Chapter 1

Introduction

1.1 Brief Introduction

Proteins are one of the most important biomolecules and play a critical role in almost all biological beings/bodies. It is estimated that the human body can generate upwards of 2 million different proteins. The process of creating proteins is as important as their usage. Proteins are created of usually multiple amino acid chains. A single protein can be structured from over twenty different amino acid chains, frequently made up of long amino acid sequences which are in-turn held together with peptide bonds.

Understanding proteins and their relationships with amino acids has been an unending issue in molecular biology with significant implications throughout the scientific community. Using Machine Learning techniques, our goal is to learn the connection between amino acids (unaligned) and the function between and premost of the protein families as much as possible.

1.2 Problem Statement

What are the steps required to correctly classify a protein's sequence of amino acid to the correct protein families' accession as accurately as possible, given the databasee. The main purpose of the database is to ease the Pfam dataset as a multiclass machine learning classification task.

To state differently, if specified a sequence of amino acid, specific to a protein set, predict the sequence's protein class.

An example of Pfam families starting with E are shown in Table 1.1.

ID	Accession	Has 3D	Change status	Description
E1-E2_ATPase	PF00122	✓	Changed	E1-E2 ATPase
E1-N	PF14463		Changed	E1 N-terminal domain
E1_4HB	PF16191	✓	Changed	Ubiquitin-activating enzyme E1 four-helix bundle
E1_DerP2_DerF2	PF02221	✓	Changed	ML domain
E1_dh	PF00676	✓	Changed	Dehydrogenase E1 component
E1_FCCH	PF16190	✓	Changed	Ubiquitin-activating enzyme E1 FCCH domain
E1_UFD	PF09358	✓	Changed	Ubiquitin fold domain
E2	PF08199		Changed	Bacteriophage E2-like protein
E2F_CC-MB	PF16421	✓	Changed	E2F transcription factor CC-MB domain
E2F_TDP	PF02319	✓	Changed	E2F/DP family winged-helix DNA-binding domain
E2R135	PF11570	✓	Changed	Coiled-coil receptor-binding R-domain of colicin E2
E2_bind	PF08825	✓	Changed	E2 binding domain
E3_binding	PF02817	✓	Changed	e3 binding domain
E3_UbLigase_EDD	PF11547	✓	Changed	E3 ubiquitin ligase EDD
E3_UbLigase_R4	PF13764		Changed	E3 ubiquitin-protein ligase UBR4
E3_UbLigase_RBR	PF18091	✓	New	E3 Ubiquitin Ligase RBR C-terminal domain
E3_UFM1_ligase	PF09743		Changed	E3 UFM1-protein ligase 1
E6	PF00518	✓	Changed	Early Protein (E6)
E7	PF00527	✓	Changed	E7 protein, Early protein
E7R	PF17467		Changed	Viral Protein E7
EABR	PF12180	✓	Changed	TSG101 and ALIX binding domain of CEP55
Ead_Ea22	PF13935		Changed	Ead/Ea22-like protein
EAF	PF09816		Changed	RNA polymerase II transcription elongation factor
Eaf7	PF07904	✓	Changed	Chromatin modification-related protein EAF7
EAGR_box	PF16713	✓	Changed	Enriched in aromatic and glycine Residues box
EAL	PF00563	✓	Changed	EAL domain
EamA	PF00892	✓	Changed	EamA-like transporter family
EAP30	PF04157	✓	Changed	EAP30/Vps36 family
Eapp_C	PF10238		Changed	E2F-associated phosphoprotein
EAR	PF07897		Changed	Ethylene-responsive binding factor-associated repression
EarP	PF10093	✓	Changed	Elongation-Factor P (EF-P) rhamnosyltransferase EarP
EAV_GP5	PF15981		Changed	Envelope glycoprotein GP 5 of equine arteritis virus
EAV_GS	PF01309		Changed	Equine arteritis virus small envelope glycoprotein
EB	PF01683		Changed	EB module

Table 1.1 A set example of Pfam families starting with the letter E.

ID – Entry name, **Accession** – The unique identifier, **Has 3D** – Is a 3D structure available?

Change status - Any update to the family in new version, **Description**

Chapter 2

RELATED WORK & RESEARCH

2.1 Background Research

While the intertwining of Machine Learning and Biotechnology has been quite scarce so far, it does provide a new avenue toward new and improved research and progression methods for accurately predicting and determining different sets of data for the many streams in Biotechnology. The interest for in this project spiked when I learned about a research group lead by Dr. Gagan Deep from Wakeforest School of Medicine. He was part of the Cancer Biology department in the School and was keen on providing an internship role in his new research project aimed at collaborating the streams of computer science and biotechnology to identify consistent patterns and create algorithms to categorize these patterns in the data. The way these patterns are calculated are using the algorithms on the observed exosomes and their content (protein, RNA, lipids and metabolites). [1]

Although this project was not carried forward very keenly after a month of work, the idea of working on similar projects was still intriguing after we discovered many articles and papers published on similar topics. As our focus was on analyzing and classifying different inputs to the correct categories, we found the Pfam dataset, which provides complete and accurate classification of protein families and domains. [2]

The deep learning model that we decided to use on this dataset was the ProtCNN (CNN Model). Previous state of the art approaches included algorithms such as HMM and BLASTp. The main issues with using techniques such as BLASTp is that their computational costs outweigh their efficiency and accuracy, especially since BLASTp's scales linearly with the size of the dataset, which grows exponentially. [3]. Although these processes have been quite successful so far, almost one-third of the proteins haven't been successfully annotated through all the characterized sequences. [4] . Another great algorithm that we found that could have potential was Needleman and Wunsch. They had created a dynamic programming algorithm which found similarities between DNA and protein sequences [5]. This was not considered as it was a bit outdated.

The advantage of using deep leaning models instead of the aforementioned algorithms is that deep learning should provide an avenue to circumvent these hindrances and also directly predict all the annotations belonging to proteins. The model can learn the distribution between the multiple classes of proteins while evaluating them at the same time. [6].

2.2 Related Works

2.2.1 BLASTp

Stephan 1990 [1], synthesize a way of rapid sequence comparison or basic local alignment search tool (BLAST) which directly approximates alignments that optimize a measure of local similarity, the maximal segment pair (MSP) score. BLAST finds regions of similarity between biological sequences. The basic logic behind it is to compare the protein sequences to sequence databases and calculate the statistical significance. BLASTp (Protein BLAST) compares one or more input protein sequences to one or more protein sequences in the database. BLASTp, can be implemented in various ways and can also be applied to different scenarios such as straight-forward DNA and protein sequence database searches, motif searches, gene identification searches, or also in the analysis of multiple regions of similarity in long DNA sequences. In addition to its flexibility and tractability to mathematical analysis, BLAST is an order of magnitude faster than a number of sequence comparison tools of comparable sensitivity.

2.2.2 Profile HMM Models

Eddy 2011 [2], described an accelerated heuristic known as the ‘‘multiple segment Viterbi’’ (MSV) algorithm for profile Hidden Markov Models (HMMs). Profile HMMs and probabilistic inference methods had already made important contributions in this field. However, practical use of profile HMM methods had been hindered by the computational expense of existing, at the time of 2011, software implementations but the results were still not satisfactory. The way MSV algorithm works is that it computes an optimal sum of multiple ungapped local alignment segments using a striped vector-parallel approach which was previously described and used for fast Smith/Waterman alignment. MSV scores follow the same statistical distribution as gapped optimal local alignment scores, allowing rapid evaluation of significance of an MSV score and thus facilitating its use as a heuristic filter which in turn cuts down on the training time and resources which is required to train the model.

Chapter 3

Requirement Analysis

We understand that Requirement analysis is a critical factor for the success of a project. There were multiple decisions that we had to take throughout the duration of our project, especially with the nation-wide lockdown limiting resources and time. Most of the agreements that were formed surrounding our project were altered later as a consequence of the lockdown.

3.1 Software/Hardware Requirements

Table 3.1: Project Setup (pre-lockdown)

#	Project Setup (pre-lockdown)
1	Windows 10, Python, Nvidia DGX, Local PC
2	Standards that must be followed (default coding standard, etc.)
3	Special access privileges needed, terms & conditions, release to open source, etc.

Our decisions after the lockdown, note changes to cloud-based services

Table 3.2: Project Setup (post-lockdown)

#	Project Setup (post -lockdown)
1	Windows 10, Python, Microsoft Azure, Google Colab, Python Libraries, Local PC
2	Standards that must be followed (default coding standard, etc.)
3	Special access privileges needed, terms & conditions, release to open source, etc.

3.2 Stakeholders

Table 3.3: Stakeholders

Stakeholder	Role
Dr. Suneet Kumar Gupta	Mentor
Dr. Deepak Garg	Mentor
Vastav Kalia	Manager, Developer, Tester
Anidya Vedant	Manager, Developer, Tester

3.3 Project Tracking

Information	Description
Code Storage	Project code will be stored in GIT repository.
File Sharing	Files shared through GIT repository and Cloud Services like Google Drive etc.
Project Documents and Assignments	Semester reports (milestone reports), video, poster and other project related documents stored on Microsoft OneDrive & YouTube

3.4 Validation of Requirements

Since the main goal is to meet user needs, we will verify that our model is adequate to deliver the necessary results. Any misinterpretations that a user might have regarding the requirements should be "validated". Even though this step is highly inefficient for the two of us in this project, it leads to substantial drop development risks. We will validate the following points as required.

i. The diagrams, tables and descriptions are consistent.
ii. The system should act as expected during development.
iii. The user should not feel scammed or be dissatisfied with the properties of the project.
iv. Finally, the project should follow the set schedule as per the guidelines set by the college.

Chapter 4
Proposed Approach and Solution

4.1 Proposed Solution

The project's main aim is to be able to correctly classify (or predict) an amino acid sequence to the correct family of proteins it belongs to. By using different models on the given dataset (e.g CNN (ProtCNN) and LSTM), we should be able to match the amino acid sequences which are unaligned with the annotations through the collection of 17929 protein families in the Pfam database.

If required, two to three models could be trained if desired accuracy and classification is not achieved. If significant results are achieved, machine learning models will be able to become core components of prospective tools for protein prediction tools.

4.2 Goals and Objectives

4.2.1 Main Goal: Finding a good machine/deep learning model as an alternative to an already established method (BLASTp) to better classify the relations between proteins and their amino acid sequences.

4.2.2 Objectives: The first objective was to understand the different classifications of proteins and the types of amino acid sequences that could be related to a specific protein. Since the database was labelled, we were able to observe the number of unaligned amino acid sequences, which were usually in the range of 50-250. The next objective was to filter out the potential deep learning models that will have the potential to provide pristine accuracies and be efficient at the same time.

Combining two to three different techniques for each model will help in reducing time and effort required for correctly running the dataset. The major objective will be to achieve accuracies greater than the already implemented methods (BLASTp) and will hopefully also reduce the time required. Even though we are able to achieve higher accuracies, it should not be at the cost of considerable loss of time, which means increased efficiency, if possible.

4.3 ProtCNN

Here we have a look at the input, embedding and prediction networks that constitute almost all the deep learning models. The thing to notice here is that the first and the last part of the model (i.e. input and prediction networks) essentially have the same functional form for all the models.

The input maps a sequence of L amino acids (1-D) to a L x 20 binary array (2-D). The columns of Lx20 are one-hot encoding of the input amino acid. For example, A is represented are [1,0,0, . . .] and similarly, ADE is also represented as a set of one hot encoded vectors

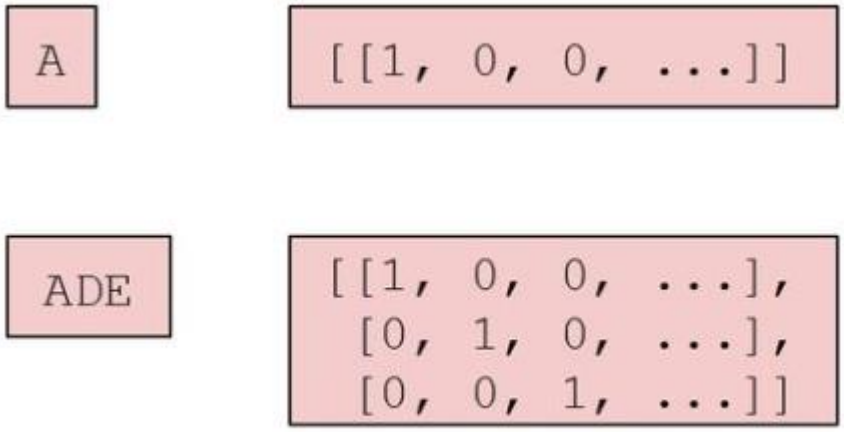


Figure 4.1- One hot Encoding of input vector

The encoded one hot vector is then padded to the length of the longest sequence in the batch with all-zero vectors appended on the right. For example, the vectors above are padded with zero like mentioned below

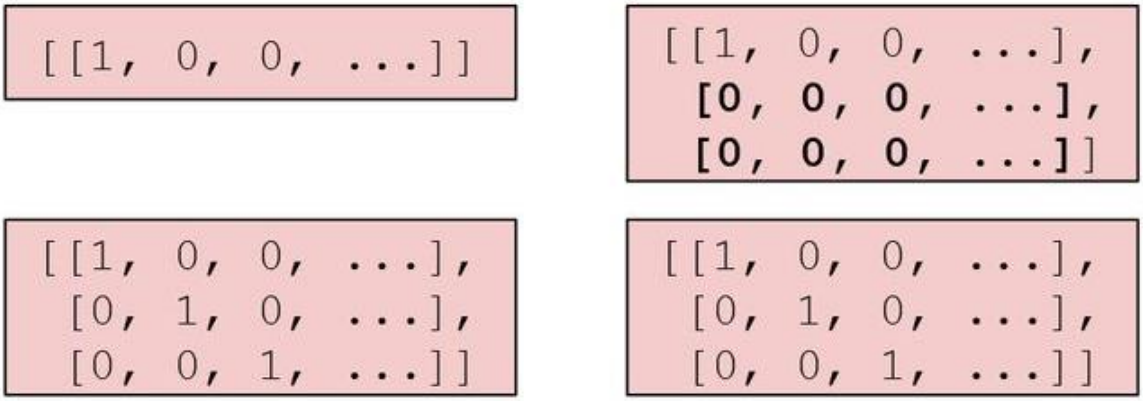


Figure 4.2- Encoded vectors with padded zeroes

The next network, that is the embedding network maps the Lx20 array (containing the padded one hot encoded vectors) to a bigger LxF array which contains embedding for each and every sequence. Here F is a tunable parameter (the default value of F for ProtCNN is 1100). For example, the padded one hot encoded vectors in the Figure are embedded as below

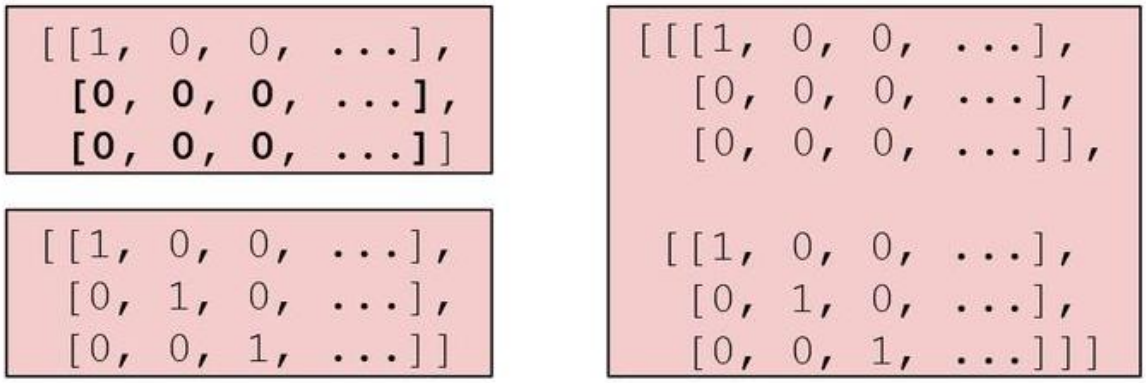


Figure 4.3- Batch of embbeded one-hot encoded vectors

For embedding purposes, ProtCNN uses residues of convolutional neural network (ResNets), which means that each layer feeds into the next layer and directly into the layers about 2–3 hops away. For a better understanding refer the image of a single residual block in Figure .

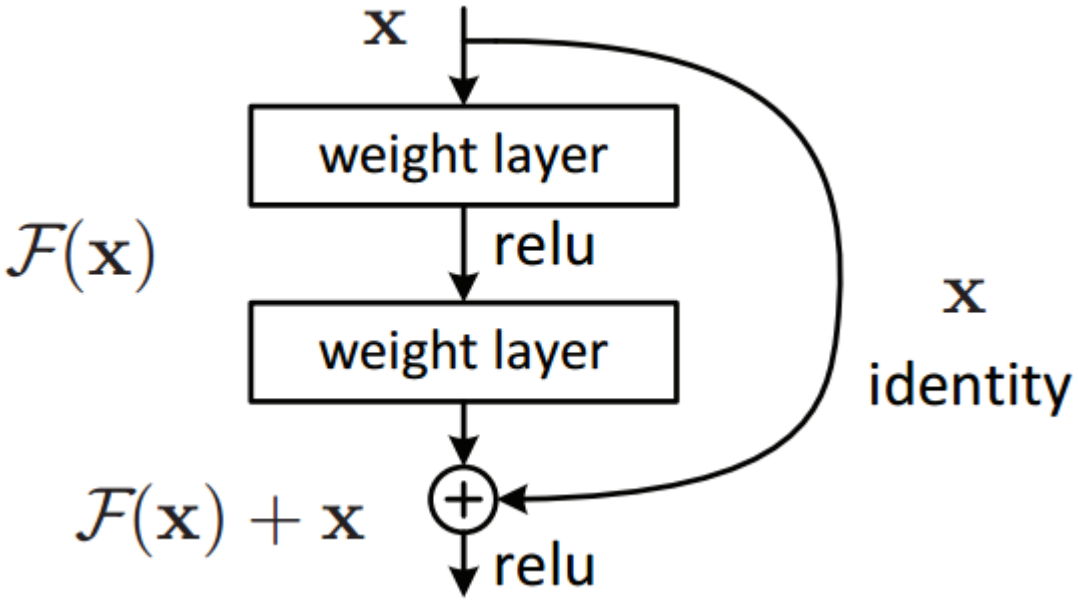


Figure 4.4- Single Residual Block

For this project we have used ResNets with dilated convolutions, the architecture of which is depicted in Figure . Dilated convolutions is just a convolution applied on the input but with defined gaps or applied after being dilated (See *Figure*)

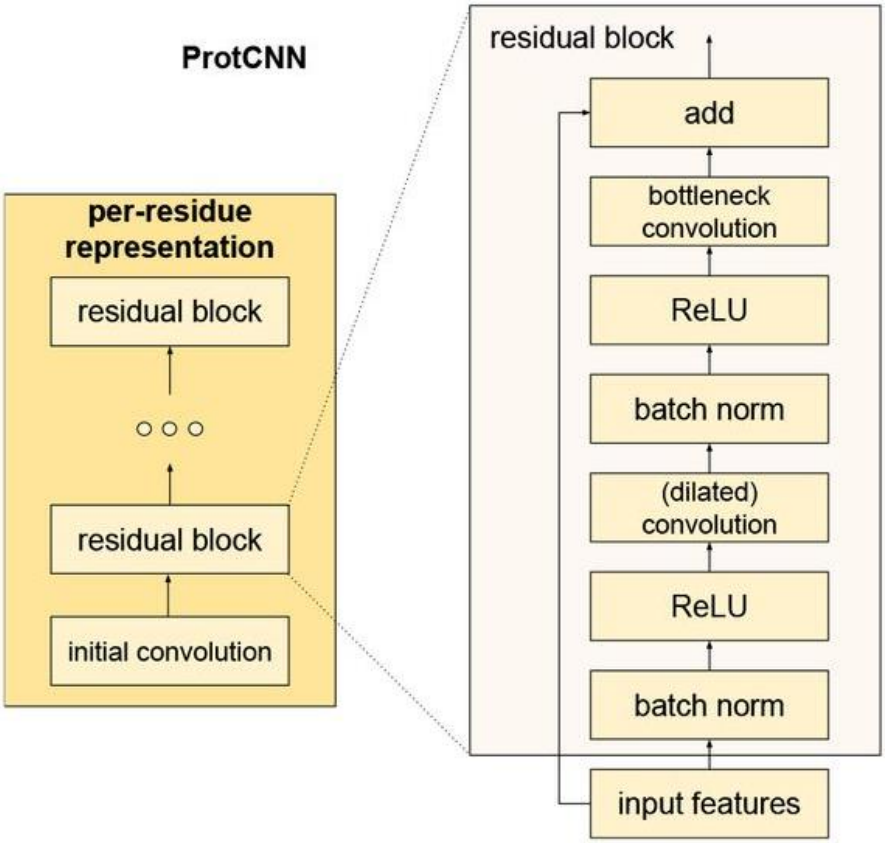


Figure 4.5- ResNet architecture used in the project

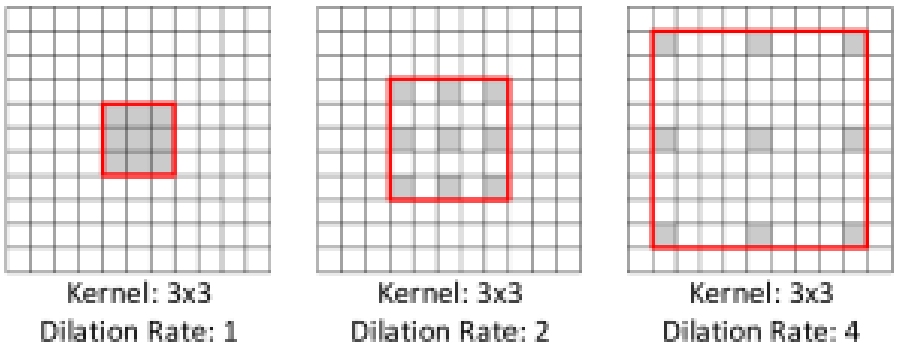


Figure 4.6- Dilated Convolutions

All in all, to summarize and give a short overview of model and the entire functioning of the model we can have a look at the Figure .

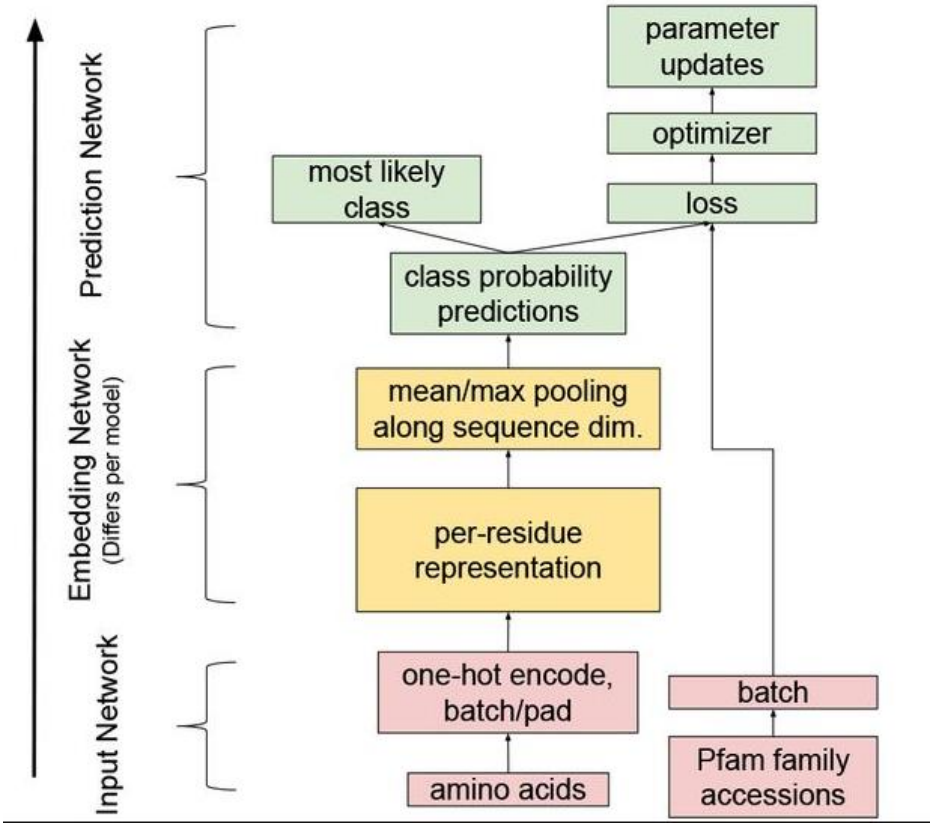


Figure 4.7- Overview

Chapter 5
Project Planning and Execution

5.1 Project Setup

The project utilize deep and machine learning models and results are compared between each of these models. Google Colab/ Microsoft Azure were used for testing the code while a local machine was preferred for training the models as the files were quite large with many different libraries required to be installed manually. Our initial plan was to use the DGX servers at college but due to the nationwide lockdown in India, we switched to training some trial models on Microsoft Azure as well due to lack of resources at home.

5.2 Dataset Used

The main dataset used for this project was the Pfam dataset (v 32.0, released September 2018) and contains 17,929 families. The database, including protein families and multiple sequences which were all generated using hidden Markov models. We were provided with five features namely:

- **sequence:** This is mostly the input features for the models. Sequence here refers to the amino acid sequences, a total of 24 (4 uncommon).

- **family_accession:** This acts as the label to the models. Data here is represented in the form of PFxxxxx.y.

- **sequence_name:** Sequence name, in the form "uniprot_accession_id/start_index-end_index".

- **aligned_sequence:** The single sequences are listed here (not of multiple alignment).

- **family_id:** Names of the families (one word).

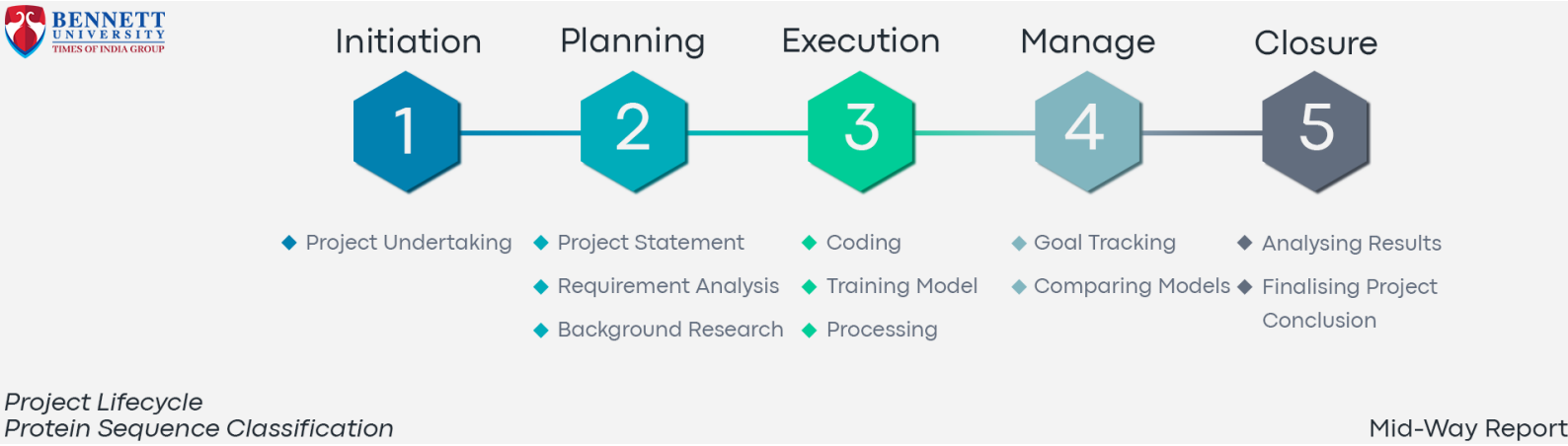
5.3 Project Lifecycle

As the project is mostly research based, there isn't an apparent lifecycle. If we follow a research project's process, the first task according to that was to decide what project we would be willing to undertake and work on, with an extensive background research and good statement. The total requirements were also calculated based specifically on the software and hardware end. Our model training and coding was next with our current progress being in this area. We have been successful in training other models despite some setback due to Covid-19 lockdown, but the progress is acceptable.

Chapter 6

Design diagrams/ UML diagrams/ Flow Charts/ E-R diagrams

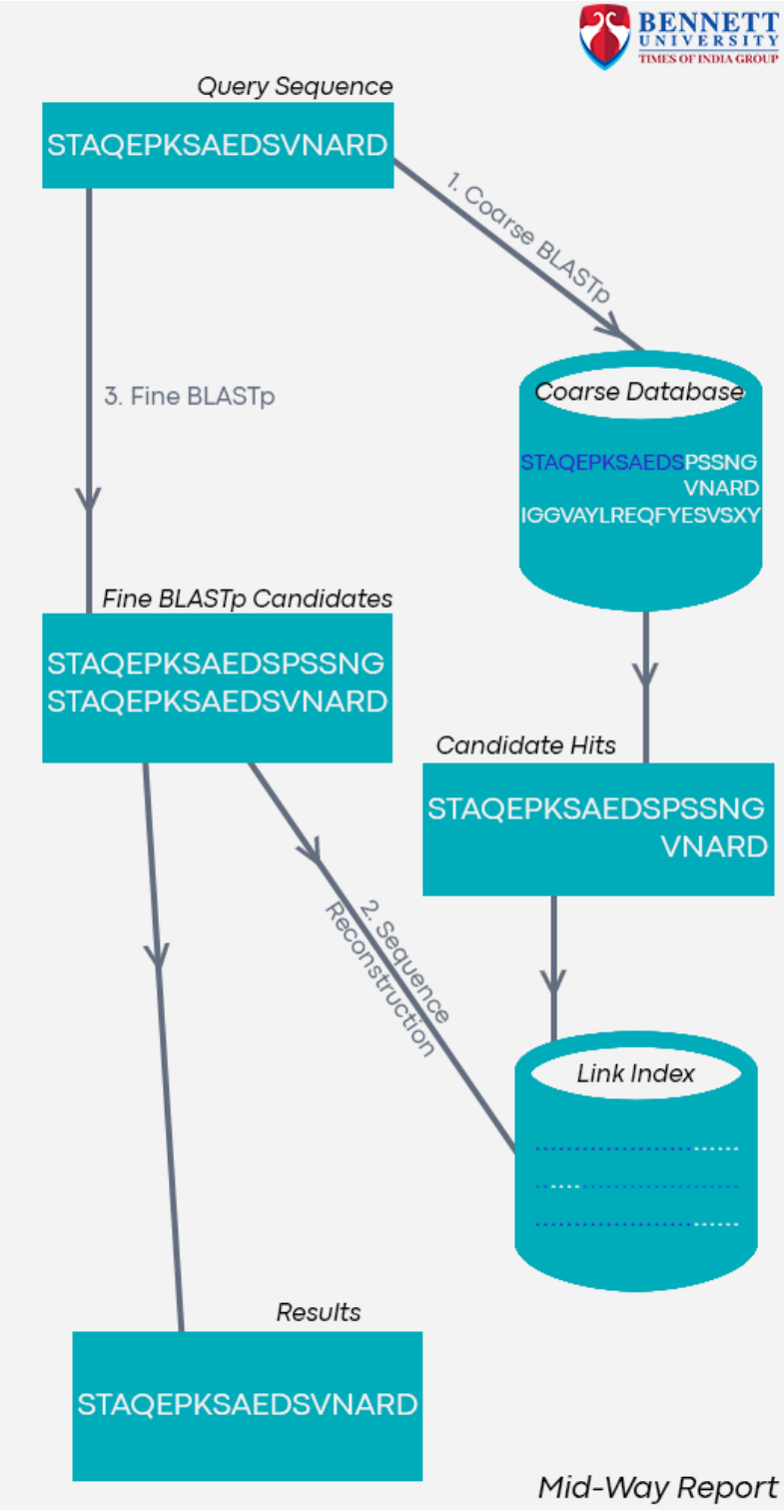
6.1 Project Lifecycle



Project Lifecycle
Protein Sequence Classification

Mid-Way Report

6.2 Flow Chart of BLASTp Sequence Matching



Two stage search phase (coarse and fine) for BLASTp

Chapter 7

Result Comparison and Analysis

We used the Pfam Dataset, as a benchmark and compared the results of deep learning models with already existing state-of-the-art alignment based methods such as BLASTp [1] and profile HMM models [2]. The dataset is split in two different ways, the first being the random split of the 17929 families in 80-10-10 partition meaning, 80% training split, 10% validation split and 10% training split. The other split being a clustered split which is done specifically for the distant homologs. Some of the protein sequence may be one of the outliers and random split does not exactly train the model for the outliers and thus, to train the model on the outliers, we performed a clustered split and then trained the models on the clusters.

One thing to keep in mind is that since proteins generally are related through evolution, it causes a wide variety of similarities to occur across different sequences which when split randomly land across different sets and in turn cause similarities to occur over different splits. To eradict this issue we factor in the maximum percent identity of each test sequence with sequences in the train test.

For 30-90% maximum identity with the training set, our model ProtCNN performed significantly better than the alignment-based methods. It was also observed that grouping several ProtCNN and then taking the majority of the predicted outcome also has an effect on the results. Furthermore, it was also observed that although a lot of different groupings affected the results, a group of 13 ProtCNN model works best. It was labelled as ProtENN. ProtENN is significantly more accurate than alignment-based methods for all sequence bins with less than 90% identity to the training set. Using the held-out test error rate as a function of the maximum percent sequence identity across all training sequences we got the results as mentioned in Figure and Figure 7.2

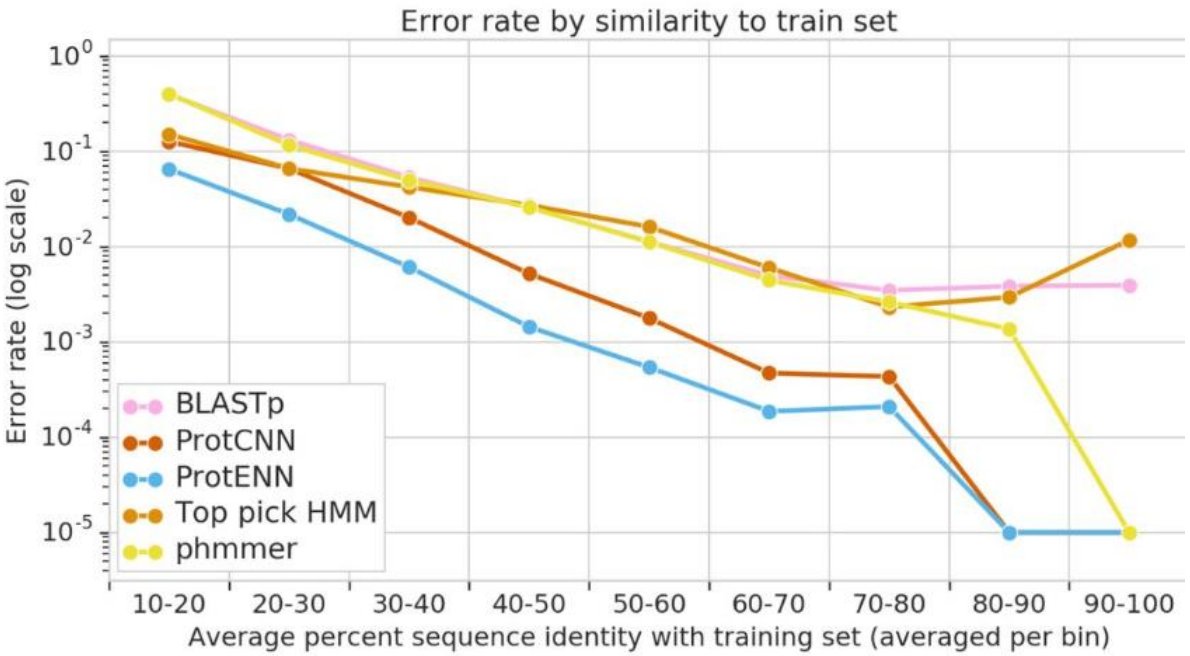


Figure 7.1- Random split results

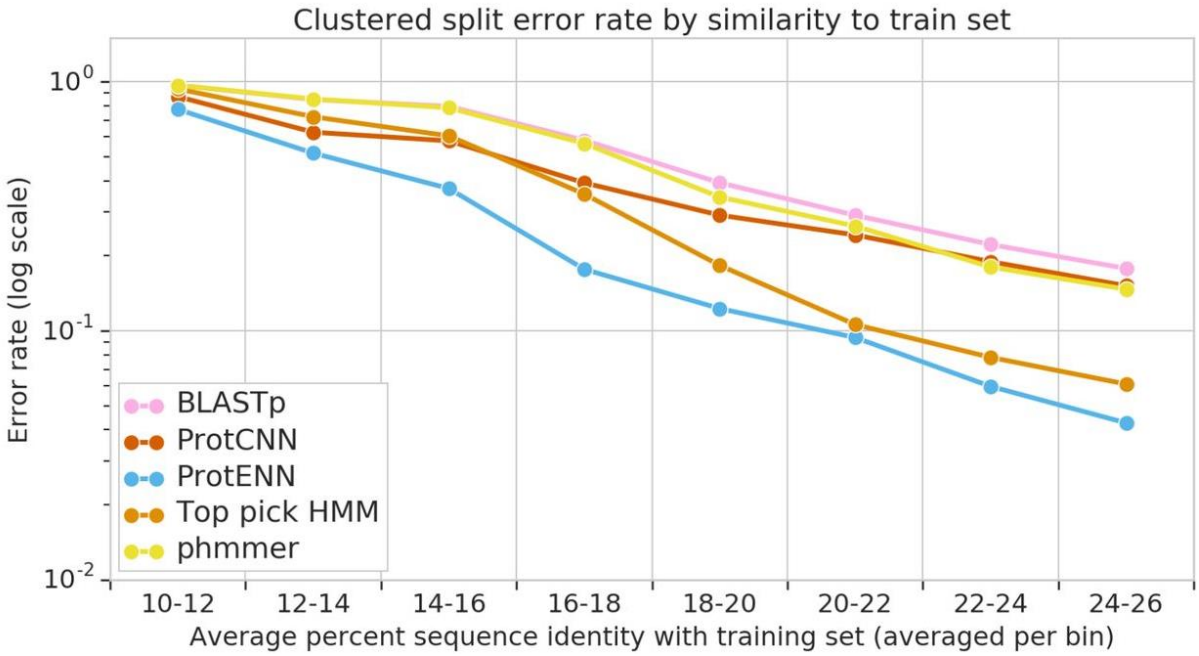


Figure 7.2- Clustered Split Results

As you can probably see from Figure , both the deep learning models, i.e. ProtCNN and ProtENN make fewer errors than the already existing models such as BLASTp and HMM.

Figure 7.2 shows that although Top pick HMM did perform better than ProtCNN, it was actually the ProtENN model which was able to deal with the clustered data in the best way possible and emerge victorious when pitched against other methods.

A numerical analysis of the above observation can be seen in the Table below.

Model	Error rate	Number of errors
Top Pick HMM	18.1%	3844
phmmer	32.6%	6942
BLASTp	35.9 %	7639
ProtCNN	27.7 %	5882
ProtENN	**12.2%**	**2590**

Table 7.1 Error rate of different Models