimental model to obtain a final model.....	46
4.3. SYSTEM STRUCTURE	46
4.3.1. System Overview.....	47
4.3.1.1. Data preparation.....	47
4.3.1.2. Classification	53
4.4. SYSTEM RUNNING	58
4.4.1. Component A	58
CHAPTER 5: RESULTS AND EVALUATION.....	60
5.1 TESTING METHODOLOGY.....	61
5.1.1. Data preparation	61
5.1.2. Classification.....	62
5.2. RESULTS.....	63

5.2.1.	<i>Best Results Cases</i>	63
5.2.2.	<i>Worst Results Cases</i>	64
5.2.3.	<i>Limitations</i>	64
5.3.	EVALUATION.....	64
5.3.1.	<i>Accuracy Evaluation</i>	64
5.3.2.	<i>Time Performance</i>	73
CHAPTER 6: CONCLUSION AND FUTURE WORK		74
6.1.	CONCLUSION	75
6.2.	PROBLEM ISSUES.....	75
6.2.1.	<i>Technical issues:</i>	75
6.2.2.	<i>Scientific issues:</i>	75
6.3.	FUTURE WORK.....	76
LIST OF REFERENCES:		77

List of Figures:

FIGURE 1: ADAPTER PROTEIN FROM SKAP2 FAMILY WITH A 1U5E CODE FROM PROTEIN DATA BANK.	7
FIGURE 2: THE PHASES AND COSTS OF DRUG DEVELOPMENT [1][2].	11
FIGURE 3: CLARIFY THE SHAPE OF DNA, GENES, AND CHROMOSOMES.	13
FIGURE 4: CLARIFY THE SHAPE OF THE DNA.	14
FIGURE 5: ILLUSTRATES THE PROCESS OF TRANSCRIPTION AND TRANSLATION IN A CELL.....	14
FIGURE 6: THE PRIMARY STRUCTURE OF A PROTEIN IS REPRESENTED BY EACH SMALLER CIRCLE, WHICH IMPLIES AN AMINO ACID.	15
FIGURE 7: ILLUSTRATION OF THE RELATIONSHIP BETWEEN POLYPEPTIDES, PROTEINS, AND AMINO ACIDS.	16
FIGURE 8: THE FOUR LEVELS OF PROTEIN STRUCTURE.	17
FIGURE 9: AN EXAMPLE OF A FASTA FILE.....	18
FIGURE 10: PROTEIN STRUCTURE REPRESENTATION. (PDB-ID: 1A1E).....	18
FIGURE 11: THIS DIAGRAM SHOWS THE MEMBRANE PROTEINS.	20
FIGURE 12: VIRTUAL SCREENING.....	21
FIGURE 13: HANDWRITTEN DIGIT RECOGNITION LABELLED DATA EVERY HANDWRITTEN NUMBER IS GIVEN A LABEL THAT REPRESENTS ITS ACTUAL VALUE.....	21
FIGURE 14: THIS IS AN APPROXIMATE COUNT OF PUBLISHED DEEP LEARNING AND DEEP LEARNING BIOINFORMATICS PAPERS YEAR-TO-DATE.	22
FIGURE 15: OVERALL STRUCTURAL GROWTH OVER THE PAST YEAR, AS TRACKED BY PDB STATISTICS.	23
FIGURE 16: AS THE QUANTITY OF DATA INCREASES, LEARNING PERFORMANCE IMPROVES.....	24
FIGURE 17: GOOGLE'S INCREASE IN TPU SPEED ON RESNET50 TRAINING.	25
FIGURE 18: THE MOST FREQUENTLY USED SET OF LBS PREDICTION INDICATORS.	26
FIGURE 19: A METHODOLOGY FLOWCHART SHOWING THE CREATION OF MOLECULAR GRAPHS AND THE USE OF MACHINE LEARNING FOR CLASSIFYING PROTEINS.	27
FIGURE 20: ROC CURVES FOR MODEL.....	28
FIGURE 21: FEATURE SELECTION ACHIEVED.....	29
FIGURE 22: 2D IDENTIFICATION OF CLATHRIN PROTEINS BY INCORPORATING HYPERPARAMETER OPTIMIZATION IN DEEP LEARNING AND PSSM PROFILES.	31
FIGURE 23: THE AMINO ACID COMPOSITION OF THE DISULFIDE CHAIN AND NON-FREE DISULFIDE SEQUENCE.	33
FIGURE 24: QUERY FOR GETTING THE ADAPTOR PROTEINS.....	35
FIGURE 25: ILLUSTRATION OF PREPARING THE DATASET.....	36
FIGURE 26: VISUAL MOLECULAR DYNAMICS PROGRAM.	37
FIGURE 27:FEED-FORWARD NEURAL NETWORK VS RECURRENT NEURAL NETWORK.	40
FIGURE 28: RNN AND FEED-FORWARD NEURAL NETWORK ARCHITECTURE.....	41
FIGURE 29: ARCHITECTURAL TYPES OF RNN.	42
FIGURE 30: THIS ILLUSTRATES THE IDEA OF FORWARD AND BACKWARD PROPAGATION IN A FEED-FORWARD NETWORK.....	43
FIGURE 31: A VISUAL ILLUSTRATION OF AN RNN, WITH ITS THREE INPUT GATES.	44
FIGURE 32: UNIPROT WEBSITE.	47

FIGURE 33: ILLUSTRATION TO OBTAIN THE PROTEIN SEQUENCE.....	48
FIGURE 34: FASTA FORMAT FOR A0JN7.....	48
FIGURE 35: QUERY OF GETTING DATA.....	49
FIGURE 36: ILLUSTRATION OF BASIC LOCAL ALIGNMENT SEARCH TOOL.....	50
FIGURE 37: ILLUSTRATION OF PSSM.....	51
FIGURE 38: EQUATION OF PSSM.....	51
FIGURE 39: PSSM FILE CREATION USING PSI-BLAST.....	52
FIGURE 40: FEATURE EXTRACTION VIA CNN.....	53
FIGURE 41: GRU ARCHITECTURE.....	55
FIGURE 42: EQUATION OF RESET GATE.....	55
FIGURE 43: A NON-LINEAR ACTIVATION TANH FUNCTION.....	56
FIGURE 44: EQUATION OF GRU GATE UPDATE.....	56
FIGURE 45: GRU EQUATION.....	56
FIGURE 46: RNN MODEL ARCHITECTURE FOR PREDICTING PROTEIN ADAPTORS.....	57
FIGURE 47: A WORKING FLOW FOR OUR MODEL.....	58
FIGURE 48: ILLUSTRATION OF ONE OF THE INPUTS PSSM FILE FOR SPECIFIC PROTEIN.....	59
FIGURE 49: A SAMPLE OF AN ANNOTATED DATASET.....	62
FIGURE 50: A PART OF A SAMPLE FROM THE ANNOTATED DATASET.....	62
FIGURE 52: ILLUSTRATION OF THE BEST CASES RESULTS.....	63
FIGURE 53: ILLUSTRATION OF THE WORST CASES RESULTS.....	64
FIGURE 54: ILLUSTRATION OF MEASUREMENTS THAT WERE USED FOR OUR MODEL.....	65
FIGURE 55: ILLUSTRATION OF TRAINING AND VALIDATION LOSS.....	66
FIGURE 56: ILLUSTRATION OF TRAINING AND VALIDATION ACCURACY.....	66
FIGURE 57: ILLUSTRATION OF TRAINING AND VALIDATION SENSITIVITY.....	66
FIGURE 58: ILLUSTRATION OF TRAINING AND VALIDATION MCC.....	67
FIGURE 59: ILLUSTRATION OF TRAINING AND VALIDATION SPECIFICITY.....	67
FIGURE 60: ILLUSTRATION OF TRAINING AND VALIDATION AREA UNDER THE CURVE.....	67
FIGURE 61:ILLUSTRATION OF TRAINING AND VALIDATION LOSS FOR THE SECOND MODEL.....	68
FIGURE 62: ILLUSTRATION OF TRAINING AND VALIDATION ACCURACY FOR SECOND MODEL.....	68
FIGURE 63: ILLUSTRATION OF TRAINING AND VALIDATION SENSITIVITY FOR SECOND MODEL.....	69
FIGURE 64: ILLUSTRATION OF TRAINING AND VALIDATION SPECIFICITY FOR SECOND MODEL.....	69
FIGURE 65: ILLUSTRATION OF TRAINING AND VALIDATION MCC FOR SECOND MODEL.....	69
FIGURE 66:ILLUSTRATION OF TRAINING AND VALIDATION AREA UNDER THE CURVE.....	70
FIGURE 67: COMPARISON OF OUR TWO RESULTS OF THE MODEL.....	70
FIGURE 68: COMPARISON OF PREVIOUS WORK WITH SENSITIVITY EVALUATION.....	71
FIGURE 69: COMPARISON OF PREVIOUS WORK WITH SPECIFICITY EVALUATION.....	71
FIGURE 70: COMPARISON OF PREVIOUS WORK WITH ACCURACY EVALUATION.....	72
FIGURE 71: COMPARISON OF PREVIOUS WORK WITH AREA UNDER THE CURVE EVALUATION.....	72
FIGURE 72: COMPARISON OF PREVIOUS WORK WITH MATHEW’S CORRELATION COEFFICIENT EVALUATION.....	72

CHAPTER 1: INTRODUCTION

1.1. Introduction

Protein function prediction is the process of finding out a protein's structural or biochemical characteristics by working backwards by sequencing the genome. Now that protein structure has captured the attention of scientists, other important features, like predictive performance, are also growing rapidly. Many different approaches have been proposed in the last few decades for this goal. The best two ways to select features are to either focus on powerful features or to have a strong machine learning approach.

Previous researchers have revealed that strong feature sets can have the desired effect, for instance, biochemical properties such as Amino acid index database (AAindex), position-specific scoring matrix (PSSM), is usually effective for obtaining satisfactory prediction results.

With the recent development of deep learning, a number of researchers have focused on attempts to use bioinformatics research to assist in protein function prediction. It has recently been demonstrated [4][5] that some of the works, as these, have been successful. We're motivated by these two facts, and use them to find a better technique for protein function prediction based on strong features and deep learning. In this work, we focused extra effort on the prediction of the adaptor protein, which is one of the most important players in signal transduction and there is a figure for displaying the illustration of adaptor protein using visual molecular dynamics program as shown in Figure 1: Adapter protein from SKAP2 family with a 1u5e code from protein data bank..

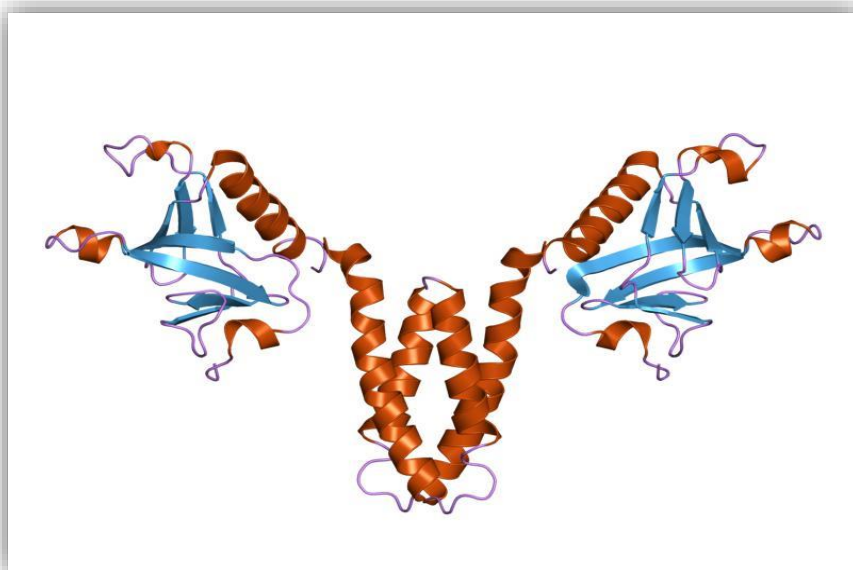


Figure 1: Adapter protein from SKAP2 family with a 1u5e code from protein data bank.

Pathway, commonly known as signal transduction, occurs when a cell signal moves from the outside to the membrane to the inner membrane. Signals must be brought to cells to be useful as a system for communicating decisions. The development of this pattern follows a progression of cell-surface receptors. One of the main goals of researchers who use signal transduction research is to find out the elements that make the pathways communicate with each other. A rising class of proteins that are much important in signal transduction are adaptors. The adaptor proteins in this dataset can be grouped into numerous functional modules that hold protein-binding partners together. Additionally, they are able to help to facilitate the formation of the signaling complexes [6]. They have a role in intercellular interactions and have a major influence on signal transduction mechanisms committed by their surface receptors.

specifically, many diseases have been shown to be associated with the adaptor proteins. as hematologic targets, for instance, Gab adaptor proteins have therapeutic applications in disease states of malformations. The XB130 protein is implicated in the causation of cancer **Error! Reference source not found.** Also, SLAP proteins (SLAP-1 and SLAP-2) play important roles in the development of osteoporosis, as well as numerous cancers and other malignancies [8]. For use in the treatment of chronic kidney disease, ADP has also been found to be a therapeutic target [9]. Additionally, a review paper from shows that adapter proteins are involved in the regulation of heart diseases. Furthermore, the involvement of adaptor protein complex 4 in virulent bacteria-induced hypersensitive cell death is proposed [10].

The study of signal transduction and adaptor proteins are critically important for their respective functions and structures, and recent progress has been made in elucidating the mechanisms. But it's both time-consuming and costly to try these experimental techniques. The method is of developing automatic prediction methods for adapter proteins is crucial in order to reduce the dependence on computational resourcefulness and ensure speed and accuracy.

One of the best sequence-protein annotations sets to differentiate evolutionary information in biology is PSSM. Much research has been done on protein structure prediction methods by using PSSM profiles such as fold recognition, phosphoglycation, succinylation, and subcellular localization prediction studies [11][12][13]. Although a variety of different methods have found an amino acid sequence solution, none has yet

found a solution to the problem of data loss in the PSSM profiles. In this thesis, we show an innovative method for overcoming the issue using a Recurrent Neural Network (RNN).

Neural networks, however, normally assume independent relationships between input signals, and this is seldom found in reality. Additionally, having the relevant genome sequence relationships aids in protein function prediction.

We present a novel deep learning pipeline which contains RNNs and PSSM profiles to identify adaptor proteins. Recurrent neural networks have shown various properties of protein sequences in the ability to extrapolate include extraction of sequential information. Still, how to apply the formula on PSSM profiles has not been fully determined.

1.2. Objective

The aim of this thesis is to develop a new method based on protein sequences that can be used to predict protein adaptors. This methodology will include a Site-specific Results Matrix (PSSM) as a feature. Moreover, this approach will utilize the architecture of a recurrent neural network (RNN) model and gated repetitive units (GRUs) to achieve a prediction of whether or not a protein is an adaptor by labeling it zero or one. Most importantly, the contributions of this thesis include:

1. Establishing the first sequence-based model for discriminating adaptors whether if it is an adaptor or general protein.
2. Designing an effective deep learning model for protein function prediction built by using RNN and PSSM profiles.
3. A benchmark dataset and recently discovered data is presented for adapter proteins.
4. Providing important information to researchers and biologists about the protein structure adaptors.

1.3. Motivation

Many protein-binding modules are found in the adaptor proteins, which are used to create larger signaling complexes. Through multiple protein-protein association, cellular signals can be propagated that ensure a corresponding response to the environment.

Signaling proteins are becoming more important in drug discovery due to the use of molecular targets to be pursued quickly, effectively, and at a low cost. The high structural complexity makes it difficult to associate the signaling activity with the structure. If an adaptor protein associates with its associated proteins, it can determine how the pathway is spread throughout the cell. It is an example. Which can be activated by numerous growth factors and signaling pathways, controls multiple biochemical pathways.

1.4. Problem Statement

Despite recent advancements, drug development is still very expensive and time-consuming. It currently costs \$2.6 billion and takes 13.5 years to bring a drug to market [1][2][3]. The two main phases of drug development are the preclinical phase and the clinical trials phase. So, in the preclinical phase to explore possible interactions with a target protein, millions of compounds are screened in high-throughput in vitro assays. Concentrations of compounds that we discovered in this preliminary screen, referred to as hits, are generally around micromolar (in the micromolar range) in affinity to the target. thus, the number of hits is subsequently turned into lead compounds, which have a higher affinity (in the nanomolar range). Leads are developed with properties resembling those of drugs, such as low toxicity, and to reduce off-target effects. Lastly, a leading is tested in animal models to see if it works. in clinical trials, compounds that have previously passed through the preclinical trial's stages are put to the test to see if they are beneficial in humans, starting with the smallest of the cohorts.

In the studies, it is often argued that the clinical trials phase is far more expensive than the preclinical phase, which has benefited from automation efforts massively. Actually, on a per drug basis, preclinical development accounted for only 7% of the total development cost. Due to the high attrition in drug development, especially in the preclinical phase, the preclinical phase accounts for 32% of the total cost of each successful drug. On top of that, preclinical development typically takes 5.5 years, which is 41% of the time it takes to complete development of each drug [1][2] as shows in *Figure 2: The phases and costs of drug development [1][2]..* Because of this, there remains a powerful need to speed up and reduce the cost of the preliminary studies. Computational methods are of great interest, as they enormously reduce the number of expensive and time-consuming wet laboratory experiments that are required.

	Preclinical				Clinical Trials			
	Target-to-hit	Hit-to-lead	Lead optimization	Preclinical	Phase I	Phase II	Phase III	Submission to launch
Cost per drug (millions)	\$1	\$2.5	\$10	\$5	\$15	\$40	\$150	\$40
	\$33.5 (7%)				\$245 (93%)			
Cost per successful drug (millions)	\$24	\$49	\$146	\$62	\$128	\$185	\$235	\$44
	\$281 (32%)				\$592 (68%)			
Time per drug (years)	1.0	1.5	2.0	1.0	1.5	2.5	2.5	1.5
	5.5 (41%)				8 (59%)			

Figure 2: The phases and costs of drug development [1][2].

Many computational methods fall under the umbrella term of binding affinity models, which attempt to predict the binding affinity of a given compound and target protein. They can significantly speed up the early stages of the preclinical trial phase. For instance, using these methods, hit identification and lead optimization can be accelerated. First, a million chemical binding affinities are predicted in silico. So only compounds predicted to have high affinity are tested in vitro, reduced wet laboratory experiments are needed. After hits have been verified, they can be lead optimizers by utilizing the model to predict binding affinity for similar compounds, and then checking binding affinity using experimental methods only for compounds that are predicted to have higher affinity.

1.5. Thesis Layout

The first chapter of this thesis will introduce the project and its goal. The second chapter will consist of a literature review and a background on prior research in the same area of study. Chapter three has two distinct parts. The materials used in the project, as well as the method by which good results were achieved, are described here. Chapter four will cover the implementation specifics of the system. Next, we will evaluate and test the proposed solution in chapter five. Chapter six, which concludes the project, had problems and required future work.

CHAPTER 2: Background and Literature Review

2.1. Background Information

2.1.1 Cell

The cells are the fundamental building blocks of life. To form a cell, cells contain the hereditary material and can make copies of themselves. Cells have many parts which serve specific purposes. A gene is composed of a small section of DNA that provides instructions for a specific protein as shown in *Figure 3*: Clarify the shape of DNA, Genes, and Chromosomes.. The function of genes is to store hereditary information. Each gene contains the information needed to create specific proteins needed by the organism. DNA is the genetic material in the cell and can be found in chromosomes and mitochondria. A chromosome consists of many genes. DNA is a coiled, double-helix shaped molecule that resembles a spiral staircase.

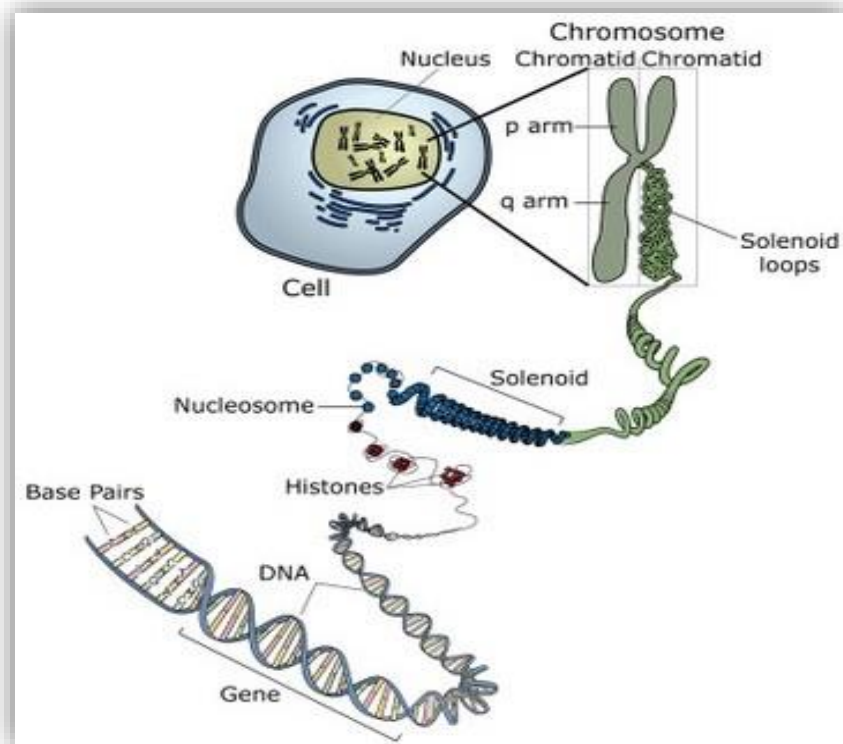


Figure 3: Clarify the shape of DNA, Genes, and Chromosomes.

2.1.2 DNA

DNA is composed of chemicals called nucleotides. DNA is the guidelines for life. Adenine (A), thymine (T), guanine (G), and cytosine (C) are the only four nucleotides commonly found in all organisms except for some viruses. DNA is built up of sugar molecules, phosphate groups, and base pairs of (A, G, C, T) As is evident

in this *Figure 4*: Clarify the shape of the DNA.. Always matches T and G always matches C in DNA strands [15]. During the replication process, DNA is able to copy itself.

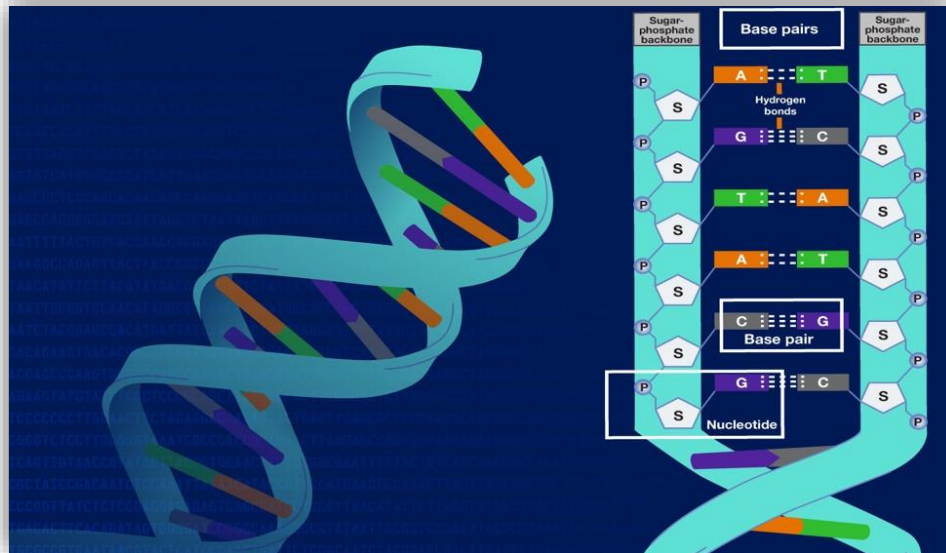


Figure 4: Clarify the shape of the DNA.

In addition, it can also be translated into RNA. During transcription, information contained in DNA is sent to RNA, which produces uracil (U) instead of thymine (T) in order to fit with the pattern of adenine. After transcription, RNA would be translated into amino acid chains, and the amino acid chains will go on to form proteins. When a nucleotide is translated into an amino acid, every three nucleotides determine one type of amino acid. Proteins are made up of chains of amino acids as demonstrated in this *Figure 5*: illustrates the process of transcription and translation in a cell..

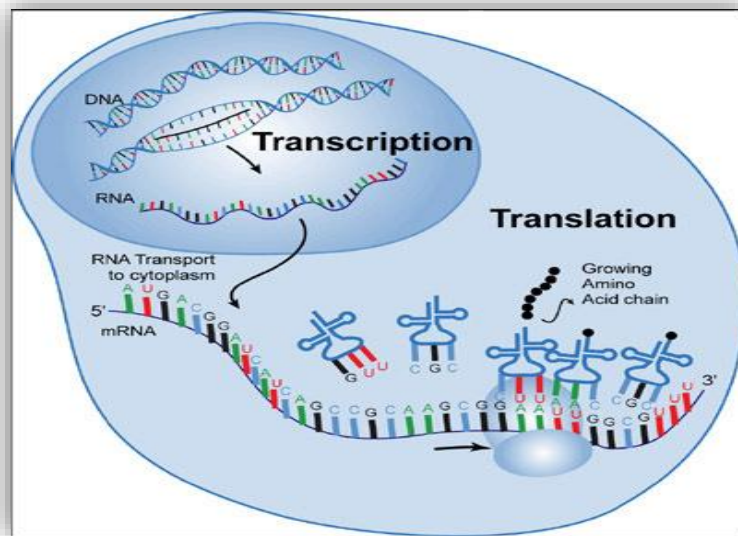


Figure 5: illustrates the process of transcription and translation in a cell.

2.1.3 Protein

The protein is composed of one or more polypeptide chain lengths. Proteins are large biomolecules, which are referred to as macromolecules. The protein molecule is made up of 20 different amino acids throughout particular. Some proteins serve many functions, such as antibodies, transport proteins, contractile, storage, and structural proteins, while others act as enzymes, hormones, and structural proteins[16]. Proteins are absolutely essential to organisms, and they are involved in almost every aspect of cell process. It is estimated that there are 1 to 3 billion proteins in human bodies [17]. In *Figure 6*: The primary structure of a protein is represented by each smaller circle, which implies an amino acid. The primary structure of a protein is determined by its amino acid sequence. This information is the most publicly available data we can obtain, as Uniprot (<https://www.uniprot.org/>) is available [18]. By clicking on “Swiss-Prot” and then “Human” and then “Download”, all human proteins as well as their sequences can be found and downloaded. Additional information such as the function and gene ontology information contained in Uniprot is available to users who select “Columns” before downloading. The dataset downloaded can be further processed to meet specific needs.

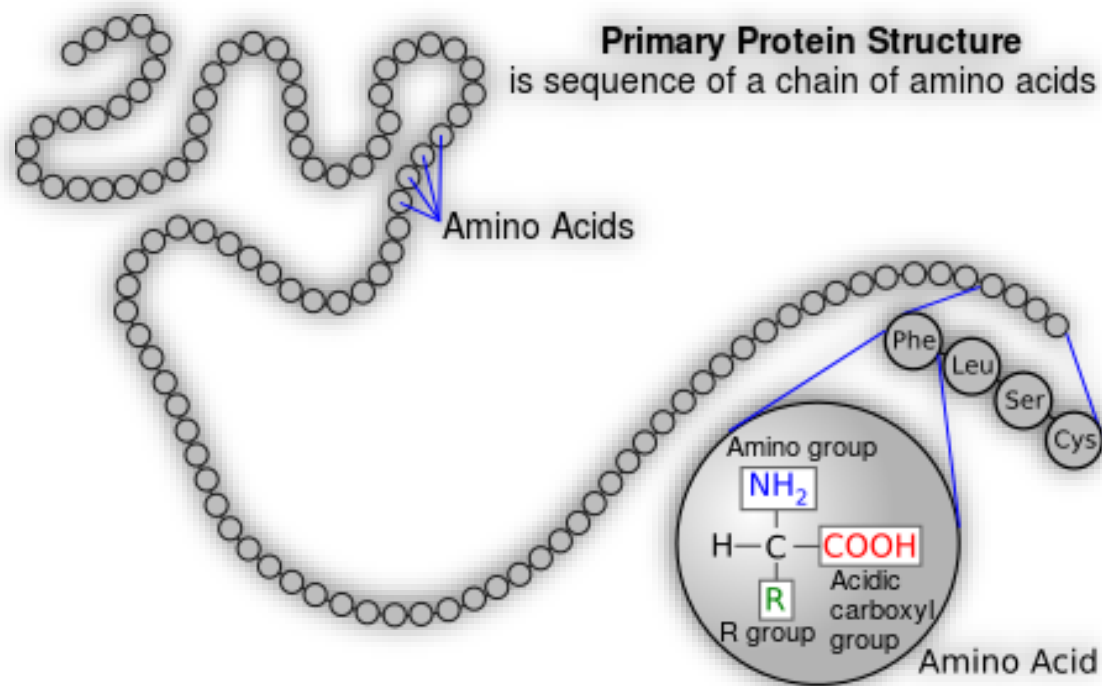


Figure 6: The primary structure of a protein is represented by each smaller circle, which implies an amino acid.

As shows Figure 7: Illustration of the relationship between polypeptides, proteins, and amino acids. in Protein is composed of one to many polypeptides. An amino acid or monosaccharide residue may be found in a protein or starch molecule. The amino acid residue is a part of an amino acid that gives it a unique identity from other amino acids. Its structural features, such as how it interacts with water help guide the construction of a finished protein.

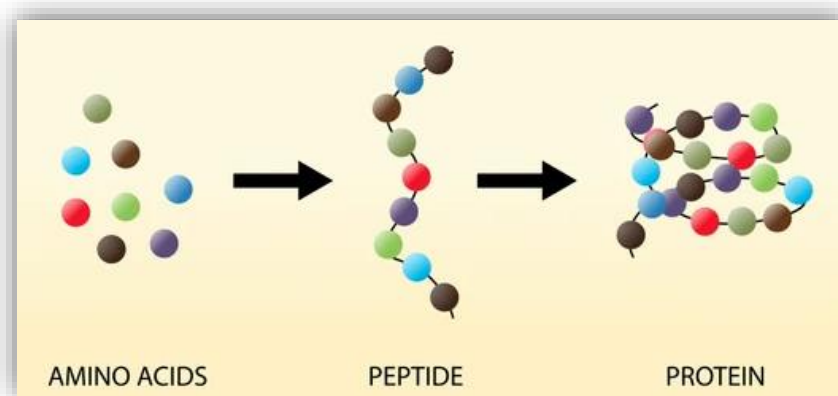


Figure 7: Illustration of the relationship between polypeptides, proteins, and amino acids.

According to the *Figure 8: The four levels of protein structure.*, proteins have secondary, tertiary, and quaternary structures as well. Local structures are stabilized by hydrogen bonds within the protein chain itself. Alpha helix and beta pleated sheet are the most common local structures. In other words, the tertiary structure is the overall spatial relationship of the secondary structure of multiple chains. "Tertiary structure" is sometimes used interchangeably with "fold". The tertiary structure determines the primary purpose of a protein. The quaternary structure describes the structure of proteins that are grouped together. So, both the protein's primary and secondary structure share a sequence of amino acids the order of the amino acids determines the shape of a primary structure. Therefore, the amino acid sequence that specifies the primary structure of a protein is codified by the DNA sequence Specific tertiary interaction locks (Sequence) prevents a structure from existing unless the parts of a domain follow the established order. A tertiary structure emerges from the interactions among the amino acid side chains. When the proteins that are rich in tertiary structures are brought together through intermolecular interactions, they form tertiary complexes. Protein secondary structure can be used to predict tertiary structure as well. Proteins follow secondary structure according to their H-bonding pattern.

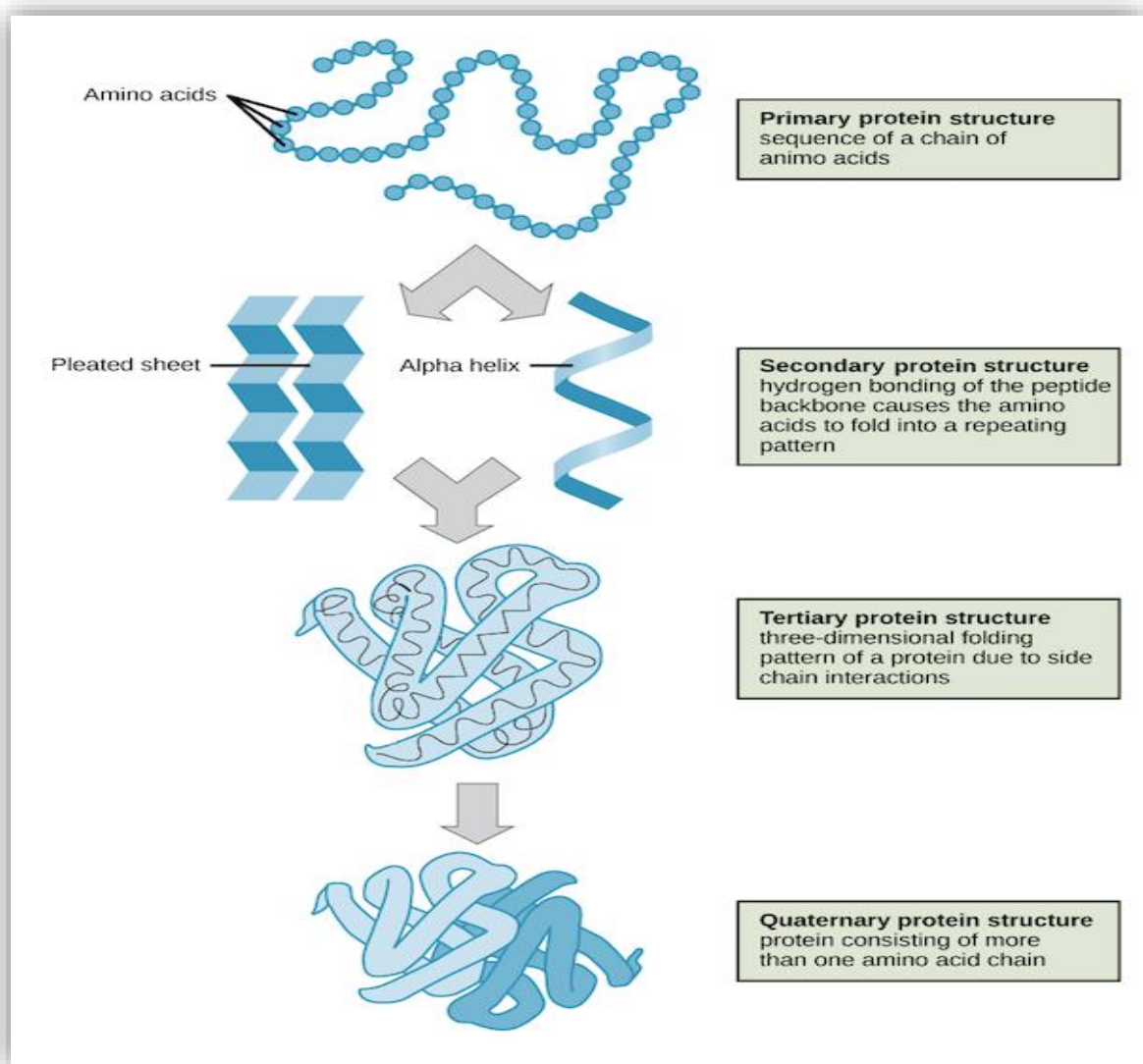


Figure 8: The four levels of protein structure.

Also as mentioned, the primary structure data of organisms is the most widely available data. FASTA format is commonly used to represent this kind of data. Single-letter codes are used to identify the peptide sequences of each protein in the FASTA file. Each protein has two identities: the protein name and its amino acid sequence. The line starting with '>' characters and containing protein names starts with a '>' sign and ends with the protein name. Each letter in the sequence line encodes an amino acid. Figure 9: An example of a FASTA file. provides an example with the amino acid sequence of protein P32479. This sequence has the letters MKVVKFPWLAHREESRKYEIYTVDVSHDGKRLA.

```
>P32479
MKVVKFPWLAHREESRKYEIYTVDVSHDGKRLA
>P43571
MGIRRLVSVITRPIINKVNSSGQYSRVLATREDQDKASPKYMNDKIAKKPYTYR
```

Figure 9: An example of a FASTA file.

Proteins play a prominent role in the natural environment. Proteins not only play a functional, but also a structural role. they are large, complex molecules that provide many important functions throughout the body. They are necessary in cells and contribute to the structure, function, and regulation of tissues and organs. These proteins give the cells structure. Proteins are macro-molecules that are made up of a specific amino acid sequence shows in Figure 10: Protein structure representation. (PDB-id: 1a1e). Small molecules, composed of an amino group (NH₂), a carboxyl group (COOH), and a hydrogen atom attached to a carbon, are known as amino acids. The three-dimensional structure of proteins is known as conformation. It is important to note that rather than being a rigid lump of material, the systems can also have moving parts whose mechanical actions are coupled to chemical occurrences. The chain of amino acids is flexible[19].

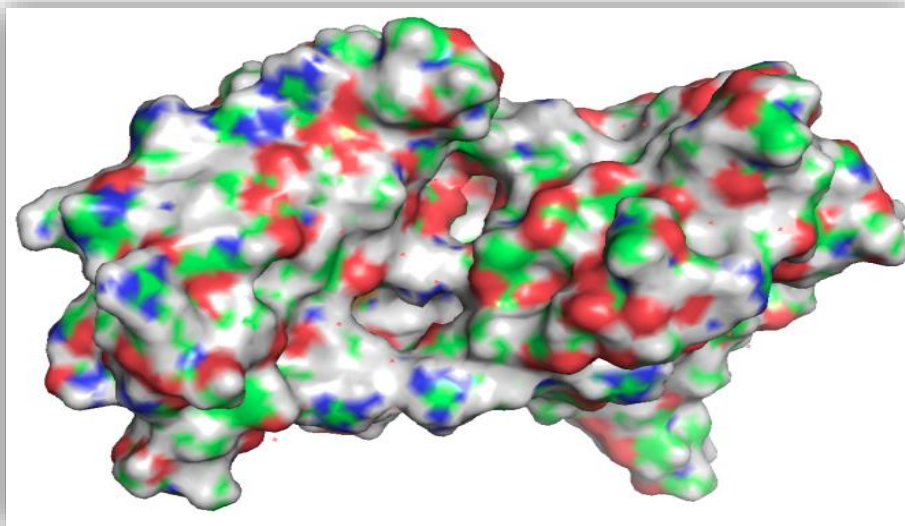


Figure 10: Protein structure representation. (PDB-id: 1a1e)

2.1.4 Background on the protein adaptor

To rapidly respond to changes in the environment is essential for the continued existence of cells. A cell can receive many signals and then process them into a unified strategy. Furthermore, a cell also could send signals to the surroundings. Signals could be mechanical, chemical, optical, or thermal [20].

Examples of biological signals are growth factors, neurotransmitters, extracellular components, and extracellular matrix. Because it is possible for any molecule to travel in the environment, these signals have an extensive local or global effect [21]. Neurotransmitters are an example of the very short-range signaling molecules. The travel of neurotransmitters could be between adjacent neurons or muscle cells. The other long-range molecules are the follicle-stimulating hormone, which travels from the brain to the ovary to initiate the egg maturation. Sensory cells in the skin, ear, and the human blood vessels could be exposed to mechanical stimulation.

The signals travel into a cell and are then translated by many protein receptors [50]. Molecules elicit a distinct type of a physiological response. The Dopamine receptors are different, and others can even respond to light or pressure. The receptors contain two membrane-spanning regions such as transmembrane proteins; one binds the signaling molecules outside the cell and the other starts the signal-specific pathways inside.

The class of membrane receptors is divided into three subgroups: G-protein-coupled [51][52], ion channel, and enzyme-linked receptors [53] as shown in Figure 11: This diagram shows the membrane proteins.. This is how these receptors make external signals into internal ones: ion-channeling, protein action, and enzyme-mediated activation. Because of the two areas of the cell (the internal and the external receptors), the signal molecules can act on the external molecule and the cellular components without entering cell of signal molecules. The other receptors are located deep within the cell, even in the nucleus, and may bind to molecules that can pass through the plasma membrane (gases, steroid hormones, and such)[54].

When the receptor receives a signal, it can conduct a series of biochemical reactions because of the shape that has been imposed by the interaction. these signal-

cascade pathways can amplify the message to create multiple cellular responses. The activation of receptors can cause the production of small molecules, such as cAMP, which are known as second messengers, to activate and regulate cellular signaling pathways [55]. The activation of adenylyocyclase is involved in the production of hundreds or thousands of cAMPs that activates protein kinase, which phosphorylates multiple substrates. When the enzyme phosphodiesterase degrades cAMP, the cAMP signaling ceases.

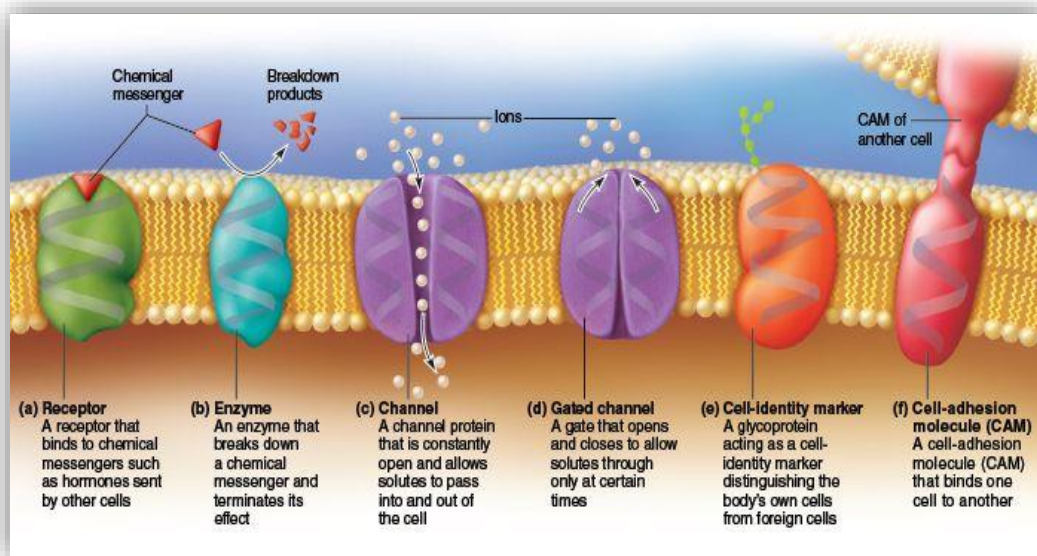


Figure 11: This diagram shows the membrane proteins.

The best drugs for biological activity can be used to manipulate the signaling proteins. Therefore, to gain a deeper understanding of the mechanisms of how proteins work with signaling is critical. Although the use of experimental methods and tests is expensive and time-consuming, theoretical treatments are the ones that offer a practical solution for this screening. Linking protein structure to signaling activity may be done using Machine Learning techniques. The molecule's three-specific characteristics can be represented in molecular descriptors, molecular topology, peptide-dynamic representation, peptide sequence, and physical characteristics of the amino acids.

2.1.5 Virtual Screening

The most popular technique in drug discovery is a physical screening of a large library of chemicals against some biological target (high-through put screening). Compounds are tested to see if they bind together with the target. Another approach would be to use Virtual Screening, a computational technique, to search for chemicals

in large libraries [19]. Machine learning models are currently being applied to develop scoring functions fast enough to help perform virtual screening in a quicker timeframe. As shown in Figure 12: Virtual Screening, that this is the use of virtual screening to find new ligands.

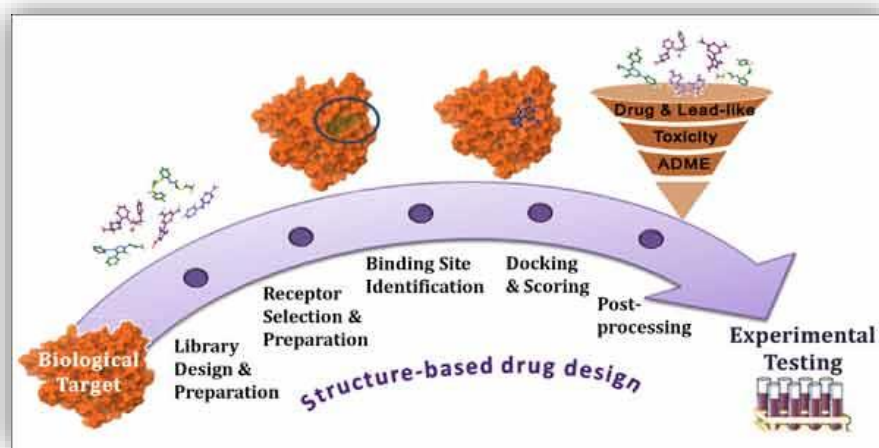


Figure 12: Virtual Screening.

2.1.6 Deep Learning in Bioinformatics

The area of machine learning known as deep learning is a part of it [20]. It is inspired by the artificial neural networks structure and function, which is also referred to as artificial neural networks. Any computational programmed can be thought of as a transformation on given inputs at a high level. Machine learning algorithms learn the computations by learning how they are traditionally done. The mathematical transformation is learned from previously observed examples, often without having rules defined. To give an example, researchers used to design specific rules to capture the characteristics of each number in the Handwritten Digit Recognition problem. While deep learning is appointed, only the structure of a network is developed. The network “learns” how to determine a handwritten number based on viewing a significant amount of labelled data, as shown in Figure 13: Handwritten Digit Recognition labelled data Every handwritten number is given a label that represents its actual value..

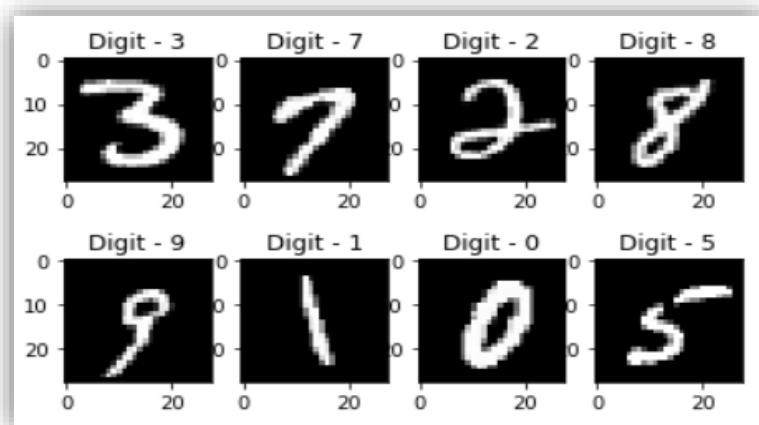


Figure 13: Handwritten Digit Recognition labelled data Every handwritten number is given a label that represents its actual value.

Many computational biology problems have been redefined as mathematical problems. A further example of this is sequence alignment, tree comparison, and similarity search. Many of these problems have very well algorithmic solutions. While this is true, there are also a large number of bioinformatics problems that are not even close to being understood, and they are extremely fundamental problems, such as protein region identification, protein structure prediction, protein-ligand interaction identification, and biomedical image classification [21]. It is difficult to solve these problems because the underlying mechanisms are complicated, and even biologists cannot provide clear explanations of all the factors and how they are connected.

As more biological data becomes available, the number of hard biological problems is increasing, and deep learning methods are increasingly used to try to solve them. As shown in Figure 14: This is an approximate count of published deep learning and deep learning bioinformatics papers year-to-date. , from 2010 onwards, there has been a dramatic increase in the number of deep learning bioinformatics methodologies.

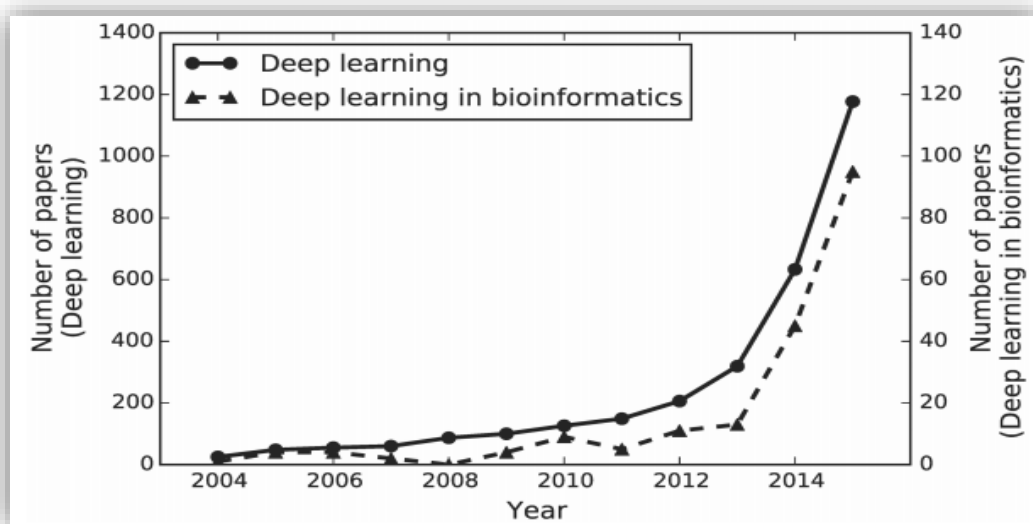


Figure 14: This is an approximate count of published deep learning and deep learning bioinformatics papers year-to-date.

This is caused by several factors. First, an enormous number of biomedical data has been compiled, which provides a solid base for deep learning performance improvements. Let's take an example like the one in Figure 15: Overall structural growth over the past year, as tracked by PDB Statistics. , where from year 2000 to 2020, the actual number of PDB [22] released structures began to rise from 13,589 to 162,529, which is nearly 12 times larger in comparison to two decades ago.

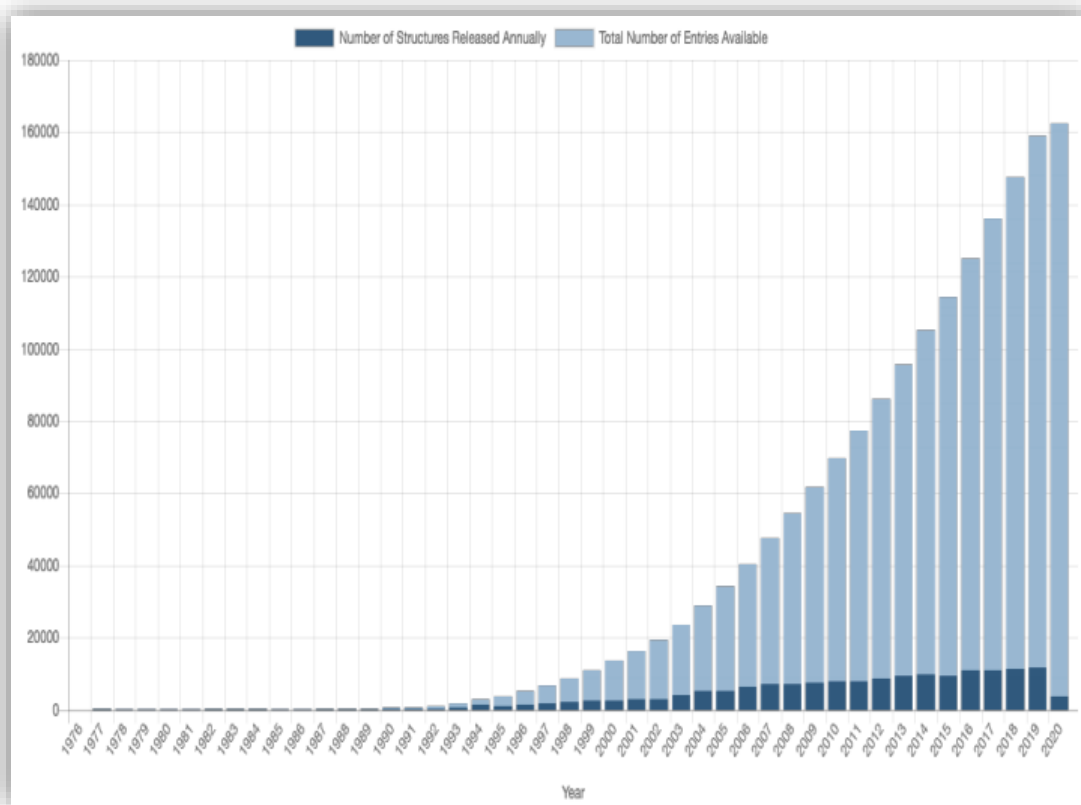


Figure 15: Overall structural growth over the past year, as tracked by PDB Statistics.

As displayed in Figure 16: as the quantity of data increases, learning performance improves., deep learning methods tend to improve with more data, while traditional methods become optimised when given a specific amount of data. Second, significant effort has been invested in hardware co-design for deep learning software in order to speed up training. There are various deep learning frameworks and compilers available on the software side. Several of them be using free and open source software with tight community integration. In the popular frameworks, you will find Google's TensorFlow [23], PyTorch [24], Caffe [25], and high-level API libraries like Keras [26]. More recently, MindSpore [27], a software developed by Huawei. Besides compiler optimizations, many other possibilities of optimization have also been explored. For instance, Google's Machine Learning Intermediate Representation [28] project aims to unify the deep learning intermediate representation in order to make APIs and hardware integration more seamless. The goal of the TVM [29] project is to have a well-tested kernel implementation on various hardware.

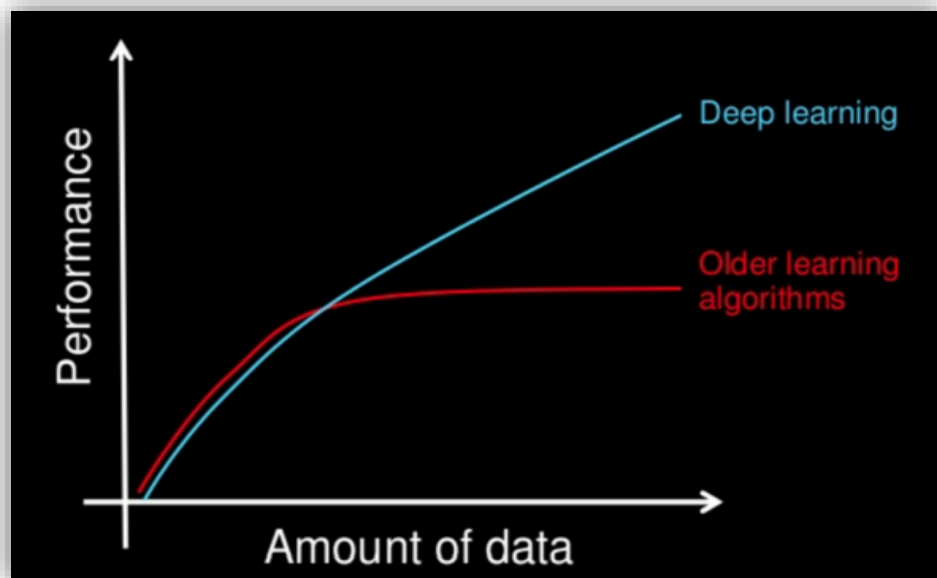


Figure 16: as the quantity of data increases, learning performance improves.

The software has made using deep learning easier and more efficient during runtime. General purpose CPUs were first used to support deep learning on the hardware side. Because of the recent popularity of GPUs, GPUs are now very popular because of their vector computation unit, which greatly accelerates deep learning vector computations. The advancement of specialised hardware that focuses on the cube unit by major companies has also occurred in the recent past. The Cube unit can perform matrix multiplication in a single cycle, further speeding up deep learning computations. A famous case in point is Google's Tensor Processing Unit (TPU), the chips acquired by Habana Lab (from Intel), and Huawei's Ascend chips. Figure 17: Google's increase in TPU speed on Resnet50 training, illustrates the full setup of TPU potentially speeding up 200x the training on Resnet50. Third, complex biological problems are too difficult for hand-crafted algorithms. Deep learning on the other hand, on the other hand, benefits from the rising volume of data and uncovers the underlying patterns hidden in high-dimensional data that the human mind cannot perceive. Fourth, many researchers have entered the field of AI because of the influence of events and completions. To name a few, in the 2012 ImageNet Large Scale Visual Recognition Challenge, AlexNet [30] came in first and gained an enormous 10.8% advantage over the second place. In October 2015, AlphaGo [31] defeated Lee Sedol, a professional Go player, on a full-sized board. In 2018, the CASP13 protein-folding competition had Deepmind's AlphaFold [32] as its winner.

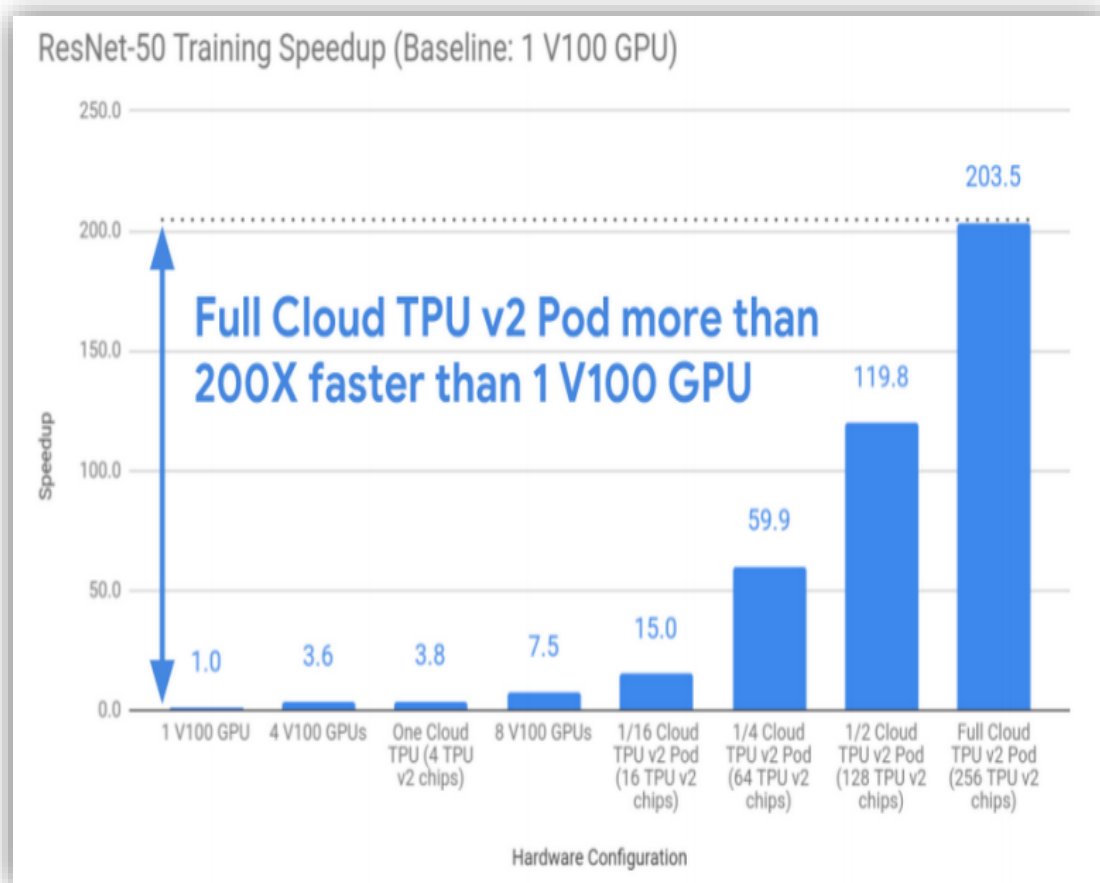


Figure 17: Google's increase in TPU speed on Resnet50 training.

2.1.7 The significance and background of protein adaptor research

The Protein structure is classified in the form of a quantitative, structural, and functional Quantitative-Structure-Activity Relationship (QSAR) model [30]. the QSAR models were widely applied to antifungal, macrofunguon, macroviral drugs, and wider still to macromolecular macromolecules such as oligonucleotides and nucleic acids [31][32]. Formal previous studies dealt with the antiprotein action, transmembrane protein regulators, cell death activity-related or cancer-related proteins [33]. This class of modelling strategies also utilises other class types of models, and the structure or sequence of the proteins are already determined [34].

The use of an evaluation index for protein adaptor prediction methods in this field is very similar to that of evaluating the accuracy of the algorithm. This is the most frequently used set of protein adaptor prediction indicators. They are: sensitivity, accuracy, specificity, precision, and Mattheu's correlation coefficients in Figure 18: The most frequently used set of LBS prediction indicators..

$$\begin{aligned}
Sen &= \frac{TP}{TP+FN} \\
Acc &= \frac{TP+TN}{TP+FN+TN+FP} \\
Spe &= \frac{TN}{TN+FP} \\
Pre &= \frac{TP}{TP+FP} \\
MCC &= \frac{(TP \cdot TN) - (FP \cdot FN)}{\sqrt{(TP+FP) \cdot (TP+FN) \cdot (TN+FP) \cdot (TN+FN)}}
\end{aligned}$$

Figure 18: The most frequently used set of LBS prediction indicators.

For TP (TruePositive), the number of samples in which the protein adaptor is correctly predicted is counted. For TN (TrueNegative), the number of samples in which the protein adaptor is incorrectly predicted is counted. For FP (FalsePositives), the number of samples in which the protein adaptor is predicted correctly is counted. For FN (FalseNegatives), the number of samples in which the protein adaptor is incorrectly predicted is counted[16][20].

2.2 Previous Work

Here are various literature review resources, each of which will be analyzed and evaluated using the previously stated approaches.

2.2.1 Classification of signaling proteins based on molecular star graph descriptors using Machine Learning models.

2.2.1.1 Strategy and Structure

In Figure 19: A methodology flowchart showing the creation of molecular graphs and the use of machine learning for classifying proteins. is an example of the methodology employed in the current study. Two classes of proteins are determined: Those that contain only signaling sequences are known as signaling proteins, and those that contain only non-signaling sequences. Signaling and non-signaling protein amino acid sequences are present in the database. In the third

stage, the sequences of amino acids were also converted into topological protein topology indices, with S2SNets. Classifiers that describe the amino acid sequence representation are then used to discover the perfect QSAR classification model which can be used to predict the signaling pathway for new protein sequences. Machine learning methods are often applied to biological problems throughout order to find out if they have biological functions or functions as discussed previously, or more recently for instance.

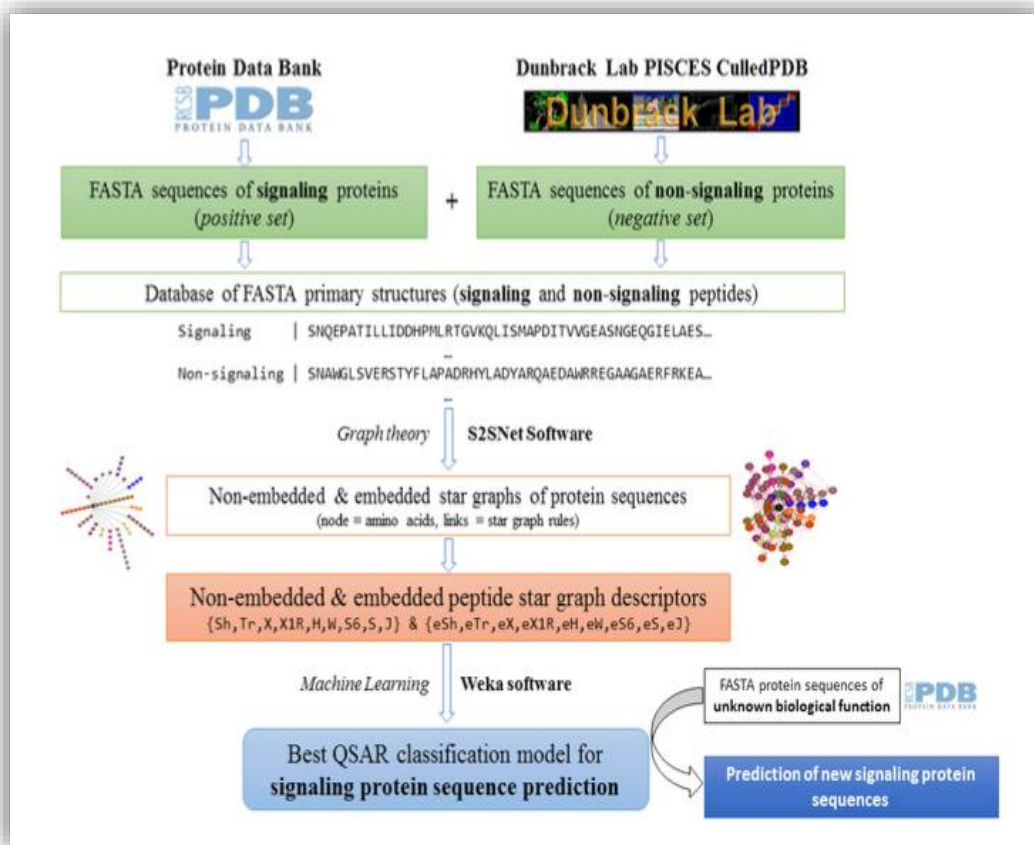


Figure 19: A methodology flowchart showing the creation of molecular graphs and the use of machine learning for classifying proteins.

2.2.1.2 Data

The protein database that contains both signaling and non-signaling primary structure-related protein sequences in FASTA format.

2.2.1.3 Method Evaluation

To test the predictive models for signaling and non-detection classes, a confusion matrix is used. Some commonly used accuracy measures used for two-classifiers are: classification rate, precision, sensitivity, and area under the ROC

curve (AUROC). the lower the effort required in testing and inspection, and the fewer nonfunctional modules are found. However, in order to obtain an F-measure, the benefits of recall must be traded off against precision. The AUROC is preferred over precision for classifier performance comparison.

2.2.1.4 Results

It represents the final dataset with samples (predicting protein chains) that demonstrate and samples that disprove protein synthesis/ The final dataset consists of 1824 signifying proteins (a data point) and those that falsify the hypothesis (2432 counter-signalling proteins) (nonsignaling protein chains). A lot of these proteins were analysed with the S2SNet programme in order to have the 42 different topological indexes shown in this research. Classification performance has been generated using six different classifiers. It has now been established that four distinct approaches have been used to get baseline best results from any feature-selection technique. After an experimental analysis, statistical comparison study, they found that null-hypothesis RFE-LAP is statistically superior to the others.

This experiment will determine the starting performance level for the use of six cutting-edge Machine Learning algorithms in protein classification. therefore, a more detailed evaluation of the project method should be conducted. The results are obtained with SVM with RBF that give the best accuracy for the best AUROL value of 0.948. These results were compared to the others, and it was found that the use of LibLINEAR f (0.657), NBe (0.6.19) and J48d resulted in significantly improved them (0.687). Additionally, the effects of RFc (0.935) and KBs (0.939) were also reduced, but by a smaller margin. Figure 20: ROC curves for model. depicts the ROC curves for reference models.

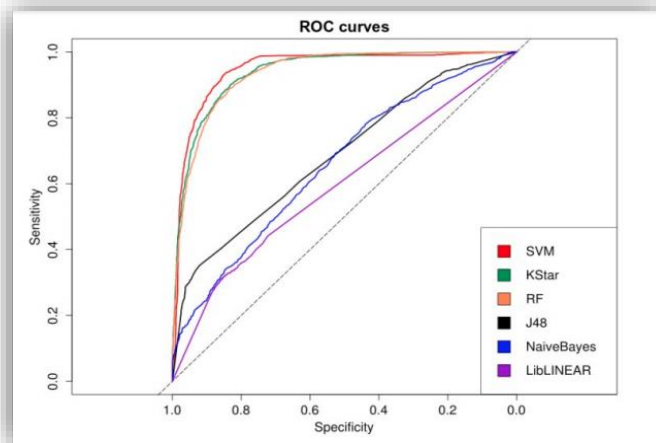


Figure 20: ROC curves for model.

For Bioinformatics, it is common to carry out a feature selection process to reduce the total number of features while improving algorithmic performance. Four different Feature Selection Classification Models have been tried: Particle Swarm Optimization with SVM, RF Support Vector Machines with RBF, and RFE-Random Forests (RFE-RF). In order to ensure the models have been run 5 times, see the error bar plot in Figure 21: Feature selection achieved..

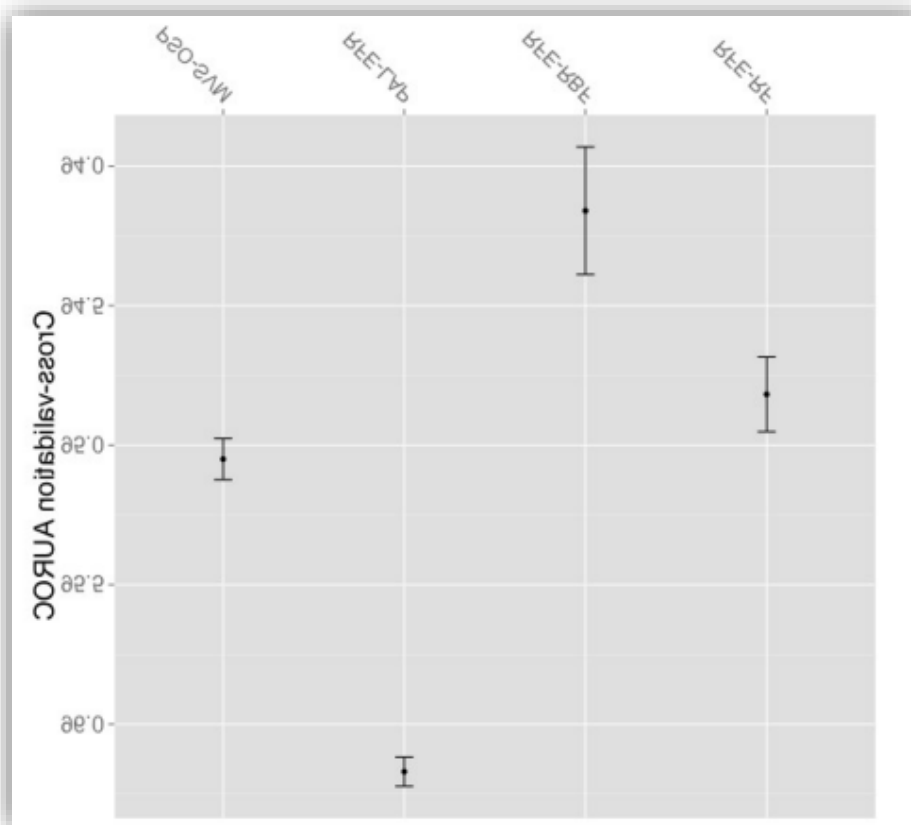


Figure 21: Feature selection achieved.

2.2.2 Identification of Clathrin proteins by incorporating hyperparameter optimization in deep learning and PSSM profiles

2.2.2.1 Strategy and Structure

The amino acid sequence analysis of the secondary structure set of features was done using the PSSM matrices. PSSM is a collection of all biological and protein motifs in abundance two sequences that have the same component building blocks are made from different structural models of the same structure. Thus, PSSM profiles were successfully adopted and applied in various biological studies, e.g. RNA-binding [16] and dual tropic HIV-1[40]. The dataset was retrieved throughout FASTA format, so it had to be transformed with PSSM profiles. The sequence alignments in the non-redundant database was completed using PSI-BLAST[41].

A 400D input vector was each time subdivided by a sequence length, and then inserted into the neural network. They would use a 2D convolutional neural network to conduct this study; this is a standard neural network for deep learning.

Results for 2D CNN (as far as far as they have been applied to bioinformatics) have been found in the following areas: the proteins involved in electron transport chains [10], cytoskeleton motor proteins [52], Rab proteins [7], and SNAREs [25]. TensorFlow was used as the back-end for their deep learning. The GPU and CUDA was also used to speed up the overall system performance for the most part, CNN is made up of layers, each of which perform a particular transformation on its input. the architecture of our CNN was built using all layers in the proper order As evidenced in the case of these studies on this subject [39], the architecture and parameters of an effective model must be found using hyperparameter optimization[27].

Different problems and datasets have different layers and settings that need to suit their needs: Firstly, input vector: This study employed 20x20 PSSM input parameters by using these matrices, we propose a way to identify general proteins as the common components of clathrin Since we are using 20x20 matrices for training our 2D CNN model, we assumed that the model represents images of 20x20 pixels. The goal of using a 2D CNN model is to reveal the hidden features from the PSSM profiles. Secondly, an important core building block of a CNNs is

the convolution. Convolutions are used to extract features from the 2D input matrix. The convolution layer shifts the sliding window of values over the input. useful features in the process of PSSM profiles are learned while retaining spatial relationships among the numeric values. To build our model, we applied convolution to the 2D matrices with 3x3 sliding windows and then learned the features one unit at a time. Each neuron got input from the previous layer and then trained on them.

Sometimes, the pooling layer is inserted among the convolutional layers with the purpose of reducing the size of the matrix for the following convolution. The procedure used by this layer is referred to as “down sampling as it gets rid of superfluous values while retaining the most representative features” The pooling layer also incorporates a sliding window or regions that are moved across the input matrix, which transforms the values from their individual values into representative values. in the window (averaging) (average pooling). using a standard design with 3x3 and 3x3 grids washes.

Thirdly, Dropout was applied to improve the predictive capacity of the model was a feature that was added to the present design. the model will eliminate the neurons of a certain percentage of error when 'dropout' layer values are used, the network ignores these neurons Dropout has been used to evaluate their model's performance/capabilities.

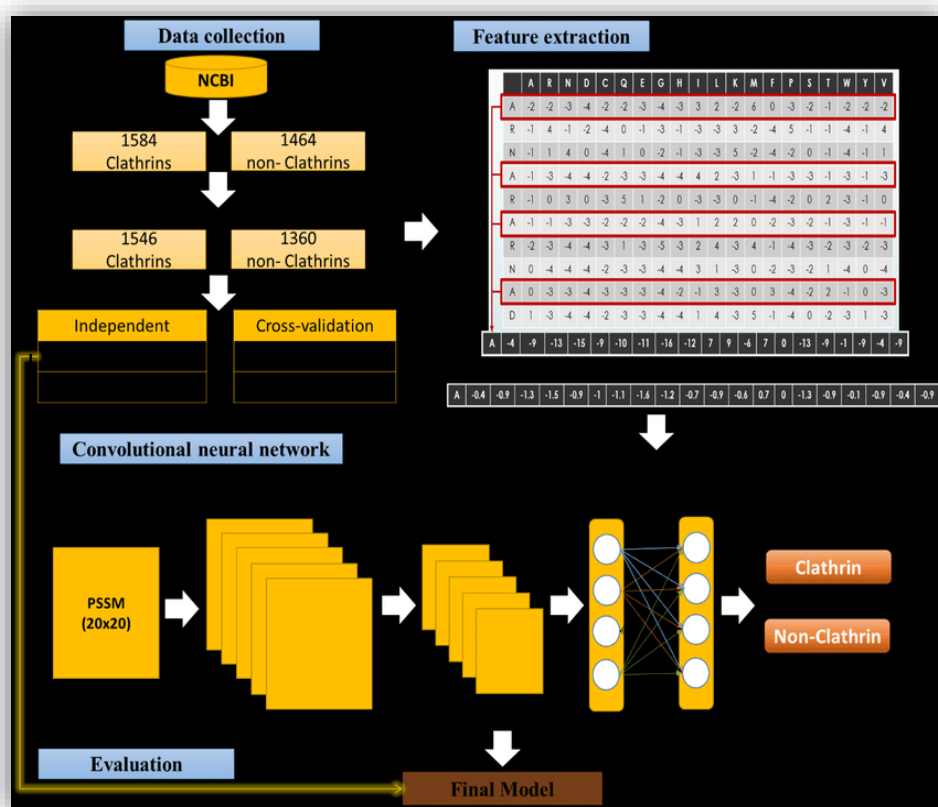


Figure 22: 2D Identification of Clathrin proteins by incorporating hyperparameter optimization in deep learning and PSSM profiles.

2.2.2.2 Data

To begin with, they searched the NCBI databases for the word “clathrin”. Because of the proposed problems, they gathered a negative set of general proteins. In order to generate a precise model, it is necessary to collect a dataset with the same functions and structures as the original. that's hard to design a precise but increases our contribution to the model. Authors should reduce the similarity of sequences in most bioinformatics problems to below a threshold of 30-40 percent. Authors should reduce the similarity of sequences in most bioinformatics problems to below a threshold of 30-40 percent. Nonetheless, in order to fully utilize the deep learning, this research wanted to remove 100% of the duplicate’s sequences. They were able to collect sufficient data for deep neural networks, allowing us to generate hidden information within each sequence. To perform this step, they used BLAST, which is a commonly used tool for biological sequence clustering. Data was then divided into cross-validation and stand-alone datasets. While cross-validation was in progress, the in the control group, there were a total of 1288 clathrins and a total of 1133 nonathrins, respectively. the dataset used in that study.

2.2.2.3 Method Evaluation

The study's main objective was to estimate if there is a clathrin sequence, so They used the phrase “Positive clathrin protein” to represent the absence of clathrin and “Non-clathrin protein” to represent the presence of a clathrin. They calculated the accuracy of each model by using a ten-fold cross-validation process on the training dataset. Using the 10-fold cross-validation to find the best model, hyperparameter optimization was used to find a good model for each dataset. To determine the current model's capacity to predict an independent dataset. This predictive performance is gauged using the established by using sensitivity, specificity, and MCC (Matthews coefficient of correlation).

2.2.2.4 Results

They analyzed the distribution of amino acid contents in clathrin and non-clathrin sequences by computing the relative frequency. They realized that it was not that different between two different kinds of data, but rather many points of similarity. the three amino acids, C, P, and N, occurred most frequently in clusters around the clathrin proteins. On the other hand, L and I are the most frequently found non-clathrin proteins. Thus, these amino acids undoubtedly were important in identifying clathrin proteins. Thus, our model might accurately forecast clathrin proteins from these special features. They 2D CNN was implemented with the TensorFlow package, which in turn used the Keras backend. First, they ran experiments using four different convolution layer settings: 32, 64, 128, and 256. During the 10-fold cross-validation, they were able to accurately distinguish between the structures of 32, 64, and 128 number of folds, their model correctly-labeled sequences had an 88.4% average accuracy. Sensitivity, specificity, and MCC, were achieved at 90%, 86.6%, and 0.77. Because of this, they implemented a convolution layer in the hidden layers. They proceeded to optimize the neural networks using rmsprop, adam, sgd, and adadelata. every time the optimization was carried out, the model was reinitialized, i.e. a new network was built to provide a fair comparison between the different optimizers. They selected Adam, who has good performance, to build their final model. Also, Adam has been chosen in similar work. They further discovered that the validation accuracy could not

improve as training accuracy increased. Thus, we decided to finish our training process at epoch 80 to reduce the time and reduce overfitting.

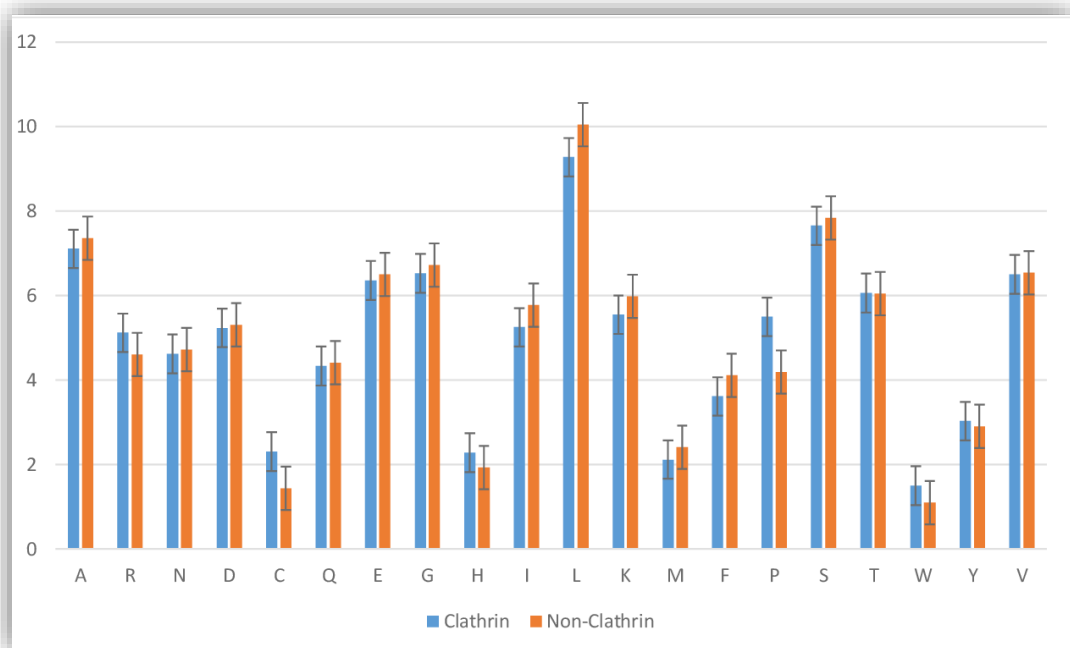


Figure 23: The amino acid composition of the disulfide chain and non-free disulfide sequence.

Chapter 3: Material and Methods

3.1. Materials

In this chapter many of the required and frequently used datasets, tools, and environments will be discussed.

3.1.1. Data

Due to the fact that our protein adaptor classification research is the first of its database is created manually, a dataset has been manually designed from well-known sources. In fact, approximately seventy-five percent of all datasets were created and we implemented the same techniques such as downloading data from known sources like UniProt and other steps to increase the number of data sets by adding new proteins to the dataset to obtain higher accuracy in the final result. It was determined that we found excellent tools for gathering data on gene products from Uniprot [61]. We retrieved all the UniProt protein sequences with GO molecular function

annotations that related to adaptor. An important selection criterion was that we applied thorough scientific study to the sequence, which means that they were found in participant papers. Thus, the complete query for data collection was as shown in the Figure 24: Query for getting the adaptor proteins..

```
"keyword:"adaptor" OR goa:"adaptor") AND reviewed:yes"
```

Figure 24: Query for getting the adaptor proteins.

Following this step, we obtained 4,049 adaptor proteins of all organisms. A solution to the proposed issue was resolved in binary classification, thus, we built a negative set of general protein sequence samples. In fact, our classifier was set out to detect adaptor proteins and non-only adaptor proteins. We needed a list of adaptors and non-adaptors for training the model properly. However, in reality, the number of non-adaptor proteins is already in the hundreds of thousands and for the dataset that related to adaptor proteins, it would definitely be a much smaller number of the non-adaptor proteins. This will seriously affect the model's performance because of having imbalanced data. Thus, in the majority of the problems that deal with bioinformatics, scientists are able to make assumptions only on the subset of negative protein data and treat it as a general protein. In this research, we selected this membrane protein, which contains a large enough number of sequences and functions for the analysis. In short, we target area most of the membrane proteins from the UniProt databases and discarded adaptor proteins. Subsequently, all the sequenced data was subsequently passed through the BLAST [42] algorithm to identity redundant sequences with greater than 30% identity. To prevent the training of the model from overfitting, this step was important. Some sequences remained as adaptors for the benchmark dataset, and were further divided into 9695 non-adaptor proteins, while others were treated as adaptor proteins and the result was 1069 as shown in Figure 25: Illustration of preparing the dataset..

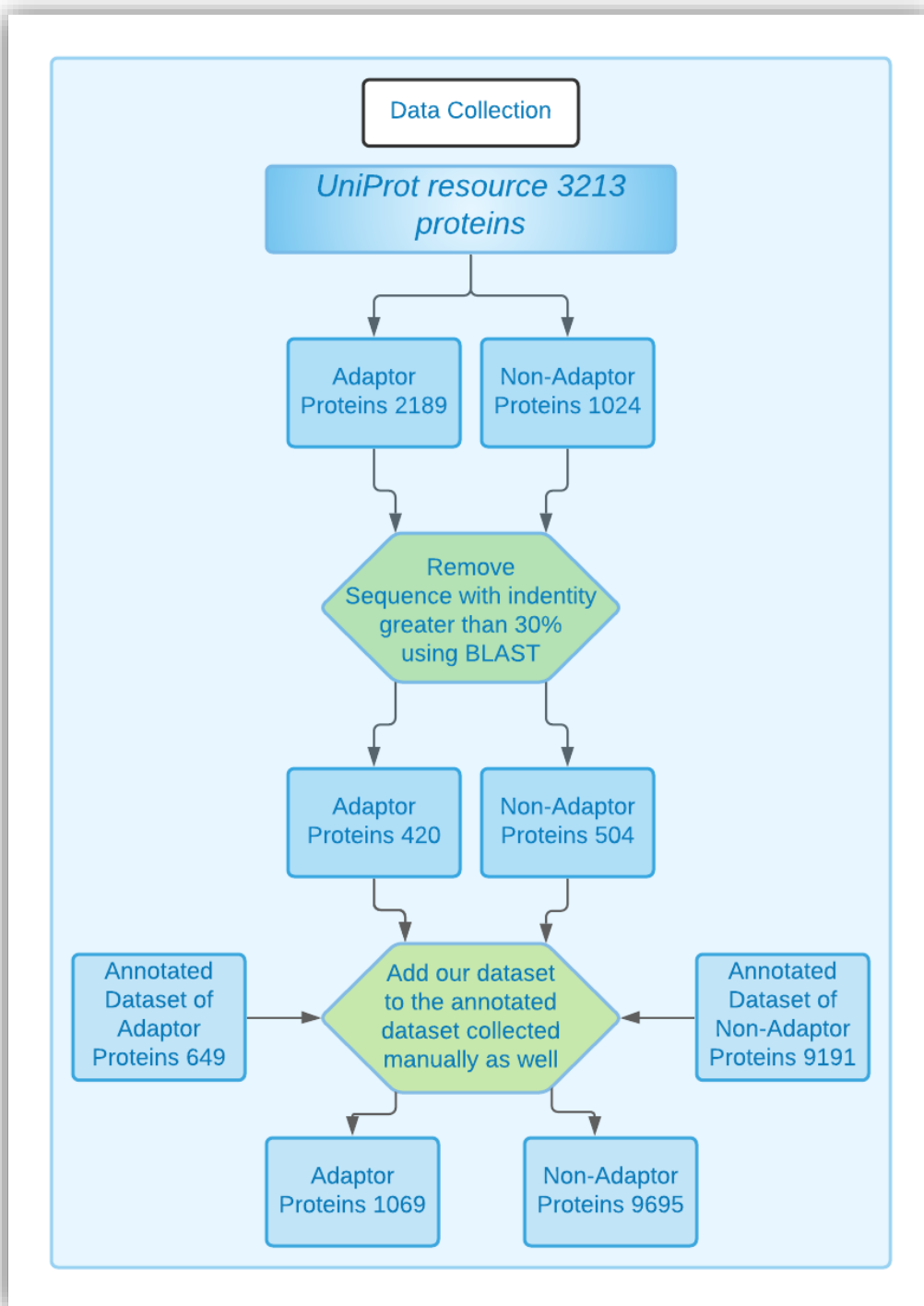


Figure 25: Illustration of preparing the dataset.

3.1.2. Tools

We used VMD to visualize each protein as shown in Figure 26: Visual Molecular Dynamics program. after downloading it in FASTA format from UniProt. Molecular visualization software called Visual Molecular Dynamics (VMD) is able to display, animate, and analyse large biomolecular systems employing 3-D graphics

and built-in scripting. In addition to supporting computers running Mac OS X, Unix, or Windows, VMD also includes source code.

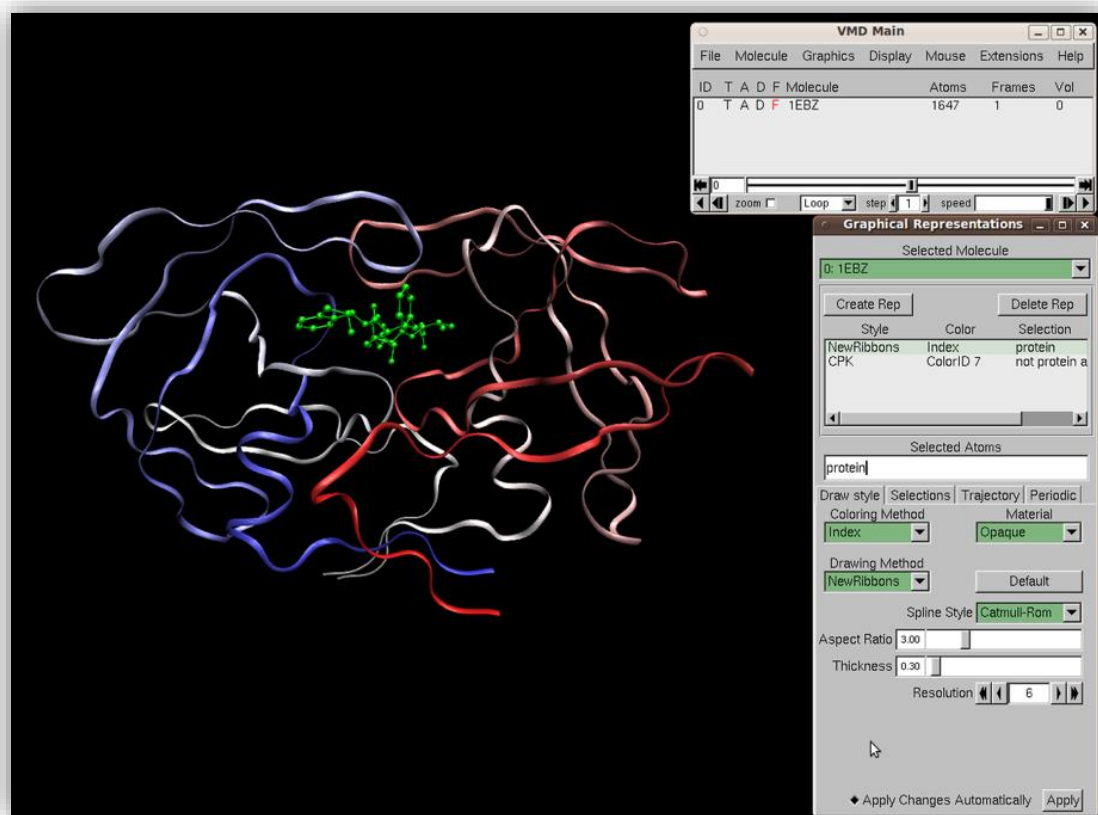


Figure 26: Visual Molecular Dynamics program.

Our thesis utilized Python to build an architectural model to predict whether or not a protein is an adaptor or non-adaptor. Python 3 is an open-source programming language that assists developers in their work and in integrating their systems. There are numerous Python libraries that include NumPy, OpenCV, Sequential, Activation, Dense, Dropout, Pyplot, Flatten, and ImageDataGenerator.

This thesis used Jupyter notebook in google Collaboratory for creating the code of the model. Equations, visualizations, narratives, code all within the same web application: The Jupyter Notebook is an open-source document representation of Python that is implemented in a web browser that you can both create and modify. Exhibited applications involve: transformation and data cleaning, numerical simulation, data visualization, statistical modelling, and machine learning.

We have been utilized libraries in google collaboratory like NumPy, sklearn, pickle, math, csv, and time. The goal of the NumPy library is to extend support for large, multi-dimensional arrays and matrices to include high-level mathematical functions. Scikit.learn is definitely the most helpful for machine learning. The sklearn package includes classification, regression, and dimensionality reduction methods. Pickle is a module in Python for object serialisation and de-unserialization. converts Python sequences, maps, dictionaries, etc to byte streams (zeroes and ones). Converting the byte streams back to Python objects is known as unpickling. We have access to math functions and constants in Python, that the we can use to carry out more intricate computations within our code. It's built-in, so you wouldn't have to do any installation to be used. the tabular file format used to exchange data between spreadsheets is a CSV (Comma Separated Values) CSV files save tables in plain text and numeric formats. Each data record is a separate. Python has a built-in module for working with CSV files that is called csv.

3.1.3. Environment

PyTorch is an open source framework of machine learning that uses the torch library, built for natural language processing and computer vision, developed mostly by Facebook's AI Lab. PyTorch is a strong in the sector of deep learning and machine learning, so it can be referred to as a research-first package. Python is among the most frequently used programming languages by data scientists. I use the term “Pythonic” when I am speaking of PyTorch to indicate a Python tool that is focused on the language. Some researchers have commented that while no other computer languages, except Python, have seen gains in machine learning algorithms that year; however, it's not a coincidence that Python and machine learning have both increased significantly this year. Compared to other deep learning libraries, PyTorch is simpler to learn. PyTorch is simple to use, making it easy for developers. Formal: Because PyTorch is deeply integrated with Python, you can use many Python-based debuggers to assist with the development process. The pdb and ipdb tools of Python can be used for this type of investigation in PyTorch.

The computational power required for this project is high, and the CPU in the local area is not capable of meeting that demand, which is why the project requires

the use of GPU power. In addition to saving a local copy of the data, the cloud is needed in order to back up the data.

A GPU from Google Colaboratory Machine learning and deep learning models, as well as numerous datasets, allow for the use of GPU-free to run Python code, build, and train deep learning models at a high speed, as opposed to laptops and basic PCs. These are the wanted project environments so that work on the project can begin.

3.2. Methods

An overview of the proposed solution methodology, which includes approaches and algorithms, will be provided in this section.

3.2.1. System architecture Overview

Recurrent neural networks (RNN) is the algorithm of the art for sequential data which are used by Apple's Siri as well as Google's voice search. That is, because this is the first algorithm which uses internal memory, it's perfectly suited for problems that deal with sequential data. More formally: Deep learning algorithms can be credited with making significant accomplishments over the recent years. An RNN is a powerful and robust and is among neural network algorithms, and is currently in use because it has an internal memory.

Like several other deep learning algorithms, recurrent neural network has existed for quite some time. Initially, they were created in the 1980s, but only recently have we've realized their full potential. Today's advanced computing capabilities with the large amounts of data we now at our disposal, and the invention of long-short term memory (LSTM) in the 1990s, have placed RNNs at the forefront. A very accurate feature of RNN's is that they possess an internal memory, which allows them to return outcomes that are highly correlated with input. This is why sequential data such as time series, text, audio, visual data, weather, video, and more are preferred. With recurrent neural networks, understanding a recurring sequence and its context will be significantly easier than with other techniques.

There are a number of important applications for RNNs, but when to use one depends on the use scenario. Whenever the sequential connection between items is more significant than the positions of each element in the temporal data series. Since

recurrent neural networks are being used in software such places like Siri and Google Translate, you can see them everywhere.

It's important to have a fundamental understanding of “traditional” feed-forward neural networks, along with sequential data, in order to grasp RNNs sequential data refers to facts that are arranged in a specific order Examples can be used to illustrate data on financial data or a DNA sequence. Sequential data is perhaps the most commonly known as a sequence of data points, like time series.

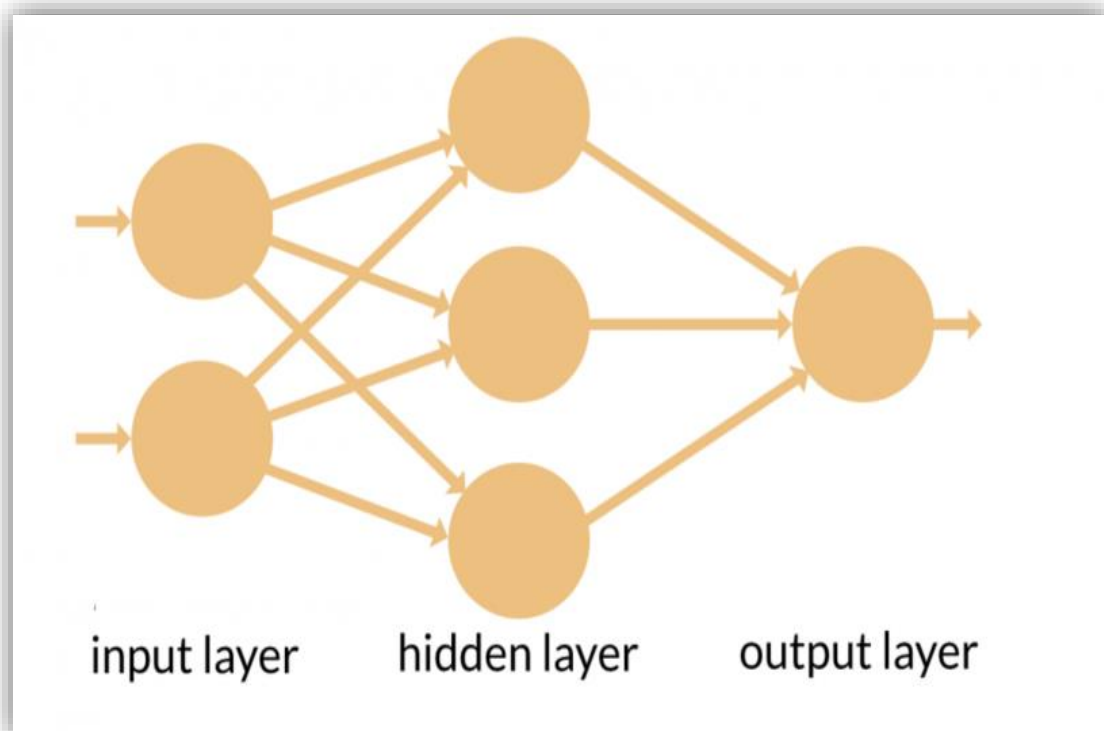


Figure 27:Feed-Forward Neural Network VS Recurrent Neural Network.

A feed-forward and recurrent neural network are named for the fact that they both use forward and feedback neural pathways to process information. In a feed-forward neural network, the data always flows from the input to the hidden layers and then out. information is disseminated from one node to the next. Feed-forward neural networks have really no memory of the input and are inefficient in predicting the output as shown in Figure 27:Feed-Forward Neural Network VS Recurrent Neural Network.. It has no notion of time order because a feed-forward networks deal with only the current inputs. It doesn't retain any other information son it can't remember or get back to information that occurred in the past expect its training.

The information goes around in a circle in a recurrent neural network. when it is making a decision, it considers input at the moment as well as previously given input. According to, Figure 28: RNN and Feed-Forward Neural network architecture. on the left illustrate the variance in information flow in a feed-forward neural network and a recurrent neural network.

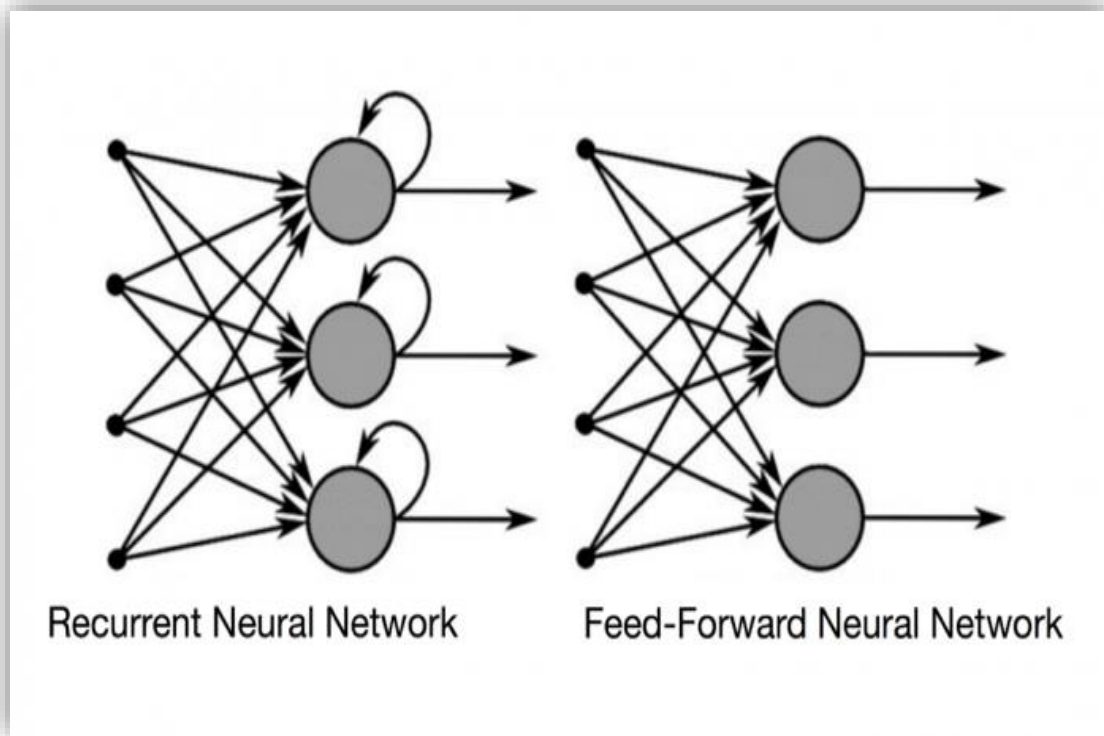


Figure 28: RNN and Feed-Forward Neural network architecture.

A simple recurrent neural network commonly has a short-used memory. They are always working together with a LSTM, but have a longer-term memory as well. For a better understanding of the idea of a recurrent neural network, try visualizing it with an example: assume that you have a feed-forward neural network and feed the characters into it one word at a time and the word of for this example is "neuron". When this sequence reaches the character "r," it has mostly forgotten about "n," "e," and "u," making this type of neural network useless in predicting which character would be next. A neural network of this type has the ability to remember characters, but is able to store more detail due to its internal memory. We will produce a copy of the algorithm, and loop it back into the network. More or less, it can be summarized as: New brain cell pathways store the recent past into the recent memory. This is why it is an RNN: it has two inputs, one of which is the most recent data. RNNs need to change because the sequence of data needs to reveal information about what will

come next. Similarly, all other feed-forward neural network algorithms assign weight matrices to their input layers. In an RNN, both the current and previous input's weights are factored in to compute the updated input. FDS takes the above into consideration, the weights are also adjusted by a recurrent neural network's descent and back propagated propagation through time (BPTT). Accordingly, to Figure 29: Architectural Types of RNN., while feed-forward networks will map one input to one output, RNNs can map many to one (classifying a voice), many to many which is the translation, and etc.

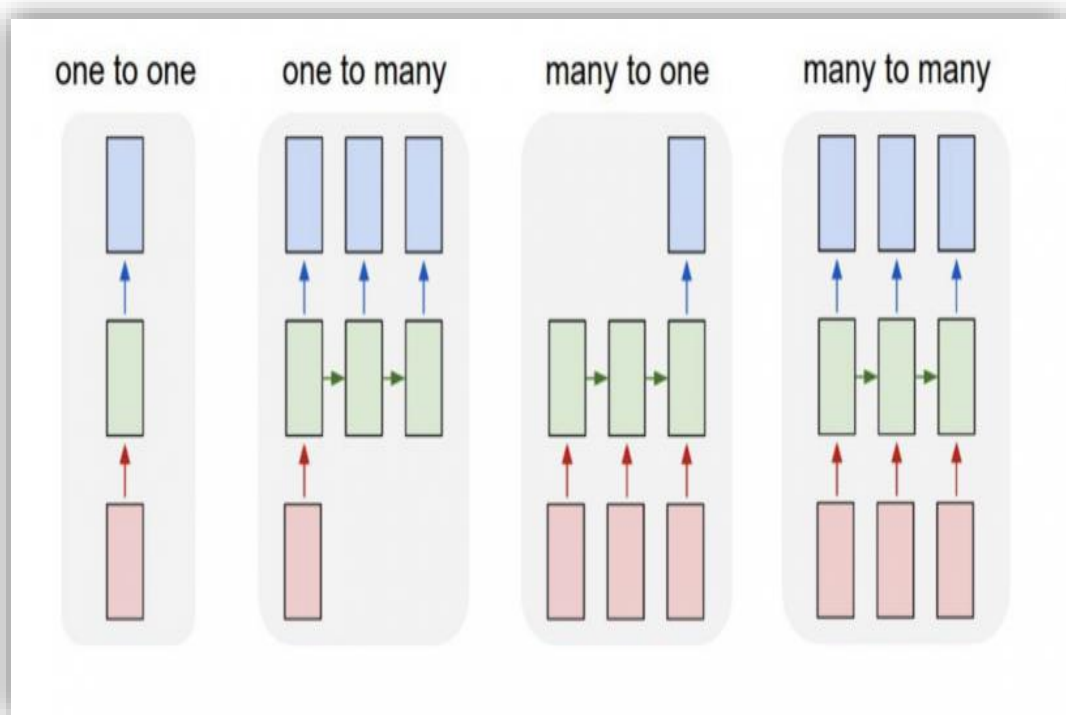


Figure 29: Architectural Types of RNN.

Formally speaking, in neural networks, you are testing the outcome of your model's predictions using forward propagation to determine if the prediction is correct or incorrect. Back propagation is finding a weight reduction that corresponds to a part of the process derivative of your neural network. We have other derivatives iteratively computed and then iteratively minimized by gradient descent gets rid of the function. Then it increases and decreases the weights according to which decreases the error. That is the process by which a neural network work during training. Basically, with back - propagation you tweak the weights while training as shown in

Figure 30: This illustrates the idea of forward and backward propagation in a feed-forward network..

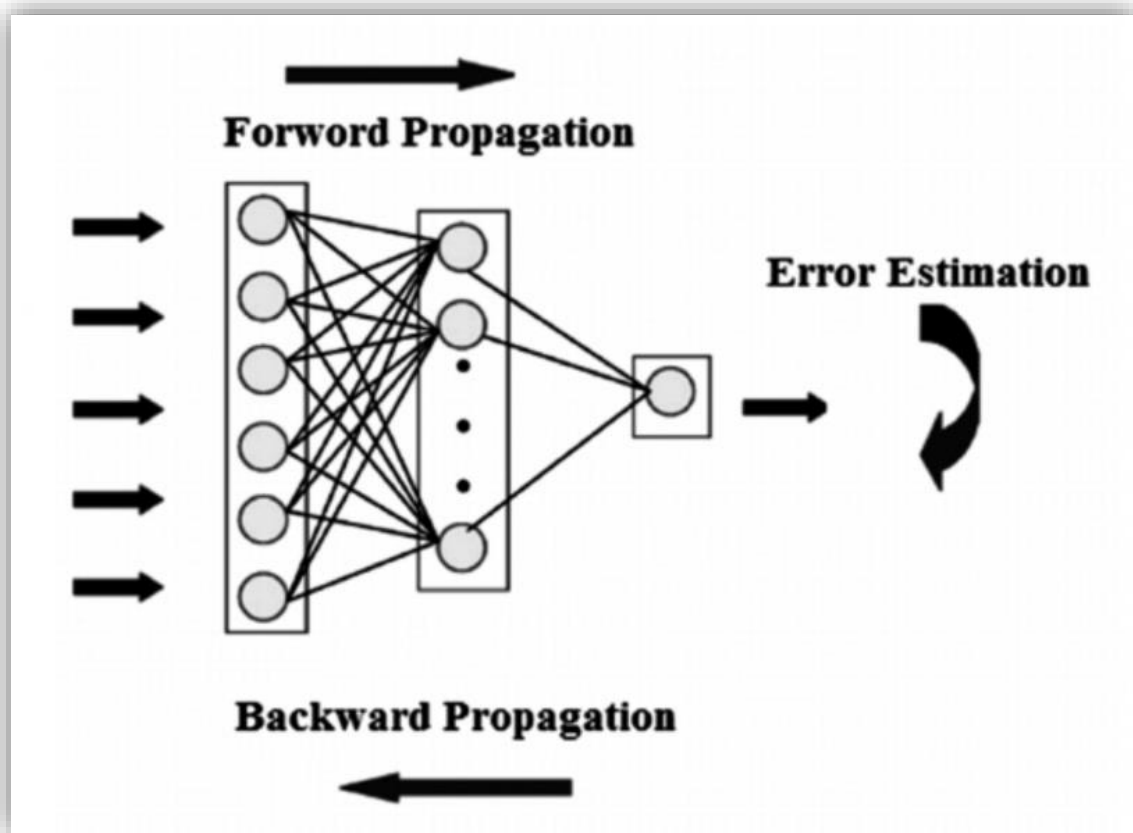


Figure 30: This illustrates the idea of forward and backward propagation in a feed-forward network.

The long-term (LSTM) and recurrent neural networks (RNN) are like other LTSMs, which further increase their storage capacity. gathering data over a long time period of time gives you more insight and time to consider the results. The building blocks of an LSTM network are often called LTSMs network. The ability of RNNs to remember inputs over such a long period of time is provided by LTSMs. LTSMs work like computers because they store information in memory. The LSTM can contains delete, write and read data from its memory. This memory may be likened to a cell door, with a gate blocking any process that does not occur in it, the gate decides whether or not to save or remove the information (as long as the gates are opened), based on how much importance it attaches to the information. The importance of various items can be obtained by using weights, which are, again, learnt by the algorithm. This is just another way of saying that it remembers what you need to know and forgets what you don't[48].

For an LSTM, you have three thresholds: an input gate, forget gate, and an output gate. These gates let or prevent new input from getting into the computer system (Input gate), eliminate/delete the information because it's unimportant (forget gate), or enable it to affect the output at the existing timing point (output gate) as shown in Figure 31: A visual illustration of an RNN, with its three input gates..

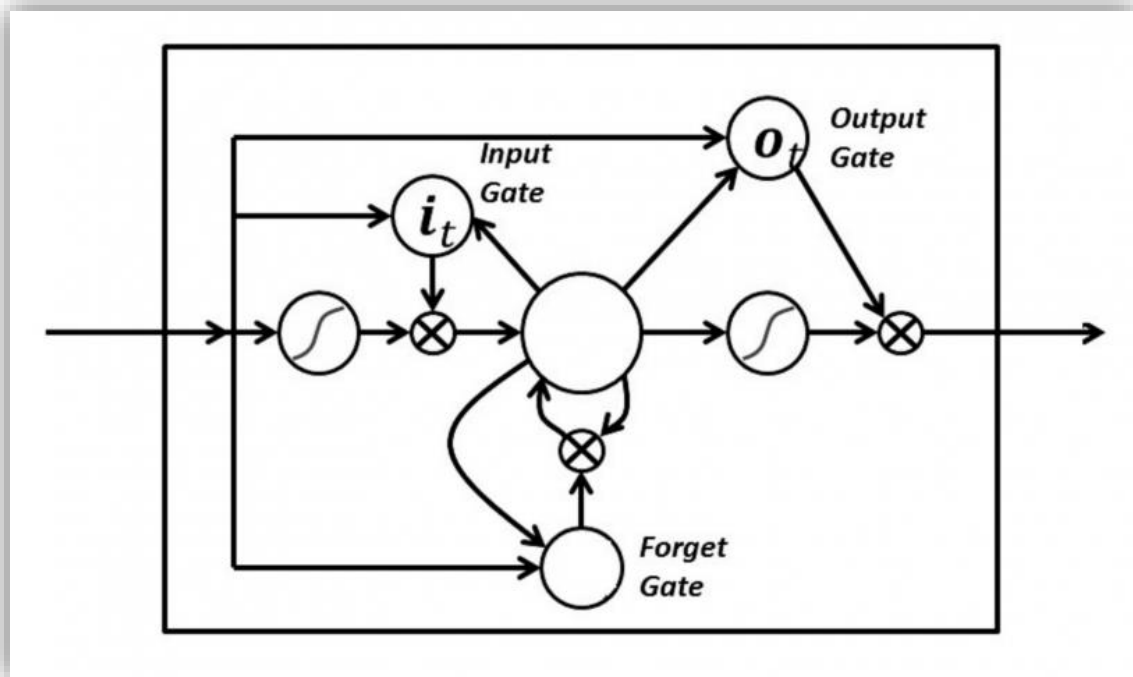


Figure 31: A visual illustration of an RNN, with its three input gates.

The gates inside an LSTM are expressed as sigmoids, which range from 0 to 1. Because they are analogue, they are able to do backpropagation. LSTM solves the vanishing gradients issue because of the learning rate is high, thusly shortening the training time while keeping the accuracy high[55].

Vanishing gradients occur when the values of a gradient are too small and the model stops learning or takes way too long as a result [55]. That it was a big issue in the 1990s, and more difficult to manage than gradients which would explode[60]. Luckily, the problem was solved by LSTM with Sepp Hochreiter and Schuetgen [57].

Chapter 4: System Implementation

4.1. Introduction

In this chapter, all of the system's implementation details will be discussed. The knowledge about both the system and how it was built will be described in detail in the first section, starting with a thorough description of the technical specifics and the considerations taken to reach the goal of this system. The second section will provide a high-level overview of the system structure and then move on to give more detail on each of its individual components. We will show the input and output of every system's component in the last section.

4.2. System Development

4.2.1. Building an experimental model to obtain a final model

In our first stage, we were unable to run the code. In our first perspective, it should have been related to some bug that we had created in the code but by the time we realized the problem and it was due to not having a high-performance computer. After that, we made sure and then made a decision to switch from our PC to Google Colab which has higher GPU and RAM performance and can have effective memory to debug our code.

We built an architectural model of a recurrent neural network. We build a demo with 100 proteins and we've estimated the dataset at 20% testing, 20% validations, and 60% training. The result was good enough on the model we built when we get the result after training and testing. We increased the dataset to 2000 proteins in the same way as the estimate for the dataset was 20% test, 20% validation, and 60% training. It also has good enough results for both training and testing that we can go to the next step without any fear that our model has wrong in any point of the architecture. Then, we started feeding more data into the architecture model to do complete training and testing by using the whole dataset that We got. The model is built by the PyTorch environment.

4.3. System Structure

This section is divided into two subsections. The overview of the whole system and the flow between its components will be illustrated in the introductory

section. The second subsection will provide and explain the Tensor-board graph and of the systems that include their architecture and functions.

4.3.1. System Overview

4.3.1.1. Data preparation

Because the topic we are looking for is rare and it is also very new. We did not find a large number of adaptors and non-adaptor proteins to make our model train as closely as possible to obtain better results. We found a manually generated dataset but we added more proteins to make the dataset as large as possible. We constructed our dataset using the same annotated dataset steps we found before. In fact, all of those who worked on this topic followed the same steps, and the tools that collecting the datasets are the same. Also, the sources that they are collecting from are the same. One of the most important resources is UniProt. UniProt is a free to access database of sequence protein and genomic properties. It contains a high percentage of genome sequence entries are derived from projects like this. This resource contains a lot of information that talking about the structural or biochemical role of proteins derived from scientific literature as shown in Figure 32: UniProt website..

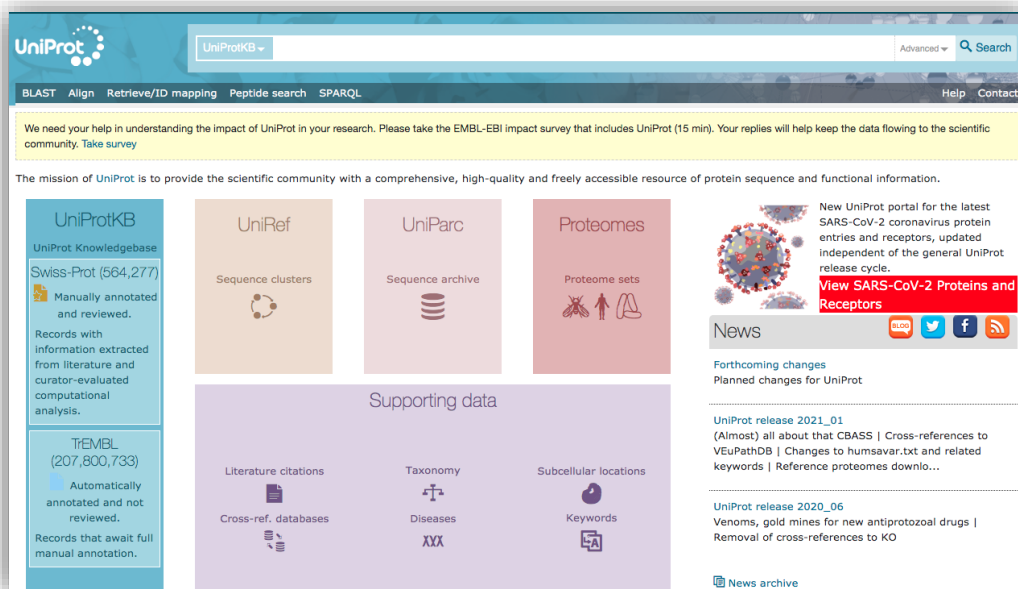


Figure 32: UniProt website.

It is possible to use FASTA as a text-format to represent whether in nucleotide sequences or peptide sequences, which represent base pairs or amino acids by a single-coded letter. A sequence will begin with a single line of text followed by lines of data in FASTA format as shown in Figure 33 and Figure 34.

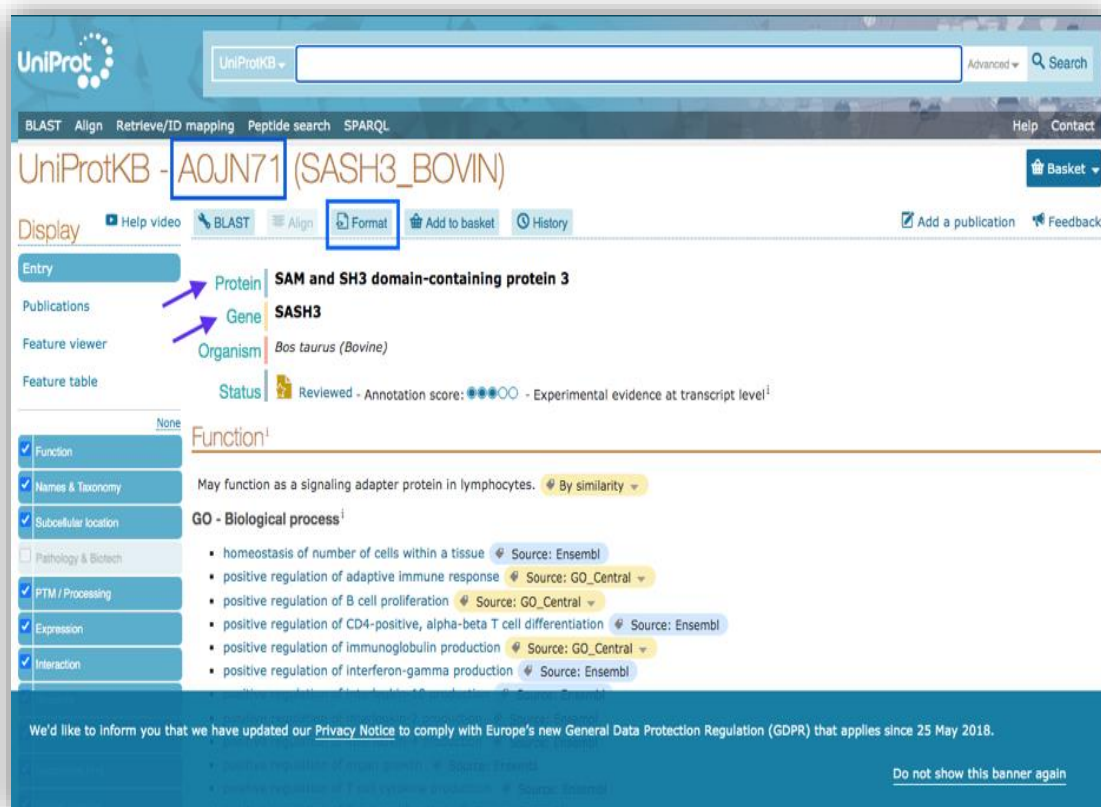


Figure 33: Illustration to obtain the protein sequence.

```
>sp|A0JN71|SASH3_BOVIN SAM and SH3 domain-containing protein 3 OS=Bos taurus OX=9913 GN=SASH3 PE=2 SV=2
MLRRKPSNASEKEPTQKKKLSLQSSSFKDFAKSKPSSPVVSEKEFNLDNIPEDSSVP
TPEDAEGSGKKLGKKWRAVISRTMNRKTGKKMKVKSSEMGDTLEEGSASPTSPDCSLDS
PGPEKMAAFSEQEERELPALSRQASTGSELCSFSPGSGNLGEESTAPQYTGPFCCRARV
HTDFTSPYDRDSLKLQGDVIQVEKPPVGTWGLLNGRMGSFKFIYDVLPEEAVGPA
RPSRRQSKGRPKPKTLHELLERIGLEEHTSTILLNGYQTLEDFKELRETHLNELNIMDP
QHRAKLLTAAELLLDYDTGSEEAEGTESGQEPAVSTVADPKVDIPRDSGCFEGSESGRD
EAELAGTEEQHLGLSLGAP
```

Figure 34: FASTA format for A0JN71.

We gathered all the UniProt GO molecular functions associated with adaptor proteins. It was essential that the reviewed sequences be published in

scientific journals Thus, the whole query to obtain the data as shown in Figure 35: Query of getting data..

```
"keyword:"adaptor" OR goa:"adaptor")) AND reviewed:yes"
```

Figure 35: Query of getting data.

In reality, the non-adaptor proteins can reach hundred thousands of proteins because it is normal to find a little number of proteins that causes of diseases to the body. But we have a serious problem that there are two different amounts of data sets that are not equal to each other, and one of them at least doubles the first amount. Sure, this issue will affect our performance model. Most of the problems with bioinformatics involve searching for non-adoppter protein and then treating them as an adaptor. And after we'd obtained a pool data set of membrane proteins, we selected a common protein, which had enough sequences and functions to meet our needs. In bioinformatics problems related to protein sequence, they faced the redundancy of these sequences. For more details, Redundancy of a sequence data is present when one or greater homologous or similar sequences are found in the same set. The implementation of similar sequences will introduce bias into certain analyses. We used the Basic Local Alignment Search Tool (BLAST) to solve our problem. This tool is used to find similar regions in protein or nucleotide sequences. It compiles the nucleotide or protein sequences in a database and computes the statistical significance of their correspondence. We treated the input of the BLAST tool as the protein sequence we got from UniProt and the output will definitely be reduced if the similarity is less than what we have defined to create our dataset without duplication as shown in Figure 36: Illustration of Basic Local Alignment Search Tool.. All of the previous studies that talking about our topic determine 30 % similarity for each sequence and it wouldn't be greater than the defined percentage in the BLAST tool. All previous studies that talking about our topic specify a 30% similarity for each sequence and will not be greater than the percentage specified in the BLAST tool.

BLASTP programs search protein databases using a protein query. [more...](#) [Reset page](#) [Bookmark](#)

Enter Query Sequence

Enter accession number(s), gi(s), or FASTA sequence(s) [Clear](#) [Query subrange](#)

From To

Or, upload file No file chosen [?](#)

Job Title

Enter a descriptive title for your BLAST search [?](#)

☐ Align two or more sequences [?](#)

Choose Search Set

Database ?

Organism Optional ☐ exclude [Add organism](#)

Enter organism common name, binomial, or tax id. Only 20 top taxa will be shown. [?](#)

Exclude Optional ☐ Models (XMI/XP) ☐ Non-redundant RefSeq proteins (WP) ☐ Uncultured/environmental sample sequences

Program Selection

Algorithm

- ☐ Quick BLASTP (Accelerated protein-protein BLAST)
- ☒ blastp (protein-protein BLAST)
- ☐ PSI-BLAST (Position-Specific Iterated BLAST)
- ☐ PHI-BLAST (Pattern Hit Initiated BLAST)
- ☐ DELTA-BLAST (Domain Enhanced Lookup Time Accelerated BLAST)

Choose a BLAST algorithm [?](#)

[BLAST](#) Search database nr using Blastp (protein-protein BLAST)

☐ Show results in a new window

New columns added to the Description Table

Click 'Select Columns' or 'Manage Columns'.




Figure 36: Illustration of Basic Local Alignment Search Tool.

One of the most important things in deploying a PSSM is avoiding information loss of connection by keeping ordering of the PSSM. Distantly related proteins were first discovered with the assistance of the Position-Specific Scoring Matrix (PSSM). It was aligned from a series of sequences by implies of sequence similarity or structural [40]. PSI-BLAS is the most often used application, which analyses PSSM profiles to detect similar proteins or similar DNA remotely [41][40]. The PSSM is made up of four components. Firstly, the position that indicates each amino acid in the index of which has been sequentially increased after a given of sequence alignment. Secondly, Probe, this represents a collection of typical sets of functionally related proteins which have been lined up sequentially by structural or sequence similarity. Thirdly, Profile is the protein data matrix consists of 20 columns, each corresponding to a single amino acid. Fourthly, the consensus, which is made up of the most identical amino acid residues at each position in the alignment of residues. It is derived from having the highest total score for each position of the profile.

The matrix $N \times 20$ is a query protein for PSSM. So, $P = \{P_{ij} : j = 1 \dots 20 \text{ and } i = 1 \dots N\}$ given as N is the number of amino acids of the protein sequence as shown in Figure 37. In order to each position of the query sequence, it generates a score P_{ij} for the j th amino acid with a large value representing a highly conserved nucleotide residue and a low value that reflects a poorly conserved position [42]. While the structures of PSSM differ slightly through one application to some other, the concepts are mostly identical.

		sequence positions							
		1	2	3	4	5	6	7	8
amino acids	A			-2.4					
	R			1.2					
	D			0.5					...
	N			-0.2					
	C			-3.1					
		...							

Figure 37: Illustration of PSSM.

Here, we are talking about the approach utilized throughout the original of PSSM work [31], where the P_{ij} (position specific score) is considered to be as shown in Figure 38: Equation of PSSM., where $w(i,k)$ is the ratio of the appearance of the amino k th acid frequencies (on the set of twenty amino acids) at this i location of the probe and also the total amount of probes, and $Y(j,k)$ is Dayhoff's mutation matrix between some of the k th and j th amino acid has been calculated. In our research, we developed PSSMs using PSI-BLAST application for each sequence of protein.

$$P_{ij} = \sum_{k=1}^{20} w(i, k) \times \bar{Y}(j, k)$$

Figure 38: Equation of PSSM.

Several researchers have seen improvements in bioinformatics applications had when PSSM was first proposed [46]. The benchmark dataset contains sequences of newly acquired proteins in FASTA format. We developed our FASTA sequence set from the use of PSSM features, which we looked up in the non-redundant (NR) database.

Some research studies predicted the protein functions by taking all the same amino acids into consideration[27]. It can convert PSSM matrix profiles from $20 \times N$ to matrix 20×20 , and all of every one of the sequences were of equal length that can be used in supervised learning classification. Critical information could have been irretrievably be discarded because the ordering with PSSM profiles is up to the designer. As a result, an RNN architecture was proposed that could process PSSM and could keep the profiles ordered. PSSM sequences of different in length can be accepted by the implemented RNN network, as a result, we were able to get more useful information out of their spatial context for protein function prediction. After generating our PSSM file from the protein sequence using the path that shown in Figure 39: PSSM File Creation Using PSI-BLAST., we can now say that we are ready to extract the features.

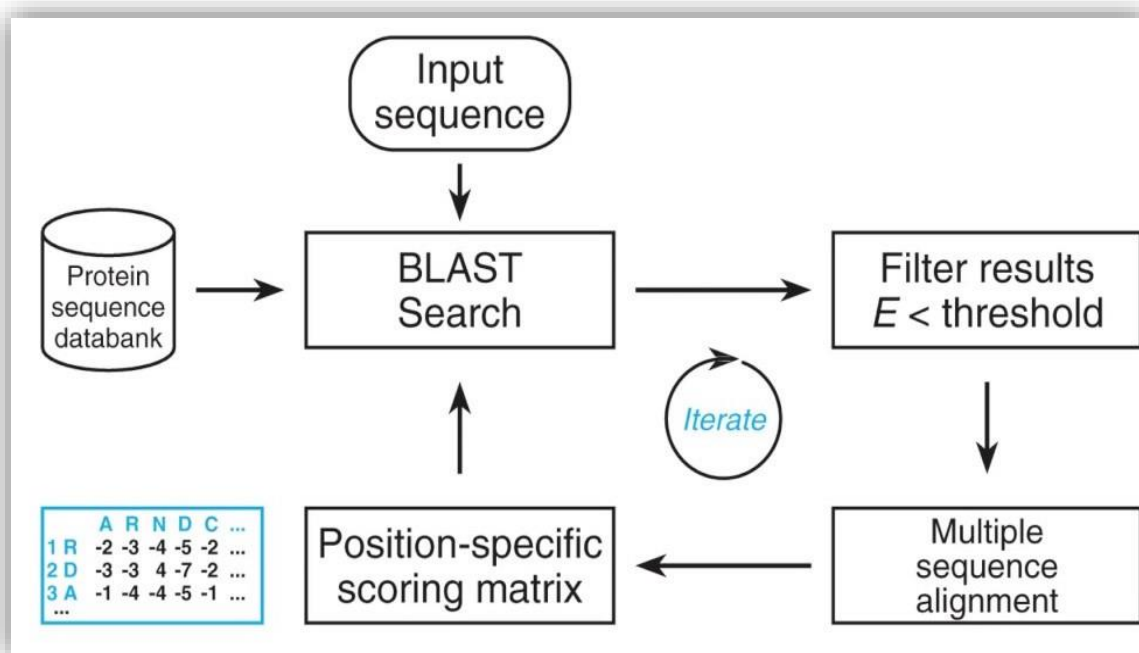


Figure 39: PSSM File Creation Using PSI-BLAST.

4.3.1.2. Classification

After that, the proposed RNN model is computed using a 1D CNN as a convolution layer to extract the feature maps from PSSM features. According to Figure 40: Feature extraction via CNN., The CNN contains two 1D convolution layers, supported by only a Rectified Linear Unit (ReLU) as well as two Average Pooling layers to improve feature reconstruction and increase the receptive field of the network. Finally, this same extracted feature maps were also fed to a RNN module for exploration of the spatial relationships throughout the PSSM data before making a final prediction.

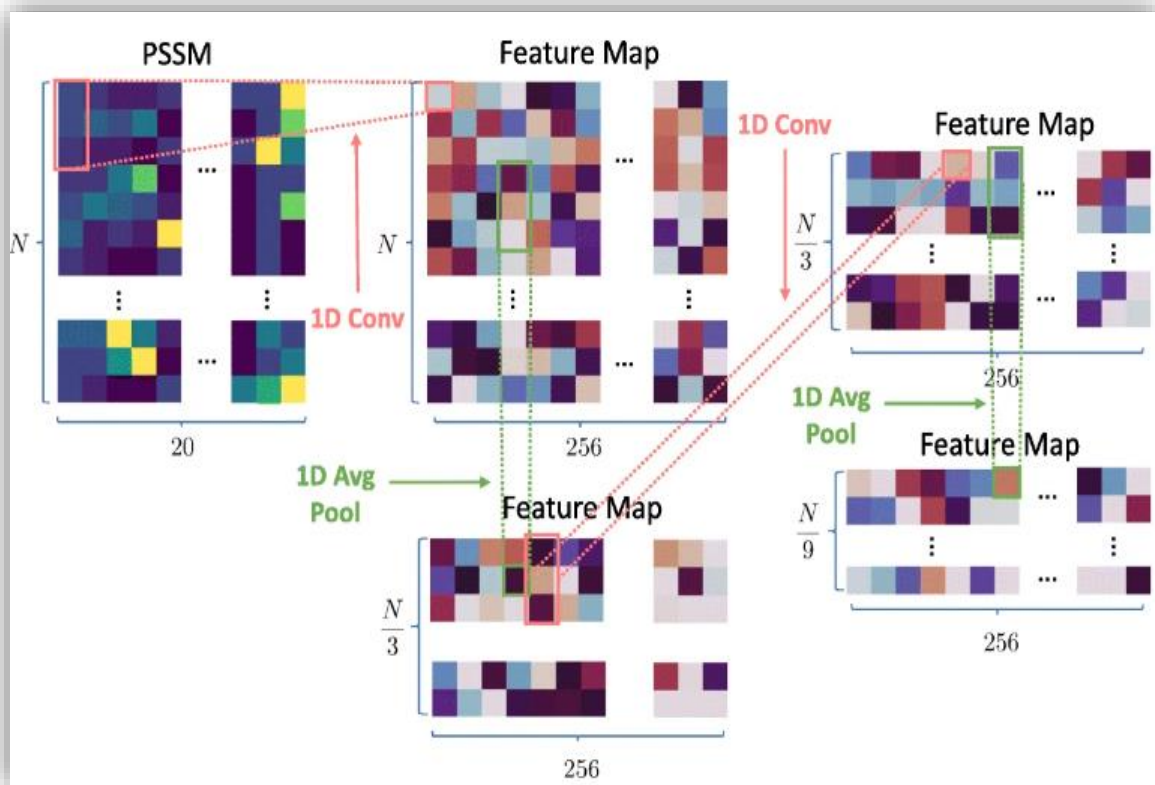


Figure 40: Feature extraction via CNN.

Several empirical studies have demonstrated that RNNs can perform well in the time series prediction task such as and language model, and speech recognition [7]. Since RNN can remember sequential data, we utilized an RNN with an advanced network architecture for this research which is the GRU.

To accomplish the aforementioned CNN results, we applied a multi-layer GRU on top of the feature maps extracted from the CNN algorithm. The standard RNN has a notable disadvantage called the gradient "vanishing" problem, which causes the network to lose a lot of information in the middle of the sequence and makes it prone to predicting using only the most recent information. Therefore, the recurrent units, like the GRU and LSTM, were thus developed and investigated to increase their capabilities to capture longer-term memory storage.

GRU is an advanced version of a standard RNN, for which the gradient vanishing problem is resolved by an update gate as well as a reset gate to eliminate what is useless information. GRU's capability allows for long linkages between both the current input and distant location information.

The structure of the GRU is the same as LSTM. GRU has fewer parameters, thus making it the perfect for small datasets. This will make the procedure easier and encourage us to use GRU as the base unit throughout our RNN module.

A Gated recurrent unit (GRU), as defined, is a recurrent network design that uses gating devices to regulate and manage the traffic between cells. Since the GRU incorporates massive amounts of information from earlier parts of the sequence, it can capture dependencies as well as synthesis them. This is done using its gating units, just like in LSTMs, which resolve the problem of traditional RNNs' gradient vanishing and explosion. Each of these gates has the task of regulating what to be kept or discarded as a time.

The GRU cell's memory storage technology enables it to retain long-term dependencies or memories. Accordingly, to Figure 41: GRU Architecture., Only two gates exist in the GRU cell: An Update and a Reset gate. just like the LSTMs, these gates throughout the GRU only permit information that is relevant to the task. These vectors contain values in the range of 0 to 1 will be multiplied also with input data, which results in the outputs having values between 0 and 1. A "zero" value in the gate input or

hidden states implies that the corresponding data is meaningless and will therefore, therefore, will return "zero" value. Conversely, the 1 value throughout the gate vector indicates that the relevant data will be used.

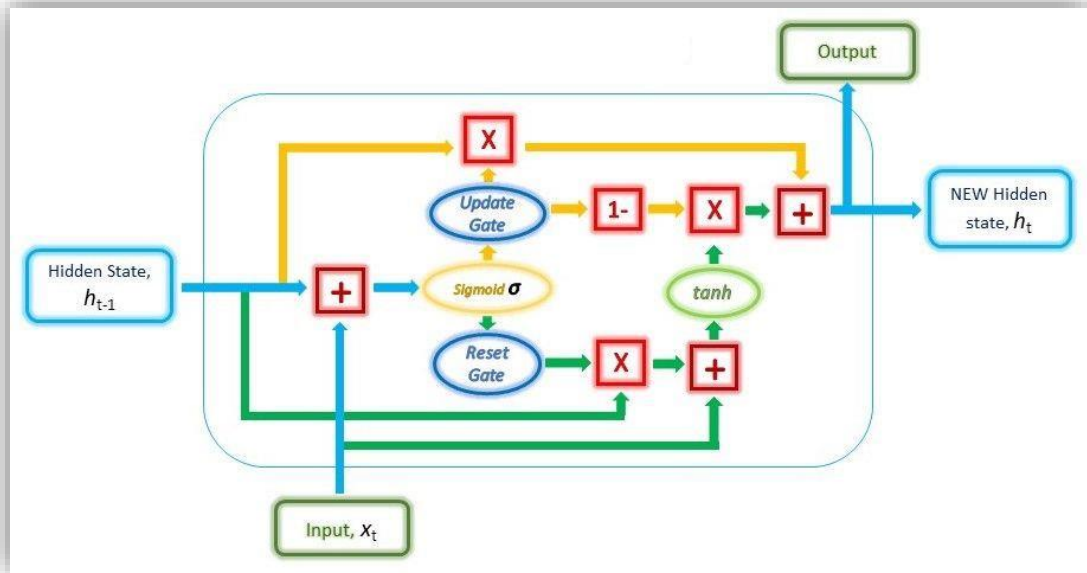


Figure 41: GRU Architecture.

The first phase will be to create a Reset gate. The latest state's gate is calculated based on the previous hidden data as well as the current data. Accordingly, to Figure 42, It is obtained by previous hidden state weighting the current input and the result, and summing them, and multiplying that value with current hidden state prior to applying a sigmoid function.

$$gate_{reset} = \sigma(W_{input_{reset}} \cdot x_t + W_{hidden_{reset}} \cdot h_{t-1})$$

Figure 42: Equation of Reset Gate.

The sigmoid function would then produce values which are bounded between 0 and 1, Letting less important knowledge come through the gate as far as it needs to. When the entire neural network has been trained with back-propagation, the weight values would be updated such that only the useful features will be remembered. The hidden state of the previous method will be multiplied by a first trainable weight and then reset vectorized element-wise. This operation is going to determine which information stays and which is

discarded. Additionally, the current input would be multiplied by a trainable weight until being added to the product of the previous hidden state and the reset vector. a non-linear activation tanh will be used on the last step in order to obtain the formula as shown in Figure 43.

$$r = \tanh(\text{gate}_{reset} \odot (W_{h_1} \cdot h_{t-1}) + W_{x_1} \cdot x_t)$$

Figure 43: A non-linear activation tanh function.

After that, we'll have to create the Gate process. The gate is derived from the previous hidden state as well as the current input data, just like the Reset gate. However, the finished vectors for each gate are generated differently using the formula, because the final weight is multiplied with the input weight before being added to the hidden state, which means they are distinct as shown in Figure 44. With this design, the gates serve their respective purposes.

$$\text{gate}_{update} = \sigma(W_{input_{update}} \cdot x_t + W_{hidden_{update}} \cdot h_{t-1})$$

Figure 44: Equation of GRU gate update.

Element-wise multiplication will then be performed with the previous hidden state to achieve our output as shown in Figure 45.

$$u = \text{gate}_{update} \odot h_{t-1}$$

Figure 45: GRU Equation.

Also, in our final output procedure, the Update vector would be used. This Update gate is used to calculate how much of the stored past information

from the hidden state needs to also be retained for the model to make a prediction.

So, we will use an RNN model in this study to differentiate adaptor proteins from non-adaptor proteins. This is shown in Figure 46 and Figure 47 an overview of the implemented RNN model. The RNN model receives PSSM features as input from multiple 1-convolution (1-D) and one-dimensional average pooling layers and computes their features. The extracted features are then fed into GRUs, where the spatial context of the entire PSSM approach is explored and employed to make the final prediction. N represents the number of values is given as input sequence. The final pass through two layers of one-dimensional CNN and 1D one-dimensional average pool only reduced the length by $N/9$. As a result, this $N/9$ vector was sent to the GRU algorithm to fed, transferring features from the last GRU (i.e. 256 features) to the beginning of the sequence (i.e. to the input of the last sequence that acquired a feature). At long last, we computed the whole output through a Fully Connected (FC) layer (512 nodes), after which it was passed to a Sigmoid layer, and this value was finally obtained as a prediction.

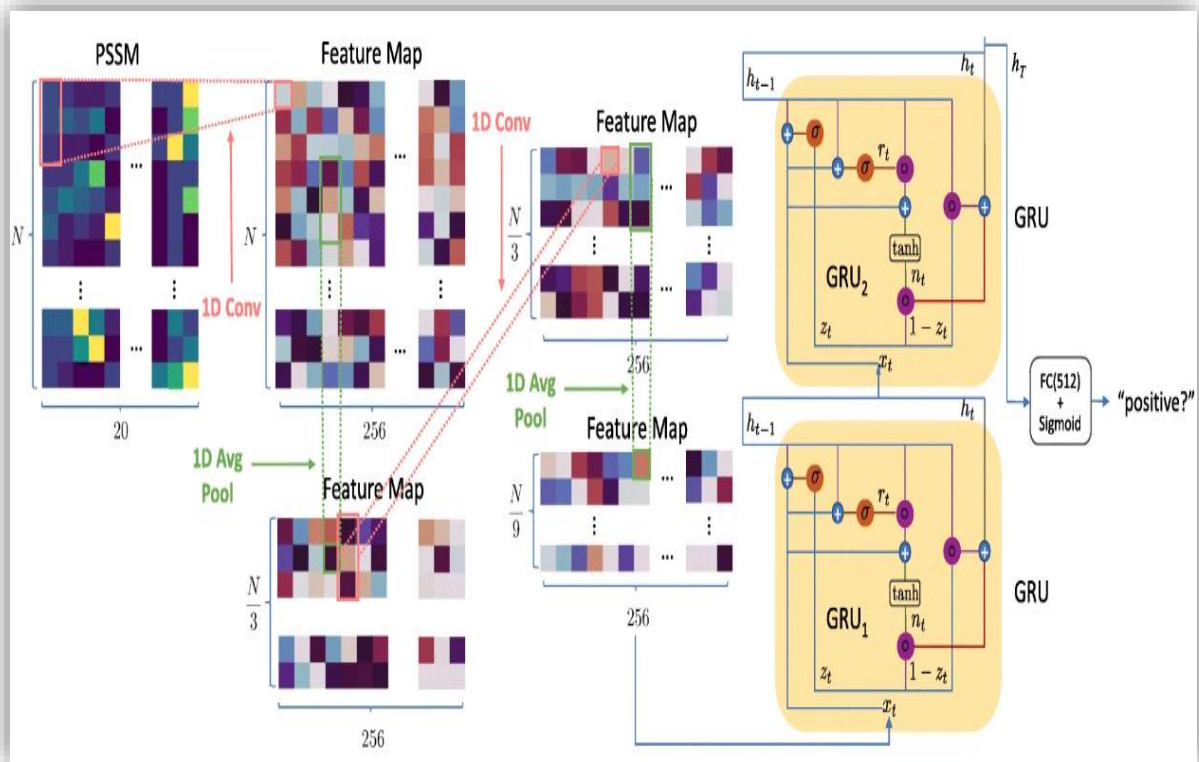


Figure 46: RNN model Architecture for predicting protein adaptors.

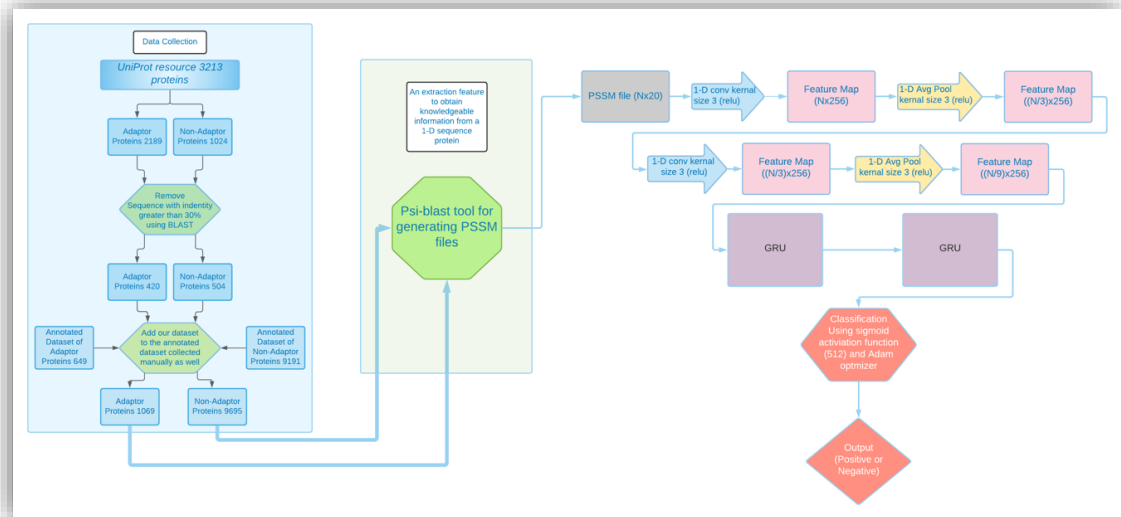


Figure 47: A working flow for our model.

4.4. System Running

This next section will demonstrate the input and output of each system's individual component and its final product.

4.4.1. Component A

The input of the entire model is a protein sequence that after we insert it into the PSSM function it actually transforms into the matrix $20 \times M$. So, 20 is the number of amino acids most likely to be used. N is the number of characters in the query sequence. In fact, our component that takes the input is the PSSM function to transfer it from 1 dimensional (protein sequence) to matrix $20 \times M$ (length of the sequence) and here is an example of our input as shown in Figure 48.

Last position-specific scoring matrix computed, weighted observed percentages rounded down, information per position, and relative weight of gapless real matches																									
	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V					
1 M	-1	-2	-3	-4	-2	-1	-3	-3	-2	1	2	-2	7	0	0	0	-3	-2	-1	-2	-1	0	0.66	0.08	
2 E	-1	0	0	2	-4	2	6	-3	0	-4	-3	0	-2	-4	-2	0	-1	-3	-3	-3	0	0	0.85	0.06	
3 V	0	-3	-3	-4	-1	-3	-3	-3	-4	2	1	-3	0	-1	-3	-2	0	-3	-2	5	4	0	0.96	0.45	0.05
4 N	-2	-1	7	1	-3	0	-1	-1	0	0	0	0	-3	-4	-2	0	0	-4	-3	-3	0	0	0.90	0.07	
5 I	-2	-3	-1	-3	-2	-1	-3	-4	-3	5	1	-2	1	-1	-3	-2	-1	-3	-2	2	0	0	0.37	0.05	
6 L	-1	-2	-3	-4	-2	-2	-3	-4	-3	1	4	3	2	1	-3	-2	-1	-2	-1	1	2	0	0.41	0.04	
7 A	-4	-2	-1	-2	-1	-1	-1	3	-2	-2	-2	-1	-1	-3	-1	1	-1	-3	-2	-1	69	0	0.43	0.03	
8 F	-2	-3	-3	-4	-2	-3	-4	-2	0	2	-3	0	6	-4	-3	-2	0	2	1	0	0	0	0.58	0.04	
9 I	-1	-3	-3	-3	-1	-3	-3	-4	-3	4	2	3	1	0	-3	-2	0	-3	-2	3	0	0	0.39	0.04	
10 A	5	-2	-2	-2	-1	-1	0	0	-2	-2	-1	-1	-3	-1	1	0	-3	-2	0	97	0	0.50	0.05		
11 T	0	-2	-1	-2	-1	-1	-2	-2	-2	1	-1	-1	-2	-2	5	-3	-2	1	0	0	0	0	0.44	0.05	
12 A	5	-2	-2	-2	-1	-1	0	0	-2	-1	-2	-1	-1	-3	-1	-1	-3	-2	0	94	0	0.47	0.05		
13 L	-2	-3	-4	-4	-2	-3	-4	-3	-2	0	3	5	2	0	-3	-3	-2	-1	1	0	0	0.56	0.04		
14 F	-2	-3	-4	-3	-3	-3	-2	0	0	1	3	0	7	0	-4	-2	1	3	-1	0	0	0.79	0.06		
15 I	-1	-3	-4	-4	-1	-3	-4	-4	5	2	-3	1	0	-3	-3	-1	-3	-2	3	2	0	0.46	0.04		
16 L	-1	-2	-2	-3	-1	-2	-2	-3	2	-2	1	0	0	0	-1	-2	-2	1	2	0	0	0.22	0.02		
17 I	-1	-3	-3	-4	-1	-3	-4	-4	3	1	-3	-2	-1	-3	-1	-4	-3	-3	0	0	0	0.39	0.03		
18 P	-1	-2	-2	-2	-3	-2	-1	-2	-3	-3	-1	-3	-4	8	0	-1	-4	-3	0	0	0	2.02	0.06		
19 T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-2	-1	-1	-3	1	2	5	-3	-2	0	0	0.60	0.06		
20 S	4	-2	-1	-1	-1	-1	0	0	-2	-1	-2	-1	-1	-3	-1	0	-3	-2	0	73	0	0.37	0.04		
21 F	-2	-3	-3	-4	-3	-3	-3	-2	0	1	-3	0	6	-4	-1	-2	0	3	-1	0	0	0.70	0.05		
22 L	-1	-2	-3	-4	-2	-2	-3	-4	-3	1	4	3	2	0	-1	-1	-2	-1	1	0	0	0.42	0.04		
23 L	-2	-3	-4	-4	-2	-2	-3	-4	-3	2	4	3	2	0	0	-3	-1	-2	-1	0	0	0.43	0.03		
24 I	-2	-3	-4	-4	-2	-3	-4	-4	-4	5	1	-3	1	1	-3	-1	-3	-1	3	0	0	0.47	0.05		
25 I	-2	-3	-3	-3	-2	-2	-3	-4	-3	3	3	3	1	0	1	-2	-1	-2	-2	1	0	0.34	0.03		
26 Y	-2	-2	-2	-4	-3	-2	-2	-4	2	-2	-1	-2	-1	3	-3	-2	-2	2	8	-2	0	1.06	0.07		
27 V	0	-3	-3	-4	-1	-3	-4	-4	3	1	-3	1	-1	-3	-2	0	-3	-2	5	0	0	0.93	0.46	0.05	
28 K	-1	2	0	-1	-3	3	1	-2	-1	-3	3	5	-1	-4	-1	0	-1	-3	-2	-3	0	0.56	0.04		
29 T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-2	-1	-1	-3	-1	1	5	-3	-2	0	0	0.60	0.06		
30 V	3	-2	-2	-2	-1	-1	0	-1	-2	1	0	-1	0	-2	-2	0	0	0	-3	2	43	0	0.18	0.03	
31 S	2	-1	0	0	-1	-1	0	-1	-2	-3	-1	-2	-3	-1	4	2	-3	-2	-2	12	0	0.43	0.05		
32 Q	-1	0	0	0	-3	5	3	0	0	-3	-3	1	-1	-4	-2	0	0	0	-3	-2	2	0	0.55	0.06	
33 N	0	-1	4	0	-2	1	1	1	0	-3	-3	0	-2	-3	-2	2	0	-3	-2	3	1	0	0.42	0.04	
34 N	-1	-1	5	3	-3	1	1	-1	0	-4	-4	1	-3	-4	-2	1	0	-4	-3	-3	0	0.54	0.05		

Figure 48: Illustration of one of the inputs PSSM file for specific protein.

Chapter 5: Results and Evaluation

In order to better understand the indoor environments presented in the previous chapter, discuss system details of the systems, some or all of the testing methodologies employed to examine the performance of the model will be explained in greater detail. Also, the results and findings of each system will be shown under different conditions in subsequent sections.

5.1 Testing Methodology

In the following segment, the used testing methodology of each system or algorithm will be detailed.

5.1.1. Data preparation

As we explained in the previous chapters, the dataset was collected manually because it is a new area that does not support all the datasets you want. After we got the annotated dataset, we actually want to scale up our dataset to feed the dataset as much as possible for our model to get better results. So, we created small PSSM files but wanted to make sure our steps were perfect and nothing wrong with our project. So, we already took a sample from the PSSM files that we got and we already know what the name of the protein is and also what the protein sequence associated with that protein is and whether it is negative or positive. In fact, it doesn't matter what it is because we apply the same techniques to both because PSI-BLAST doesn't care if it's negative or positive. It is only concerned with protein sequences and applies its math for generating the PSSM matrix. After taking this sample, we actually generated a PSSSM file related to this protein sample. We compared our sample result with the results for the data set that we sampled. The two matrices were equal. Therefore, we made sure that our steps were perfect and there was nothing wrong with our work. Then, as we described in the previous chapters, we have been increased these data sets as closely as possible manually. According to these numbers that shown in Figure 49 and Figure 50, we have ensured that the results of the matrix are perfect and there is no error between the annotated dataset and the dataset that we have created ourselves.

Last	position-specific	scoring	matrix	computed,	weighted	observed	percentages	rounded	down,	information	per	position,	and	relative	weight	of	gapless	real	matches																									
	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	V	Y	V	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	V	Y	V				
1 M	-2	-3	-4	-5	-3	-2	-3	-4	-3	0	1	-3	9	-1	-4	-3	-2	-3	-2	0	0	0	22	5	30	0	0	0	0	0	0	97	0	0	0	0	0	3	1.33	0.24				
2 T	-2	-3	-2	-3	-2	-3	-2	-3	-2	0	1	-3	9	-1	-4	-3	-2	-3	-2	0	0	0	22	5	30	0	0	0	0	0	0	97	0	0	0	0	0	2	1.33	0.24				
3 D	-2	-2	-4	-6	-4	-1	0	0	-2	-4	-4	-1	-3	-3	-1	-1	-4	-1	-4	-1	2	26	55	0	0	0	0	5	5	0	0	0	0	0	0	1	0	0	7	0.73	0.13			
4 L	-1	-3	-4	-4	-2	-3	-4	-3	-2	3	-3	2	5	-4	-2	-2	-1	1	1	1	3	0	0	0	0	0	0	0	0	0	10	37	0	6	34	0	0	10	0.44	0.09				
5 K	-1	-1	1	-1	-2	-3	1	0	-3	-2	-1	1	5	0	-2	-2	-1	2	-3	-2	1	0	0	0	0	0	4	1	0	0	1	16	57	1	2	0	0	16	0	0.35	0.11			
6 T	-1	0	-1	-2	-2	-1	-1	-2	-2	0	-2	2	-3	-3	-2	1	5	-4	-3	-1	0	2	0	0	0	0	0	0	2	0	6	0	0	0	0	1	69	0	0.54	0.14				
7 P	-1	-3	-4	-5	-3	-2	-3	-2	-3	-2	-3	-4	-3	-2	-3	-4	-3	-2	-3	-2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	0.58	0.30					
8 L	-2	-3	-4	-5	-2	-3	-4	-5	-4	1	5	-3	1	0	0	-3	-2	-3	-2	0	0	0	0	0	0	0	0	0	0	0	0	3	91	0	2	4	0	0	0	0.69	0.14			
9 S	0	-2	0	-1	-2	-1	-1	-1	-2	-3	-3	-1	-2	-3	0	5	1	-3	-3	-3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	87	5	3	0	0.72	0.17	
10 V	0	-2	-1	-2	-2	-2	-2	-3	0	-1	-2	-3	-3	-2	1	5	-3	-2	-2	8	0	0	0	0	0	0	0	0	0	0	0	2	2	0	0	2	69	0	15	0.53	0.14			
11 A	5	-2	-3	-3	-1	-2	-2	-1	-3	-1	-2	-2	-2	-3	-2	0	-1	-4	-3	0	91	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7	0.68	0.16		
12 P	-1	-4	-4	-5	-2	-3	-4	-4	-2	-3	-4	-3	-4	-3	-4	-3	-4	-3	-4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0.47	0.09		
13 V	-1	-4	-4	-4	-2	-3	-4	-4	-2	0	-3	-3	-2	-3	-1	-4	-2	-6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	96	0.80	0.17		
14 V	-1	-3	-4	-4	-2	-3	-3	-4	-4	3	4	-3	2	0	-3	-3	-1	-3	-2	2	3	0	0	0	0	0	0	0	0	0	0	23	54	0	3	2	0	0	0	15	0.44	0.08		
15 S	4	-2	-1	-2	-1	-1	-1	1	-2	-1	-1	-1	-1	-1	-2	2	1	-3	-2	1	55	0	0	0	0	0	0	0	6	0	2	7	0	0	2	23	5	0	0	0.35	0.10			
16 T	1	-2	-2	-2	-2	-2	-2	-3	1	0	-2	1	0	-2	1	4	-3	-2	0	1	4	0	0	0	0	0	0	0	0	0	0	13	9	0	5	3	0	0	53	0	2	0.33	0.10	
17 O	0	-3	-3	-3	-2	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	0	0	0	0	0	0	0	0	0	0	0	0	26	42	0	0	0	0	1	7	0	0.7	0.1		
18 W	-3	-1	-4	-5	-2	-3	-4	-4	-3	-4	-3	-4	-3	-3	-1	-4	-1	-2	-12	-1	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	75	2	0.21	0.24	
19 F	-1	-3	-4	-4	-1	-3	-4	-4	-2	1	2	-3	3	6	-4	-3	-2	-1	1	0	3	0	0	0	0	0	0	0	0	0	0	4	23	0	10	51	0	0	0	4	0.52	0.11		
20 G	0	-2	-1	-2	-2	-2	-2	-4	-2	0	-1	-2	-1	-1	-2	1	2	-3	-2	-1	5	0	0	0	0	0	0	35	0	0	8	6	0	0	2	12	24	0	0	2	0.27	0.07		
21 A	0	-3	-2	-3	-1	-2	-3	-3	-2	1	1	-2	0	4	-3	1	-1	-1	1	0	9	0	0	0	1	0	0	0	0	0	0	11	15	0	36	17	1	0	0	6	0.23	0.07		
22 L	-1	-2	-2	-3	-2	-2	-3	1	5	1	-1	-3	0	-3	-3	-2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	47	0	1	3	0	0	36	0	0.31	0.10		
23 A	5	-2	-3	-1	-2	-1	-3	-2	-2	-2	-3	-2	1	0	-4	-3	0	91	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	2	0	0	2	0.71	0.16		
24 G	0	-3	-1	-2	-3	-3	6	-5	-5	-2	-4	-3	0	-2	-4	-4	-4	4	0	0	0	0	0	0	0	0	0	90	0	0	0	0	0	0	0	7	0	0	0	1.23	0.16			
25 L	-2	-3	-3	-4	-2	-3	-3	-4	-3	2	4	-3	1	2	-3	-1	-1	-2	-1	1	0	0	0	0	0	0	0	0	0	0	0	19	57	1	1	12	0	10	0	0	0.40	0.09		
26 L	-2	-3	-3	-4	-2	-3	-3	-4	-3	2	4	-3	1	0	-3	-1	-1	-2	-2	1	0	0	0	0	0	0	0	0	0	0	0	12	73	0	3	0	9	0	0	3	0.48	0.18		
27 E	-2	-4	-4	-4	-2	-3	-4	-4	-2	0	-4	-3	-3	-3	-3	-5	-2	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.85	0.17		
28 E	-2	-1	-1	1	-5	1	7	-3	-1	-4	-4	0	-4	-4	-2	-1	-2	-4	-3	4	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.18	0.16	
29 I	-1	-4	-4	-4	-2	-3	-4	-4	5	1	-3	1	2	-4	-3	0	0	-1	2	3	0	0	0	0	0	0	0	0	0	0	0	76	3	0	2	11	0	0	2	1	0	2.54	0.10	
30 N	-3	-1	8	-0	-4	0	-1	-1	0	-5	-1	-3	-4	-3	0	-1	-5	-3	-4	0	0	97	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.25	0.17	
31 R	-2	-7	1	-3	-5	0	-1	-3	-1	-4	-3	1	-2	-4	-3	-2	-2	-4	-3	4	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.28	0.16	
32 F	-2	-4	-4	-4	-2	-3	-4	-4	-2	0	-4	-4	-4	-4	-4	-4	-4	-4	-4	0	0	0	0	0	0	0	0	0	0	0	0	10	7	84	0	0	0	0	0	0	0.99	0.14		
33 F	-3	-4	-4	-3	-4	-4	-2	-1	-1	-4	-1	8	-4	-2	-3	0	-4	-1	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	86	0	3	0	8	1	1.09	0.14	
34 P	-2	-3	-3	-3	-4	-2	-2	-3	-3	-4	-4	-2	-4	-5	9	-1	-2	-5	-4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	98	2	0	0	0	0	2.34	0.18	
35 D	-3	-3	1	7	-5	-1	1	-2	-4	-5	-2	-4	-5	-2	-2	-1	-5	-4	-4	0	0	3	97	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.37	0.15	
36 A	4	-2	-2	-3	-1	-2	0	-2	-1	0	-2	-2	1	1	-3	-2	0	0	65	0	0	0	0	0	0	0	4	0	3	13	0	2	7	4	0	0	0	0	1	35	0.88	0.08		
37 E	-2	-3	-3	-3	-2	-2	-3	1	5	1	-1	-3	0	-3	-5	-2	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	95	0	1	5	0	0	0	0	0.71	0.10		
38 I	-1	-2	-2	-3	-2	-2	-2	-2	2	-2	0	-2	3	-3	-2	0	0	2	0	0	0	0	0	0	0	0	0	1	0	11	7	0	0	16	0	8	28	0	1	25	0.21	0.06		
39 F	-2	-3	-3	-4	-3	-3	-4	-3	-1	1	-3	0	6	-4	-1	-2	0	3	-1	1	0	0	0	0	0	0	0	0	0	0	11	1	13	0	62	0	5	0	5	1	0.68	0.11		
40 P	0	-3	-2	-2	-3	-2	-2	-2	-3	-3	-4	-2	-3	-4	7	1	-1	-1	-3	9	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	72	11	2	3	0	1.40	0.13	
41 F	-3	-4	-4	-3	-4	-4	-4	-2	-1	1	-4	-1	7	-3	-2	0	3	0	1	0	0	0	0	0	0	0	0	0	0	0	0	7	0	85	1	2	0	0	3	2	1.03	0.13		
42 A	-2	-3	-1	-4	-3	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7	0	87	1	5	0	3	0	0.77	0.07			
43 S	-3	-3	-1	-4	-3	0	0	-1	-1	-3	0	-2	-3	-1	5	-2	-2	-2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	89	0	0	0	0.48	0.04	
44 F	-3	-3	-4	-3	-4	-4	-2	0	0	-4	0	7	-4	-3	1	-3	1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0.91	0.04

Figure 49: A sample of an annotated dataset.

COVID-19 is an emerging, rapidly evolving situation.

[Public health information \(CDC\)](#) |
 [Research information \(NIH\)](#) |
 [SARS-CoV-2 data \(NCBI\)](#) |
 [Prevention and treatment information \(HHS\)](#)

Structures

CHL00105 : photosystem I subunit IX

Change PSSM/Sequence

Stacked Bar View

Reset

Download Matrix to File

Tutorial ?

Draw

table

showing only

those positions where the

consensus

is

Any

?

Hide Scores

Click on any score to compare the two residues.

Click on any column to sort the matrix by that column's scores.

P - consensus sequence position C - consensus sequence residue

Master:										gi_11467211										View Master in CD									
P	C	Master	A	G	I	L	V	M	F	W	P	C	S	T	Y	N	Q	H	K	R	D	E							
1	M	1 - M	-2	-4	-1	0	-1	8	-2	-3	-4	-3	-2	-2	-3	1	-2	-3	-2	-3	3	-2							
2	R	2 - R	-2	-3	-4	-3	-4	1	-4	-4	-3	-4	0	-1	-3	3	4	-1	3	4	-2	1							
3	D	3 - D	-3	-2	-5	-5	-4	-4	-5	-3	-5	-1	-2	0	5	-1	2	-2	-2	6	0								
4	L	4 - L	-3	-5	2	3	2	0	6	-2	-5	-3	-4	-3	0	-5	-4	-4	-4	-4	-5	-5							
5	K	5 - K	-2	-3	-2	1	-1	-1	-3	-4	-3	-4	-2	2	-3	-2	2	-2	5	0	-3	-1							
6	T	6 - T	-2	-3	-1	-3	-1	-2	-4	-4	-3	-3	-1	5	-3	-2	-1	3	4	-1	-2	-1							
7	Y	7 - Y	-3	-5	-3	-2	-3	-2	2	1	-4	-4	-3	-3	9	-4	-3	1	-3	3	-5	-4							
8	L	8 - L	-3	-5	2	5	0	1	1	-3	-4	-3	-4	-3	-2	-5	-4	-4	-4	-4	-5	-4							
9	S	9 - S	0	-2	-4	-4	-3	-3	-3	-3	-2	-2	6	0	1	-1	-1	-2	-2	-2	-2	-2							
10	T	10 - V	-1	-3	-1	-2	2	-2	-3	-4	-3	-2	0	6	-3	-2	-2	3	-2	-3	-3	-3							
P	C	Master	A	G	I	L	V	M	F	W	P	C	S	T	Y	N	Q	H	K	R	D	E							

Resources

Learn Page

Amino Acid Explorer

PSSM Viewer Help

CDD Help

Show Color Key

Questions or comments

Scores

10

9

8

7

6

5

4

3

Figure 50: A part of a sample from the annotated dataset.

5.1.2. Classification

We build training code including validation and classify data as follows, 20% of complete data set for testing, 80% for training, and also, we divided 80% of training data set for validation of 20% and 60% for training.

The Sensitivity, specificity, overall, MCC (Matthew's correlation coefficient) and accuracy were used to accurately measure the prediction. True positives, false positives, false negatives, and true negatives. When we change our parameter values, we change them all every time, for example, we change the learning rate, the epoch numbers, the kernel size for the 1-D convolution layer, the 1-D average pool layer and also for we change the activation function using random values in every-time.

5.2. Results

This section summarizes results from many experiments and trials until finding the ideal solutions.

5.2.1. Best Results Cases

Our best result of the model, we have got it in the twenty-one epoch's number and the results were 0.71 for training loss, 0.84 for training accuracy, 0.92 for training sensitivity, 0.83 for training specificity, 0.54 for MCC, 0.94 for the area under the cover, 1.01 for loss validation, 0.81 for accuracy validation, 0.77 for the validation sensitivity, 0.82 for validation specificity, 0.41 for MCC validation, and 0.87 for the area under the curve.

epoch	train loss	train acc	train_sens	train_spec	train_mcc	train_auc	val loss	val acc	val_sens	val_spec	val_mcc	val_auc
0	1.1276604	0.7729598	0.8098592	0.7689032	0.3816334	0.8432977	0.9796084	0.7864123	0.8113208	0.7836861	0.3973537	0.8600686
1	1.0157929	0.8006278	0.7922535	0.8015484	0.4057935	0.8601895	0.8984155	0.8092136	0.8066038	0.8094992	0.4235563	0.8726951
2	0.9820608	0.7847012	0.8169014	0.7811613	0.3987411	0.8645538	0.925211	0.8022336	0.8254717	0.7996902	0.4239547	0.8676323
3	0.9488448	0.6785631	0.9319249	0.6507097	0.3542772	0.8773621	0.9465364	0.6905537	0.9103774	0.6664946	0.3526107	0.8704036
4	0.911297	0.8006278	0.8392019	0.7963871	0.4294026	0.886537	0.9902892	0.8054909	0.8018868	0.8058854	0.4163039	0.8713582
5	0.8897087	0.6843757	0.9389671	0.6563871	0.362644	0.8884964	1.0369062	0.6914844	0.9056604	0.6680434	0.3509777	0.8675032
6	0.841491	0.8143455	0.8591549	0.8094194	0.457162	0.9080366	0.9931891	0.8171242	0.8207547	0.8167269	0.4412808	0.8763771
7	0.835017	0.802604	0.8931925	0.7926452	0.4586884	0.9192962	0.94121	0.7957189	0.8160377	0.7934951	0.4110309	0.8664001
8	0.7848692	0.8378284	0.8931925	0.8317419	0.5070142	0.9314993	0.9457007	0.8217776	0.7641509	0.8280847	0.4194275	0.8730263
9	0.7138881	0.8178331	0.9694836	0.8011613	0.515475	0.9461196	1.1467598	0.7943229	0.8207547	0.79143	0.4116833	0.867155
10	1.1227326	0.7803488	0.7779083	0.7806169	0.3736524	0.8319334	1.0320643	0.7880056	0.8356808	0.7827657	0.4121066	0.8459135
11	1.0330104	0.737907	0.8625147	0.7242225	0.371514	0.859617	0.9840353	0.7373315	0.8826291	0.7213622	0.381267	0.863312
12	0.9883895	0.7822093	0.8155112	0.7785521	0.3949721	0.8708663	0.8930181	0.7852162	0.8544601	0.7776058	0.4182352	0.8726919
13	0.9489259	0.7916279	0.8519389	0.7850045	0.4245099	0.8825112	0.8618659	0.7912599	0.8356808	0.7863777	0.4160198	0.8782347
14	0.9095849	0.8323256	0.8249119	0.8331398	0.4655215	0.8946209	1.0458312	0.8219433	0.7605634	0.8286894	0.41845	0.8761925
15	0.8906733	0.8338372	0.8378378	0.8333979	0.4741038	0.9040891	0.8930997	0.8196188	0.7746479	0.8245614	0.4222645	0.8780748
16	0.8675588	0.7819767	0.8860165	0.770551	0.4301425	0.8973878	0.9053138	0.7740586	0.8638498	0.7641899	0.4101439	0.8759272
17	0.8151882	0.8196512	0.8813161	0.8128791	0.4750812	0.922224	0.9274638	0.800093	0.8122066	0.7987616	0.4151508	0.8817861
18	0.7769768	0.7931395	0.9189189	0.7793264	0.4595916	0.9280872	0.9459304	0.7745235	0.8497653	0.7662539	0.4035743	0.8752186
19	0.7100674	0.8484884	0.9294947	0.8395922	0.5404271	0.948437	1.0159109	0.8177592	0.7746479	0.8224974	0.4196635	0.8751338
20	1.1290548	0.7023948	0.8611765	0.6849845	0.3377642	0.8456132	1.1103802	0.6840391	0.8037383	0.678081	0.2932871	0.804371
21	0.990993	0.7648221	0.8352941	0.7570949	0.3852308	0.8601336	1.1167035	0.7491857	0.7897196	0.7447028	0.3464702	0.8218093
22	0.9604624	0.8251569	0.8023529	0.8276574	0.4436684	0.8827084	1.0023229	0.8050256	0.7149533	0.8149871	0.3727408	0.8393707
23	0.9236264	0.8348059	0.7776471	0.8410733	0.4456629	0.8879492	1.0600612	0.8092136	0.6869159	0.822739	0.3637495	0.8376561
24	0.9074435	0.8023715	0.84	0.7982456	0.4316497	0.8885597	1.0657729	0.778967	0.7663551	0.7803618	0.3670202	0.839489
25	0.8694323	0.7122762	0.9388235	0.6874355	0.3861388	0.8995622	1.0352768	0.6700791	0.864486	0.6485788	0.3132774	0.8363037
26	0.8195968	0.8215531	0.84	0.8195304	0.4572067	0.9044289	1.2301282	0.7980456	0.7523364	0.8031008	0.3829788	0.846345
27	0.7915182	0.7672634	0.9588235	0.746259	0.4497967	0.9280208	0.979368	0.7226617	0.8411215	0.7095607	0.3468162	0.8501389
28	0.7180799	0.7523832	0.98	0.7274252	0.4448625	0.9423369	1.0044801	0.6961377	0.8411215	0.6801034	0.3229477	0.8390253
29	0.6676147	0.7843525	0.9811765	0.7627709	0.4796985	0.9442513	1.1091811	0.735691	0.7943925	0.729199	0.3352963	0.8365186

Figure 51: Illustration of the best cases results.

5.2.2. Worst Results Cases

According to Figure 52, the worst result of the model, we have got it in the thirty epoch's number and the results were 1.04 for training loss, 0.73 for training accuracy, 0.82 for training sensitivity, 0.72 for training specificity, 0.35 for MCC, 0.83 for the area under the cover, 1.09 for loss validation, 0.72 for accuracy validation, 0.77 for the validation sensitivity, 0.72 for validation specificity, 0.31 for MCC validation, and 0.80 for the area under the curve.

epoch	train loss	train acc	train_sens	train_spec	train_mcc	train_auc	val loss	val acc	val_sens	val_spec	val_mcc	val_auc
0	1.37650001	0.80330156	0.22652582	0.86670968	0.07967027	0.66498917	1.35939614	0.81805491	0.26415094	0.87867837	0.1244746	0.70122783
1	1.34986231	0.47907463	0.8556338	0.43767742	0.17824026	0.67521505	1.31554291	0.48115403	0.88679245	0.43675787	0.196553	0.71198167
2	1.30476668	0.51232272	0.84507042	0.47574194	0.19288423	0.69307474	1.24758978	0.51093532	0.87735849	0.47083118	0.20935105	0.73052328
3	1.24859803	0.55603348	0.83920188	0.52490323	0.21758862	0.72052855	1.18088127	0.5532806	0.88207547	0.51729479	0.23841129	0.7576514
4	1.2030228	0.48291095	0.91784038	0.43509677	0.21519776	0.74842912	1.15995893	0.48627269	0.93396226	0.43727414	0.22590393	0.7773205
5	1.17915097	0.7113462	0.67840376	0.71496774	0.25109045	0.7617461	1.1049614	0.71800838	0.73113208	0.71657202	0.28444844	0.792146
6	1.15481615	0.671588	0.77347418	0.6603871	0.2666671	0.77163138	1.07403734	0.67659377	0.81603774	0.66133196	0.29243096	0.79922025
7	1.13898644	0.65589398	0.82511737	0.63729032	0.28100967	0.78727942	1.06376506	0.66542578	0.85849057	0.6442953	0.3053805	0.80838025
8	1.12529924	0.70948617	0.76408451	0.70348387	0.2942652	0.79159594	1.04491659	0.71149372	0.81603774	0.70005163	0.3224667	0.81304975
9	1.10787182	0.66379912	0.85328639	0.64296774	0.30184771	0.80258193	1.02985385	0.67612843	0.86320755	0.65565307	0.316424	0.81980255
10	1.37161091	0.65034884	0.51938895	0.66473093	0.11500431	0.65603522	1.35771421	0.64993027	0.47887324	0.66873065	0.09268959	0.66111184
11	1.33775408	0.61941861	0.62749706	0.61853142	0.14960629	0.6733829	1.31292184	0.62250116	0.58215962	0.62693499	0.12782889	0.68188491
12	1.28378141	0.47930233	0.88249119	0.43502387	0.19324501	0.70530703	1.25634518	0.47791725	0.89671362	0.43188855	0.20040533	0.71942422
13	1.22637889	0.68744186	0.61692127	0.69518648	0.19734978	0.72777436	1.20977144	0.70385867	0.6056338	0.71465428	0.20559114	0.73867595
14	1.18661066	0.53837209	0.90011751	0.49864499	0.23893798	0.75806613	1.17240726	0.54067875	0.92018779	0.49896801	0.25130967	0.76841718
15	1.1598914	0.61616279	0.84723854	0.59078591	0.26277335	0.76947187	1.13542605	0.62529056	0.85446009	0.6001032	0.27321398	0.77940813
16	1.14606315	0.68348837	0.77790834	0.67311911	0.27871305	0.77862569	1.12228728	0.69920967	0.78873239	0.68937049	0.29788309	0.78772221
17	1.1335924	0.6205814	0.85781434	0.59452833	0.27148682	0.78443494	1.1095061	0.63133426	0.87323944	0.60474716	0.28744473	0.79436474
18	1.12046426	0.67732558	0.81316099	0.66240805	0.29188281	0.78995106	1.08945532	0.69316597	0.82159624	0.67905057	0.30964185	0.79943749
19	1.1088774	0.62895349	0.86368978	0.6031746	0.28063905	0.79872692	1.08345781	0.64342166	0.88732394	0.61661507	0.30376299	0.80723315
20	1.36989086	0.52848175	0.75764706	0.50335397	0.15592897	0.67229922	1.36005481	0.51000465	0.71028037	0.4878553	0.11890167	0.64015552
21	1.32256004	0.63148105	0.62705882	0.63196594	0.15821417	0.68724876	1.31723989	0.61656585	0.57943925	0.62067184	0.12235219	0.65347388
22	1.25647581	0.39688445	0.94823529	0.33642931	0.18395202	0.71821951	1.28803061	0.38343416	0.92523365	0.32351421	0.1627356	0.67766186
23	1.20352426	0.57196001	0.86588235	0.53973168	0.2420806	0.75029655	1.23253625	0.55979525	0.79906542	0.53333333	0.19906751	0.7077616
24	1.16073515	0.67344804	0.79294118	0.66034572	0.27806058	0.77405467	1.21113108	0.64913913	0.70560748	0.64289406	0.21377565	0.73132169
25	1.13745616	0.70471983	0.75294118	0.69943241	0.28392526	0.78341574	1.20101116	0.68450442	0.68691589	0.68423773	0.23259635	0.74402425
26	1.10935293	0.5916066	0.9	0.55779154	0.27330755	0.7955764	1.18423355	0.56537925	0.87383178	0.53126615	0.24264564	0.75984931
27	1.09549322	0.76400837	0.70235294	0.77076883	0.31584545	0.80747648	1.1789404	0.74499767	0.65420561	0.75503876	0.27126372	0.7722838
28	1.07277327	0.73192281	0.78705882	0.72587719	0.32685342	0.82270109	1.12375269	0.71754304	0.75700935	0.7131783	0.29859841	0.78884059
29	1.04355804	0.73820042	0.82117647	0.72910217	0.3505189	0.83745394	1.09060176	0.72917636	0.77570094	0.72403101	0.31930173	0.80439277

Figure 52: Illustration of the worst cases results.

5.2.3. Limitations

When we encounter the issue of sequencing protein sequences, we run into problems when we download the data sets This means our data is tainted. This, therefore, is why we use the PSI-BLAST tool to remove sequences that contain 30% or more of identity.

5.3. Evaluation

Let us look at this subsection using charts, curves, and graphs, and results will be presented in statistical format. Besides that, additional information about the space and time utilization will be provided.

5.3.1. Accuracy Evaluation

Our RNN model has been implemented throughout PyTorch with Google collaboration. We have been created a new RNN model from scratch with specified value of epochs was equals to 30. All in all, the learning rate has been held constant at 0.0001 during training. Up to great imbalance in the dataset between adaptor and non-adaptor proteins, we implemented binary cross-entropy as part of our training process. As it turned out, the frequency factors were proportional to the inverse to the weight of the class weights. Specificity, Sensitivity, accuracy, MCC, AUC are the predictions had previously been calculated on them prior to this test. The "true positives", "false positives", "true negatives", and "false negatives" are facts, as opposed, respectively as shown in Figure 53.

$$\begin{aligned}\text{Sensitivity} &= \frac{TP}{TP + FN} \\ \text{Specificity} &= \frac{TN}{TN + FP} \\ \text{Accuracy} &= \frac{TP + TN}{TP + TN + FP + FN} \\ \text{MCC} &= \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}\end{aligned}$$

Figure 53: Illustration of measurements that were used for our model.

According to Figure 54, Figure 55, Figure 56, Figure 57, Figure 58, and Figure 59, In our recent attempts to improve the parameters of the model, we obtained these results which are the best results for our work. In fact, we used all the parameters we mentioned in the previous section which is the classification section but we used the ADAM optimizer instead of the SGD optimizer.

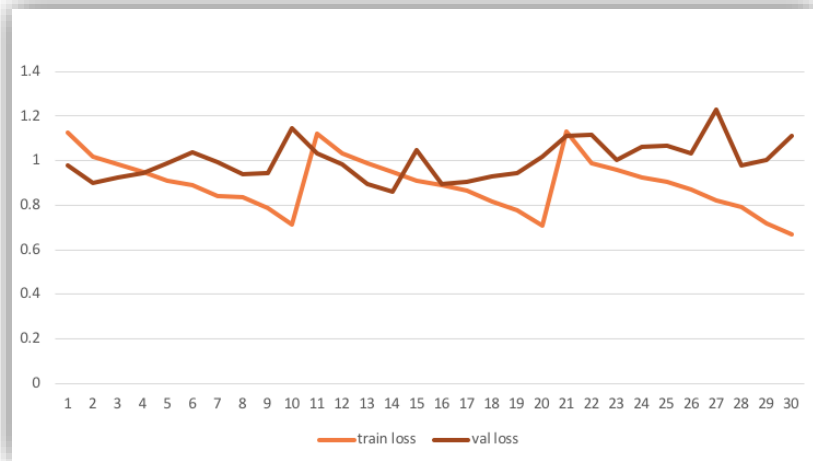


Figure 54: Illustration of training and validation loss.

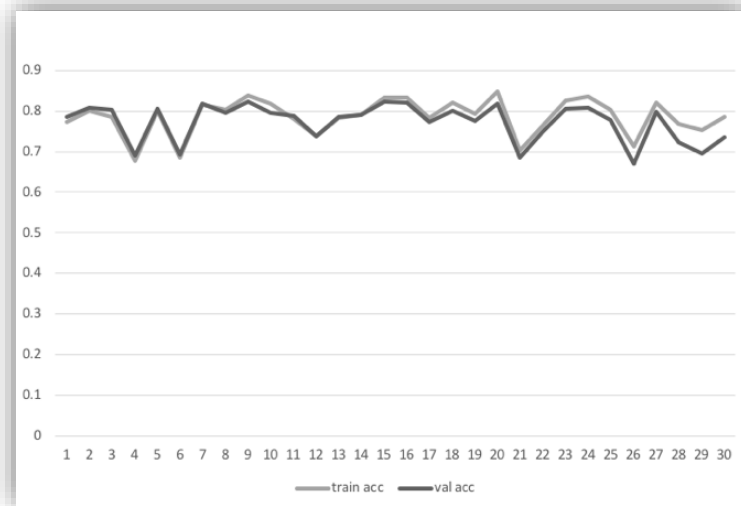


Figure 55: Illustration of training and validation accuracy.

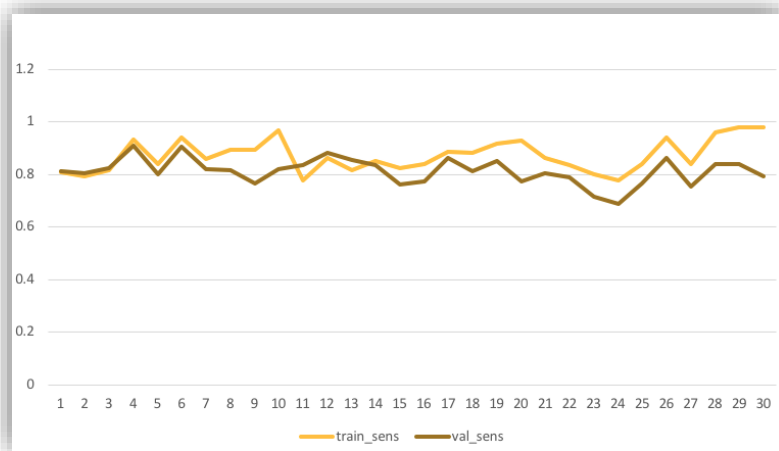


Figure 56: Illustration of training and validation sensitivity.

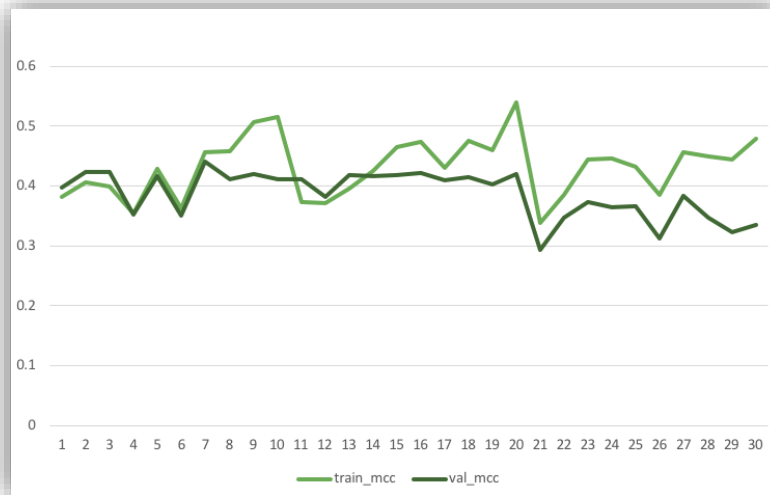


Figure 57: Illustration of training and validation MCC.

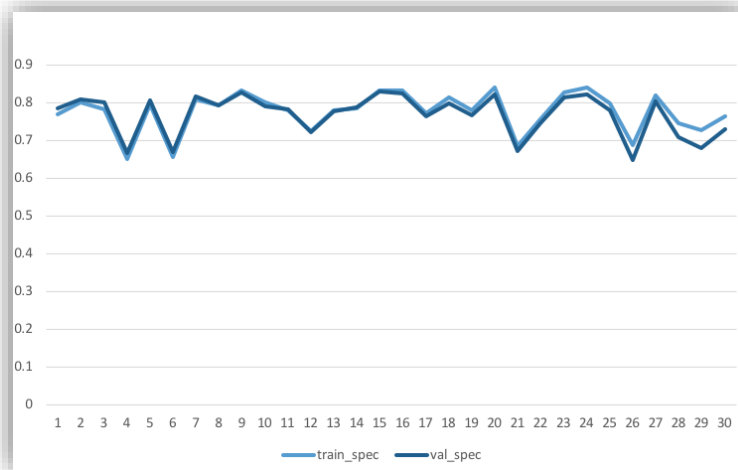


Figure 58: Illustration of training and validation specificity.

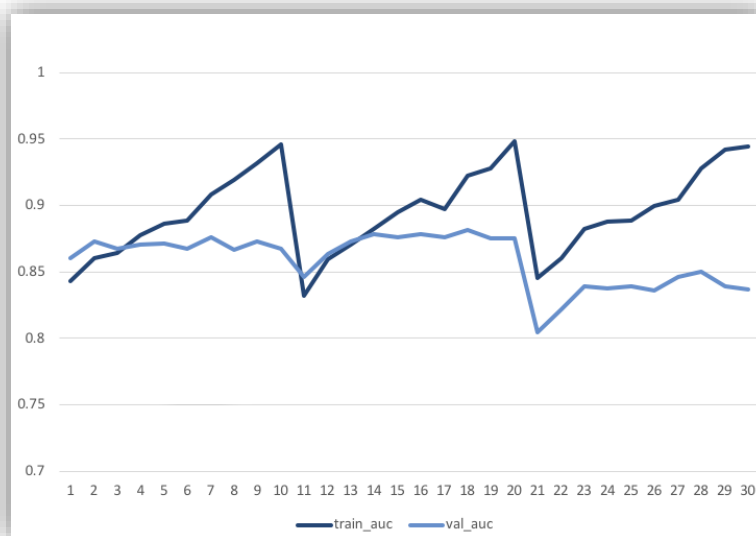


Figure 59: Illustration of training and validation area under the curve.

According to Figure 60, Figure 61, Figure 62, Figure 63, Figure 64, and Figure 65, on our first attempt to improve our parameters of the model, we have got these results and they are the worst results for our work. In fact, we used all the parameters we mentioned in the previous section which is the classification section but we have been used SGD instead of ADAM in the optimizer.

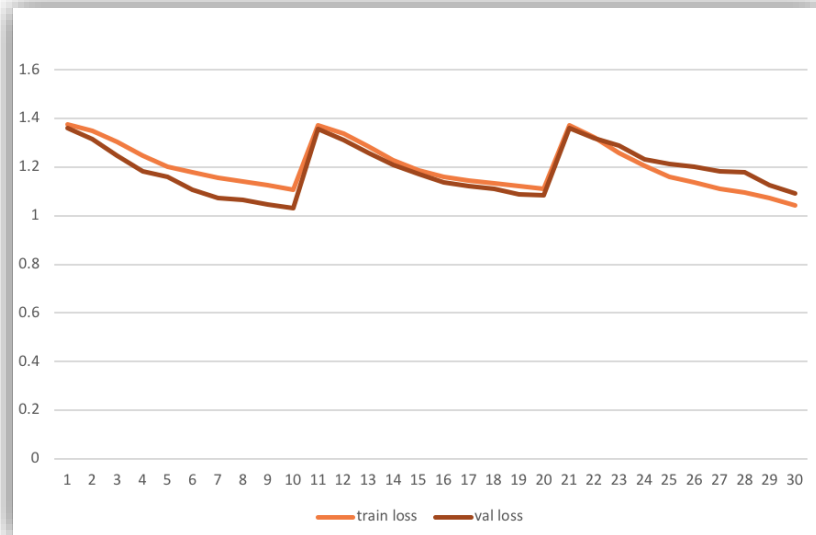


Figure 60: Illustration of training and validation loss for the second model.

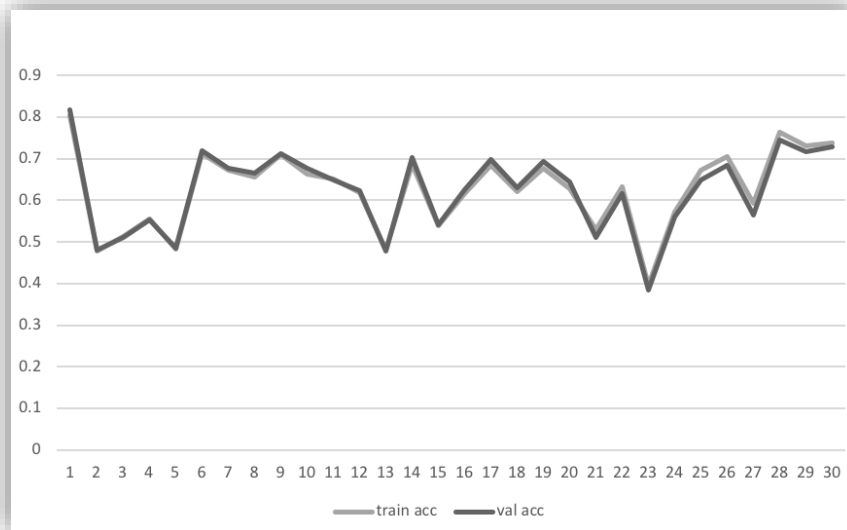


Figure 61: Illustration of training and validation accuracy for second model.

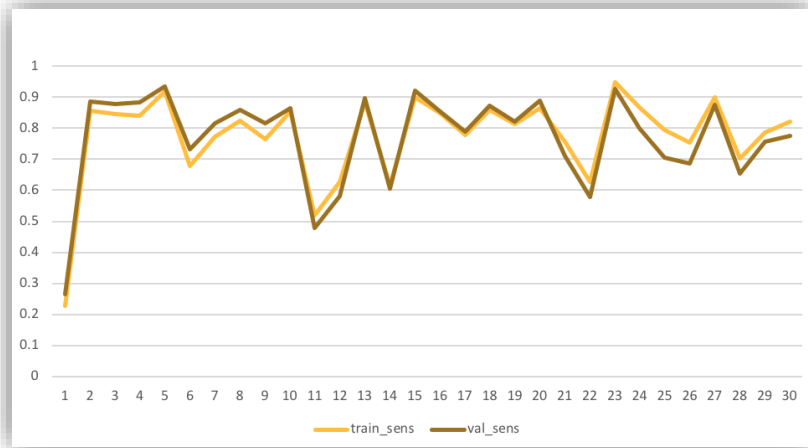


Figure 62: Illustration of training and validation sensitivity for second model.

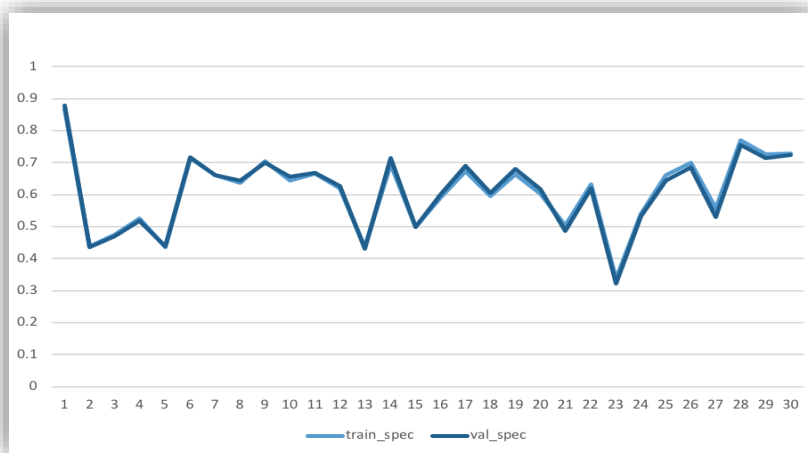


Figure 63: Illustration of training and validation specificity for second model.

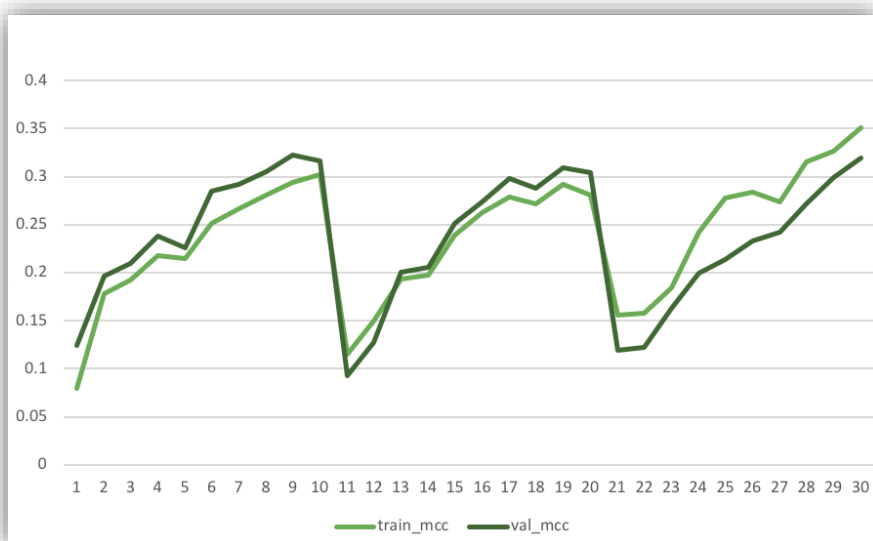


Figure 64: Illustration of training and validation MCC for second model.

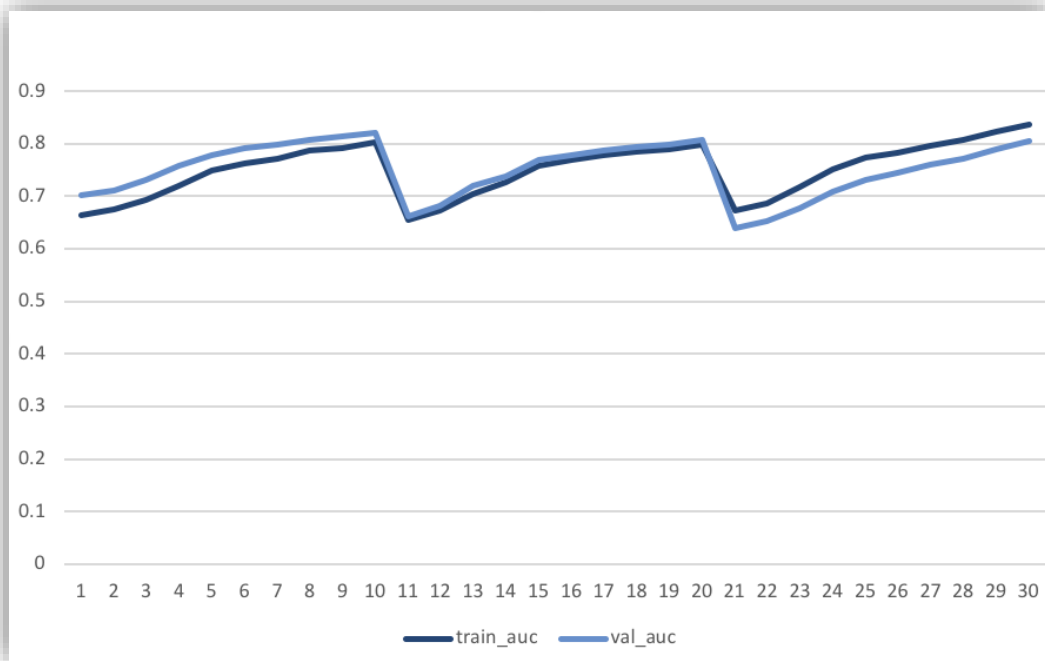


Figure 65: Illustration of training and validation area under the curve.

According to Figure 66, here is a comparison of two results which are the best and the worst results. I mean the difference when we change from SGD optimizer to ADAM optimizer. It makes great significant results.

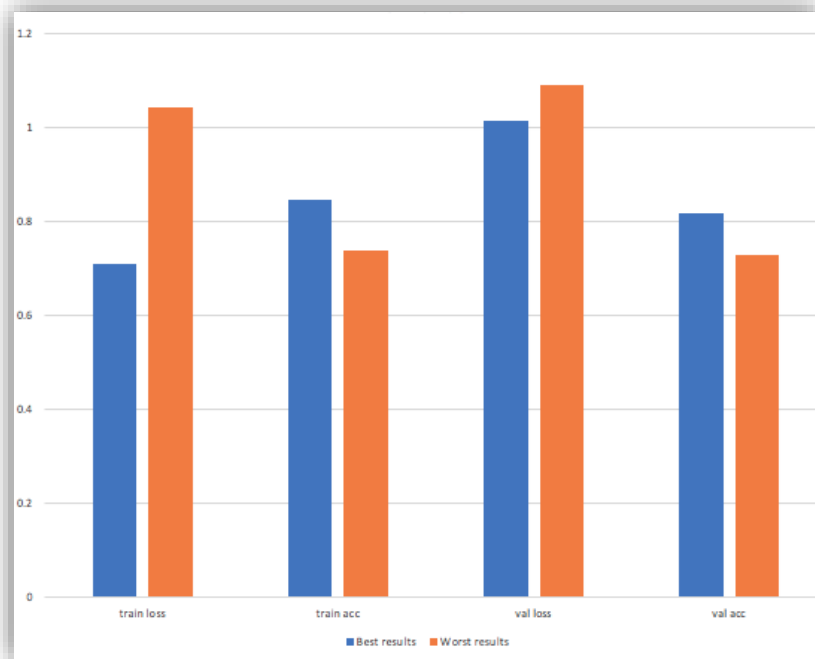


Figure 66: Comparison of our two results of the model.

According to Figure 67, Figure 68, Figure 69, Figure 70, and Figure 71, we explain the differences between the results. If we focus more between the old RNN model and our RNN model to differentiate why we get better results. we use ADAM optimizer, RELU in 1-D AVG pool layer, RELU in 1-D CONV, and Sigmoid function in classification. But they used SGD optimizer, Tanh in 1-D AVG pool layer, Tanh in 1-D CONV, and Sigmoid function in classification. There are researchers that have used these algorithms to predict adaptor and non-adaptor proteins such as SVM, KNN, RF, and CNN.

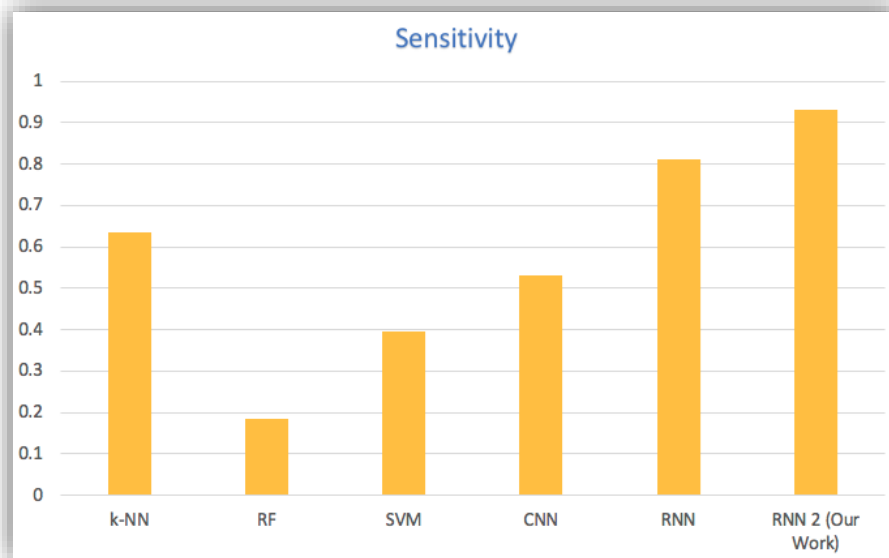


Figure 67: Comparison of previous work with sensitivity Evaluation.

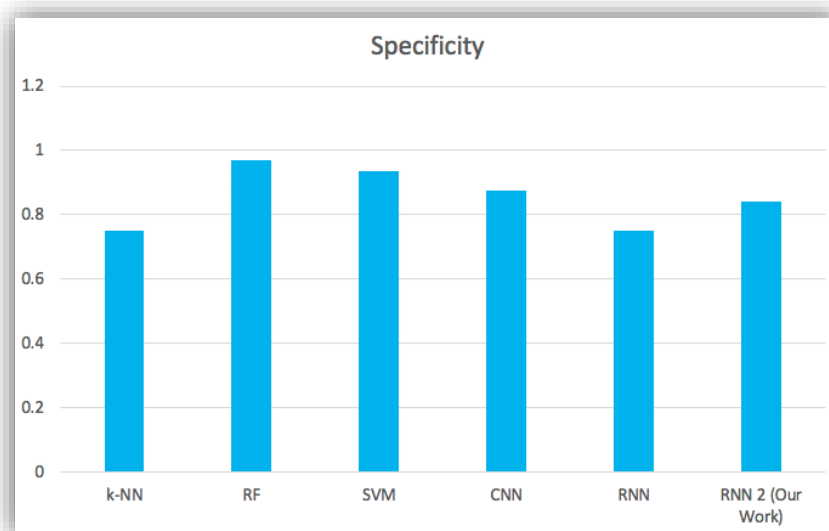


Figure 68: Comparison of previous work with Specificity Evaluation.

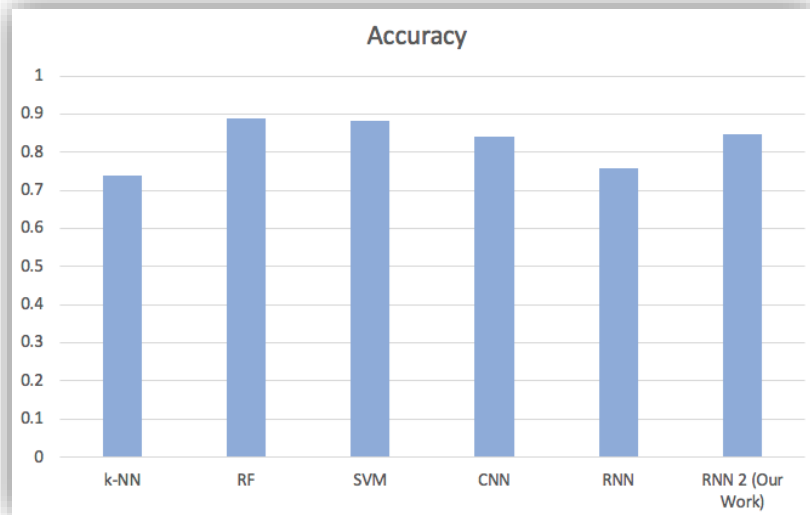


Figure 69: Comparison of previous work with Accuracy Evaluation.

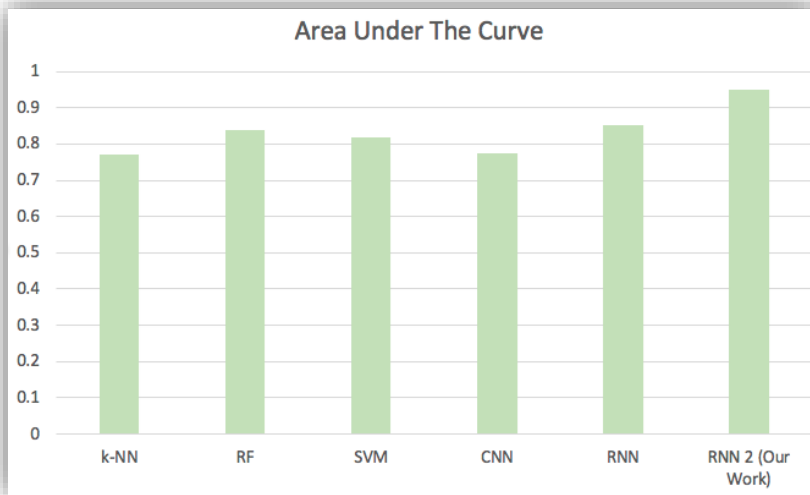


Figure 70: Comparison of previous work with Area Under the Curve Evaluation.

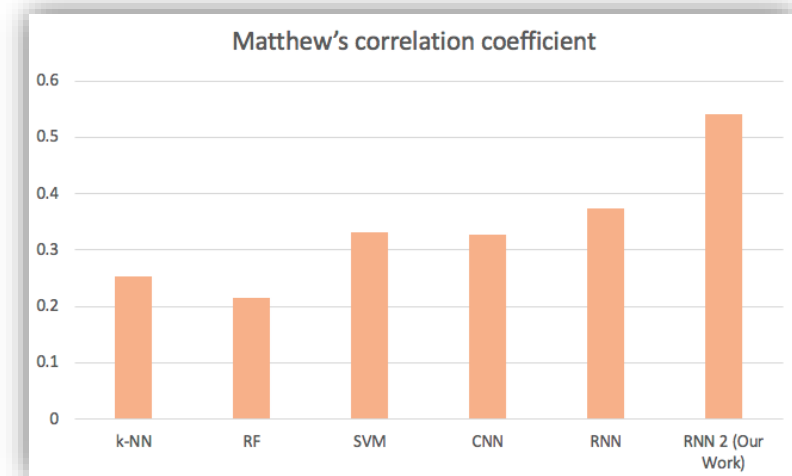


Figure 71: Comparison of previous work with Mathew's correlation coefficient Evaluation.

5.3.2. Time Performance

Using Google Colab GPUs versus using the local machine's CPUs gives you a significant time-saver. the models completed on the local CPU completed in one hour per epoch, while models completed on Google Colab GPU took 19 minutes per epoch.

Chapter 6: Conclusion and Future Work

6.1. Conclusion

This paper introduces an innovative for best results method for determining adaptor proteins, one that is purely based on RNN and PSSM profiles, and sequence data. This research that adapts this mixture to protein function prediction. with this method, we could already retain all of the PSSM training information and avoid the missing value as much as possible. We have better results than state-of-the-art techniques in terms of Sensitivity, Area under the curve as well as Mathew's correlation coefficient.

A powerful method for exploring new protein adapter proteins was examined in this study. This research allows us to use RNN and PSSM profiles for bioinformatics and computational biology. We can base our strategy for who are seeking to improve the prediction performance of different protein function problems.

6.2. Problem Issues

These are some of the problems faced when implementing the systems algorithms found in this section.

6.2.1. Technical issues:

Our model was consuming a lot of computational resources which means we have to work on a computer with high performance, especially in GPU and RAM. Google Collab provides an online environment to work on with high performance because it is specifically designed for the problems we encountered. So, we turned our project into a google collab by creating a Gmail account and uploading our project. It worked very well and was easy to use.

6.2.2. Scientific issues:

When we download the data sets to obtain protein sequences, we run into the problem of sequencing of identity. It actually makes our data biased. Therefore, we use the PSI-BLAST tool to remove these identity sequences if the sequence is similar to another with 30% or higher of the entire sequence.

One of the most serious problems is that each epoch is taking at least 19 minutes per one epoch which is leading us for wasting a lot of time just to test our model one time.

6.3. Future Work

Last but not least, physical characteristics was already successfully applied in bioinformatics [29][27]. Theoretically, this can be done as such, since it is possible to mix PSSM characteristics and physicochemical proteins. Following this, these hybrid capabilities could be inserted into our proposed architecture. It is important that future studies focus on these hybrid features in order to better the protein function prediction results.

List of References:

- [1] Richard C Mohs and Nigel H Greig. “Drug discovery and development: Role of basic biological research”. In: *Alzheimer’s & Dementia: Translational Research & Clinical Interventions* 3.4 (2017), pp. 651–657.
- [2] Steven M Paul et al. “How to improve R&D productivity: the pharmaceutical industry’s grand challenge”. In: *Nature reviews Drug discovery* 9.3 (2010), pp. 203–214.
- [3] Joseph A DiMasi, Henry G Grabowski, and Ronald W Hansen. “Innovation in the pharmaceutical industry: new estimates of R&D costs”. In: *Journal of health economics* 47 (2016), pp. 20–33.
- [4] Taju SW, Nguyen T-T-D, Le N-Q-K, Kusuma RMI, Ou Y-Y. DeepEfflux: a 2-D convolutional neural network model for identifying families of efflux proteins in transporters. *Bioinformatics*. 2018; 34(18):3111–7. <https://doi.org/10.1093/bioinformatics/bty302>.
- [5] Le N-Q-K, Nguyen BP. Prediction of FMN binding sites in electron transport chains based on 2-D CNN and PSSM profiles. *IEEE/ACM Trans Comput Biol Bioinforma*. 2019:1–9. <https://doi.org/10.1109/TCBB.2019.2932416>.
- [6] Flynn DC. Adaptor proteins. *Oncogene*. 2001; 20(44):6270. <https://doi.org/10.1038/sj.onc.1204769>.
- [7] Verma S, Vaughan T, Bunting KD. Gab adapter proteins as therapeutic targets for hematologic disease. *Adv Hematol*. 2012; 2012. <https://doi.org/10.1155/2012/380635>. Accessed 01 Apr 2019.
- [8] Marton N, Baricza E, Érsek B, Buzás EI, Nagy G. The emerging and diverse roles of src-like adaptor proteins in health and disease. *Mediators Inflamm*. 2015; 2015. <https://doi.org/10.1155/2015/952536>. Accessed 01 Apr 2019.
- [9] Isaka Y. Adaptor protein is a new therapeutic target in chronic kidney disease. *Kidney Int*. 2017; 92(6):1312–4. <https://doi.org/10.1016/j.kint.2017.06.012>.
- [10] Hatsugai N, Nakatsuji A, Unten O, Ogasawara K, Kondo M, Nishimura M, Shimada T, Katagiri F, Hara-Nishimura I. Involvement of adapter protein complex 4 in hypersensitive cell death induced by avirulent bacteria. *Plant Physiol*. 2018; 176(2):1824–34. <https://doi.org/10.1104/pp.17.01610>.
- [11] Paliwal KK, Sharma A, Lyons J, Dehzangi A. A tri-gram based feature extraction technique using linear probabilities of position specific scoring matrix for protein fold recognition. *IEEE Trans NanoBiosci*. 2014; 13(1):44–50. <https://doi.org/10.1109/TNB.2013.2296050>.
- [12] Chandra AA, Sharma A, Dehzangi A, Tsunoda T. EvolStruct-Phogly: incorporating structural properties and evolutionary information from profile bigrams for the phosphoglycylation prediction. *BMC Genomics*. 2019; 19(9):984. <https://doi.org/10.1186/s12864-018-5383-5>.

- [13] Dehzangi A, López Y, Lal SP, Taherzadeh G, Sattar A, Tsunoda T, Sharma A. Improving succinylation prediction accuracy by incorporating the secondary structure via helix, strand and coil, and evolutionary information from profile bigrams. PLOS ONE. 2018; 13:1–16. <https://doi.org/10.1371/journal.pone.0191900>.
- [14] D, Dauphin YN. Convolutional sequence to sequence learning. 2017. <https://arxiv.org/abs/1705.03122>.
- [15] Neil C Jones and Pavel Pevzner. An introduction to bioinformatics algorithms. MIT press, 2004.
- [16] Abraham White, Philip Handler, Emil Smith, DeWitt Stetten Jr, et al. Principles of biochemistry. Principles of Biochemistry., (Edn 2), 1959.
- [17] Ron Milo. What is the total number of protein molecules per cell volume? a call to rethink some published values. Bioessays, 35(12):1050–1055, 2013
- [18] UniProt Consortium et al. Uniprot: a hub for protein information. Nucleic acids research, page gku989, 2014.
- [19] Shoichet, B. (2004). Virtual screening of chemical libraries. Nature, 432(7019):862–865. 15602552[pmid].
- [20] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep learning. MIT press, 2016.
- [21] Pedro Larranaga, Borja Calvo, Roberto Santana, Concha Bielza, Josu Galdiano, Inaki Inza, Jos´e A Lozano, Rub´en Armananzas, Guzm´an Santaf´e, Aritz P´erez, et al. Machine learning in bioinformatics. Briefings in bioinformatics, 7(1):86–112, 2006.
- [22] Helen M Berman, Tammy Battistuz, Talapady N Bhat, Wolfgang F Bluhm, Philip E Bourne, Kyle Burkhardt, Zukang Feng, Gary L Gilliland, Lisa Iype, Shri Jain, et al. The protein data bank. Acta Crystallographica Section D: Biological Crystallography, 58(6):899–907, 2002.
- [23] Mart´in Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Man´e, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Vi´egas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
- [24] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlch´e

- Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems* 32, pages 8024–8035. Curran Associates, Inc., 2019
- [25] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.
- [26] Francois Chollet et al. Keras. <https://keras.io>, 2015.
- [27] Heng Liao, Jiajin Tu, Jing Xia, and Xiping Zhou. Davinci: A scalable architecture for neural network computing. In *2019 IEEE Hot Chips 31 Symposium (HCS)*, pages 1–44. IEEE, 2019.
- [28] Chris Lattner, Jacques Pienaar, Mehdi Amini, Uday Bondhugula, River Riddle, Albert Cohen, Tatiana Shpeisman, Andy Davis, Nicolas Vasilache, and Oleksandr Zinenko. Mlir: A compiler infrastructure for the end of moore’s law. *arXiv preprint arXiv:2002.11054*, 2020.
- [29] Tianqi Chen, Thierry Moreau, Ziheng Jiang, Lianmin Zheng, Eddie Yan, Haichen Shen, Meghan Cowan, Leyuan Wang, Yuwei Hu, Luis Ceze, et al. {TVM}: An automated end-to-end optimizing compiler for deep learning. In *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, pages 578–594, 2018.
- [30] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [31] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354– 359, 2017.
- [32] Andrew W Senior, Richard Evans, John Jumper, James Kirkpatrick, Laurent Sifre, Tim Green, Chongli Qin, Augustin Zidek, Alexander WR Nelson, Alex Bridgland, et al. Improved protein structure prediction using potentials from deep learning. *Nature*, pages 1–5, 2020.
- [33] A. Roy, A. Kucukural, Y. Zhang I-TASSER: a unified platform for automated protein structure and function prediction *Nat. Protoc.*, 5 (2010), pp. 725-738
- [34] Y. Zhang I-TASSER server for protein 3D structure prediction *BMC Bioinformatics*, 9 (2008), p. 40
- [35] Y. Zhang, J. Skolnick TM-align: a protein structure alignment algorithm based on the TM-score *Nucleic Acids Res.*, 33 (2005), pp. 2302-2309
- [36] B.H. Dessailly, M.F. Lensink, C.A. Orengo, S.J. Wodak LigASite—a database of biologically relevant binding sites in proteins with known apo-structures *Nucleic Acids Res.*, 36 (Database issue) (2008), pp. D667-D673
- [37] E. Perola, W.P. Walters, P.S. Charifson A detailed comparison of current docking and scoring methods on systems of pharmaceutical relevance *Proteins*, 56 (2004), pp. 235-249

- [38] H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, P.E. Bourne The Protein Data Bank Nucleic Acids Res., 28 (2000), pp. 235
- [39] Bauer RA, Günther S, Jansen D, Heeger C, Thaben P, Preissner R: SuperSite: dictionary of metabolite and drug binding sites in proteins. Nucl Acids Res 2009, 37: D195–200.
10.1093/nar/gkn618
- [40] Sobolev V, Sorokine A, Prilusky J, Abola EE, Edelman M: Automated analysis of interatomic contacts in proteins. Bioinformatics 1999, 15: 327–332.
10.1093/bioinformatics/15.4.327
- [41] Kaur H, Raghava GPS: Prediction of β -turns in proteins from multiple alignments using neural network. Protein Sci 2003, 12: 627–634. 10.1110/ps.0228903
- [42] Kaur H, Raghava GPS: A neural-network based method for prediction of gamma-turns in proteins from multiple sequence alignment. Protein Sci 2003, 12: 923–929.
10.1110/ps.0241703
- [43] Chen C, Chen L, Zou X, Cai P: Prediction of protein secondary structure content by using the concept of Chou's pseudo amino acid composition and support vector machine. Protein & Peptide Letters 2009, 16(1):27–31. 10.2174/092986609787049420
- [44] Ding H, Luo L, Lin H: Prediction of cell wall lytic enzymes using Chou's amphiphilic pseudo amino acid composition. Protein & Peptide Letters 2009, 16: 351–355.
10.2174/092986609787848045
- [45] Kumar M, Gromiha M, Raghava GPS: Prediction of RNA binding sites in a protein using SVM and PSSM profile. Proteins: Structure, Function, and Bioinformatics 2007, 71: 189–194.
10.1002/prot.21677
- [46] Kaur H, Raghava GPS: A neural-network based method for prediction of gamma-turns in proteins from multiple sequence alignment. Protein Sci 2003, 12: 923–929.
10.1110/ps.0241703
- [47] Gehring J, Auli M, Grangier D, Yarats D, Dauphin YN. Convolutional sequence to sequence learning. 2017. <https://arxiv.org/abs/1705.03122>.
- [48] Rhee, S. G., 2006. Cell signaling. H₂O₂, a necessary evil for cell signaling. Science 312, 1882-3.
- [49] Jordan, J. D., Landau, E. M., Iyengar, R., 2000. Signaling networks: the origins of cellular multitasking. Cell 103, 193-200.
- [50] Wong, J. V., Li, B., You, L., 2012. Tension and robustness in multitasking cellular networks. PLoS Comput Biol 8, e1002491.
- [51] Kobilka, B., 1992. Adrenergic receptors as models for G protein-coupled receptors. Annu Rev Neurosci 15, 87-114.
- [52] Rohrer, D. K., Kobilka, B. K., 1998. G protein-coupled receptors: functional and mechanistic insights through altered gene expression. Physiol Rev 78, 35-52.

- [53] Evans, P. D., Levitan, I. B., 1986. Receptors and ion channels. *J Exp Biol* 124, 1-4.
- [54] Dykstra, M., Cherukuri, A., Sohn, H. W., Tzeng, S. J., Pierce, S. K., 2003. Location is everything: lipid rafts and immune cell signaling. *Annu Rev Immunol* 21, 457-81.
- [55] Sassone-Corsi, P., 2012. The cyclic AMP pathway. *Cold Spring Harb Perspect Biol* 4.
- [56] Archer, S., 1978. QSAR: a critical appraisal. *NIDA Research Monograph*, 86-102.
- [57] Puzyn, T., Leszczynski, J., Cronin, M. T. D. Eds.), 2010. *Recent Advances in QSAR Studies: Methods and applications*. Springer.
- [58] González-Díaz, H., Prado-Prado, F., Pérez-Montoto, L. G., Duardo-Sánchez, A., López-Díaz, A., 2009. QSAR Models for Proteins of Parasitic Organisms, Plants and Human Guests: Theory, Applications, Legal Protection, Taxes, and Regulatory Issues. *Current Proteomics* 6, 214-227.
- [59] Prado-Prado, F., Garcia-Mera, X., Escobar, M., Alonso, N., Caamano, O., Yanez, M., Gonzalez-Diaz, H., 2012. 3D MI-DRAGON: New Model for the Reconstruction of US FDA Drug- Target Network and Theoretical-Experimental Studies of Inhibitors of Rasagiline Derivatives for AChE. *Curr Top Med Chem* 12, 1843-65.
- [60] Fernandez-Lozano, C., Gestal, M., Gonzalez-Diaz, H., Dorado, J., Pazos, A., Munteanu, C. R., 2014a. Markov mean properties for cell death-related protein classification. *J Theor Biol* 349, 12-21.
- [61] Ashburner M, Ball CA, Blake JA, Botstein D, Butler H, Cherry JM, Davis AP, Dolinski K, Dwight SS, Eppig JT, et al. Gene ontology: tool for the unification of biology. *Nature Genet.* 2000; 25(1):25. <https://doi.org/10.1038/75556>.
- [62] Altschul SF, Madden TL, Schäffer AA, Zhang J, Zhang Z, Miller W, Lipman DJ. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.* 1997; 25(17):3389–402. <https://doi.org/10.1093/nar/25.17.3389>.
- [63] Gehring J, Auli M, Grangier D, Yarats D, Dauphin YN. Convolutional sequence to sequence learning. 2017. <https://arxiv.org/abs/1705.03122>.
- [64] Hu X, Dong Q, Yang J, Zhang Y. Recognizing metal and acid radical ion binding sites by integrating ab initio modeling with template-based transferals. *Bioinformatics.* 2016; 32(21):3260.