CAFA 5 Protein Function Prediction

Name: Aadvik Kumar **IITK Email**: aadvik21@iitk.ac.in **Kaggle ID**: NekoClaviSS	**Roll No**: 210002 **Department**: BSBE **Kaggle Leaderboard Rank**: 332

Tasks	PPT	Github Repo

Introduction

The problem statement of the competition is to train a **multi-label classification** model using the data given in **UniProt's Gene Ontology**. The Gene Ontology (GO) is stored in a structured format that allows for efficient querying and analysis. The GO data is typically stored in a relational database, where each term in the ontology is represented as a row in a table. The problem is that of a multi-label because there is a single output and we have to predict one or more functional annotations for that sequence. The output will be a set of binary labels indicating the presence or absence of each GO term.
The model trained would then be used to determine the function of the protein sequence that we input into the model.

Literature Review

Approaches Other than ML

1. **Sequence Homology** - Sequence homology refers to the similarity between protein sequences. Proteins with **similar sequences** are likely to have **similar functions**. This approach assumes that proteins with similar amino acid sequences are likely to have **similar structures and functions**. It is based on the idea that evolutionarily related proteins tend to retain similar functions even if their sequences have diverged over time.
   Example: Let's say we have a newly discovered protein sequence "**ABCDEF**" with an unknown function. By comparing this sequence to a database of known protein sequences, we find a highly similar sequence, "**ABCDEG**" with the known function of **binding to DNA**. Based on the sequence homology, we can predict that the newly discovered protein "**ABCDEF**" might also have a similar **DNA binding function**.
2. **Structural Homology** - Structural homology involves comparing the **three-dimensional structures** of proteins. Proteins with **similar structures** are likely to have **similar functions.** This method assumes that the overall fold and arrangement of protein domains are conserved among functionally related proteins.
   Example: Consider a protein with an unknown function but a known structure. By comparing its structure to a protein of known function and similar structure, we can

infer that the unknown protein might have a similar function. For instance, if the unknown protein has a **similar structural fold** to an enzyme involved in catalysis, we can predict that the unknown protein might also have enzymatic activity.

3. **Domain Analysis** - Proteins are composed of **functional** and **structural units** called domains. Domain analysis involves **identifying** and **characterising** these domains within a protein sequence. Domains often have specific functions and can be found in different proteins with similar functions. Analysing protein domains helps predict the overall function of a protein.
   Example: Let's say we have a protein sequence "**ABCDE**" with an unknown function. Through domain analysis, we identified that it contains a DNA-binding domain. Based on this information, we can predict that the protein "**ABCDE**" is likely involved in DNA binding or related processes.

Gene Ontology

1. The Gene Ontology (GO) is stored in a structured format that allows for efficient querying and analysis. The GO data is typically stored in a relational database, where each term in the ontology is represented as a row in a table
2. There are several tables in the GO,
    1. **Term Table** → Info about each term in the ontology, like,
        - ID: a unique identifier for the GO term (e.g. "GO:0008150" for "biological process")
        - Name: the name of the GO term (e.g. "biological process")
        - Namespace: the ontology namespace that the GO term belongs to (e.g. "biological_process", "molecular_function", or "cellular_component")
        - Definition: a brief description of the meaning of the GO term
    2. **Relationship Table** → Relationship between GO terms. Each row is like a directed edge of a graph.
        - Parent ID: the ID of the parent term in the relationship
        - Child ID: the ID of the child term in the relationship
        - Relationship type: the type of relationship between the parent and child terms (e.g. "is_a", "part_of", or "regulates")
    3. **Gene Association Table** → Contains annotations of → Linking genes/gene products to GO terms.
        - Gene ID: a unique identifier for the gene or gene product being annotated
        - GO ID: the ID of the GO term that the gene or gene product is associated with
        - Evidence code: a code indicating the type of evidence used to support the annotation (e.g. "experimental", "computational", or "curator")
        - Qualifier: a qualifier indicating whether the annotation represents a direct or indirect association between the gene and the GO term

BLAST

BLAST or **Basic Local Alignment Search Tool**, helps in identifying similar sequences, detecting **homologous relationships**, and inferring functional and evolutionary information. BLAST works by breaking down the query sequence into smaller segments called "words" and searching for matches in a database of sequences. The algorithm quickly identifies short, exact matches between the query and the database sequences, known as "seeds." These seeds are then extended to find longer alignments using a scoring system based on sequence similarity. Finally, BLAST ranks the alignments based on their statistical significance and produces a list of high-scoring matches.

DIAMOND (**DIscOvers Motifs of Natural Domains**), a program commonly used in combination with BLAST for protein sequence comparison. DIAMOND is designed to provide **faster** sequence alignments by utilising a more efficient algorithm based on a heuristic approach called "**seed-and-extend**." When used with BLAST, DIAMOND speeds up sequence comparisons, especially for large-scale analyses.

BLAST Tool - https://blast.ncbi.nlm.nih.gov/Blast.cgi
Insight to Blast - https://www.nature.com/scitable/topicpage/basic-local-alignment-search-tool-blast-29096/
Diamond is a sequence aligner implementation through which we can apply the BLAST algorithm. It is an optimised implementation that greatly speeds up pairwise alignment of proteins and translated DNA at 100x-10,000x speed of BLAST.
Github - https://github.com/bbuchfink/diamond
A kaggle competition implementation for BLAST - https://www.kaggle.com/code/geraseva/diamond

Embeddings

Embeddings are representations of data in a lower-dimensional space that capture important features and relationships. In the context of protein sequence analysis, embeddings are **numerical** representations of words or amino acids that can be used to model and analyse textual or biological data.

ProtBERT

ProtBERT is a deep learning-based embedding model specifically designed for protein sequences. It is built upon the architecture of **BERT** (Bidirectional Encoder Representations from Transformers), which is widely used for natural language processing tasks. ProtBERT is pre-trained on large-scale **protein** sequence databases, allowing it to capture protein sequence features and learn representations that encode functional information. These learned embeddings can be used as input for downstream protein function prediction tasks.

T5

T5 (Text-To-Text Transfer Transformer) is a versatile transformer-based model developed by **Google**. Although originally designed for natural language processing, T5 can be applied to protein function prediction by treating the protein sequences as text. By fine-tuning T5 on protein sequence data, it can learn to generate embeddings that capture relevant features for function prediction. T5 can be adapted to various prediction tasks, including protein function annotation and prediction of protein-protein interactions.

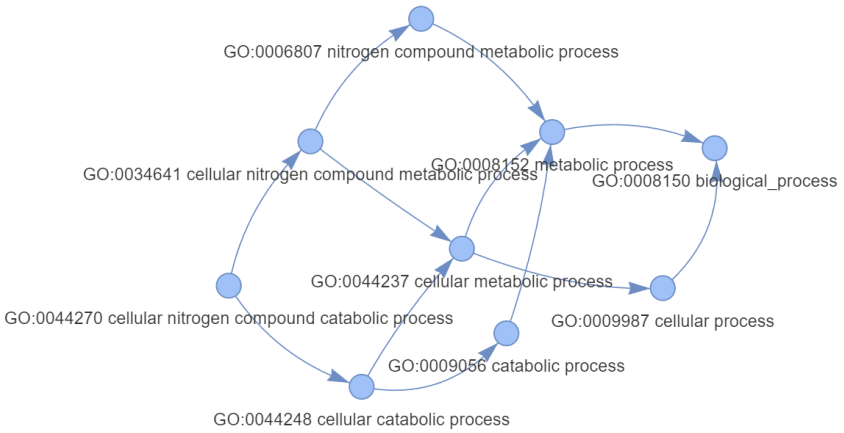
ESM-2

ESM-2 (Evolutionary Scale Models 2): ESM-2 is a deep learning-based embedding model developed by **Facebook AI Research**. It is trained on a large dataset of protein sequences and evolutionary information derived from **multiple sequence alignments**. ESM-2 captures both the primary sequence information and evolutionary conservation patterns, which are crucial for understanding protein structure and function. The embeddings generated by ESM-2 can be used as input for **downstream** tasks such as protein function prediction and protein-protein interaction prediction.

Experimentation

EDA: Link

Exploratory Data Analysis (EDA) is a crucial step in the data analysis process, as it helps us understand the underlying **patterns**, **relationships**, and **characteristics** of the data we are working with. So, we performed some EDA on the given GO dataset and observed a few key points about the data, which are as follows:



1. We observed that the GO Terms are in fact stored as **Directed Acyclic Graphs**. For example, the above graph is for the GO Term: `"GO:0034655",` which is related as,

```
{'name': 'nucleobase-containing compound catabolic process',
 'namespace': 'biological_process',
 'def': '"The chemical reactions and pathways resulting in the breakdown of nucleobases, nucleosides, nucleotid
es and nucleic acids." [GOC:mah]',
 'subset': ['goslim_chembl'],
 'synonym': ['"nucleobase, nucleoside, nucleotide and nucleic acid breakdown" EXACT []',
  '"nucleobase, nucleoside, nucleotide and nucleic acid catabolic process" RELATED [GOC:dph, GOC:tb]',
  '"nucleobase, nucleoside, nucleotide and nucleic acid catabolism" EXACT []',
  '"nucleobase, nucleoside, nucleotide and nucleic acid degradation" EXACT []'],
 'is_a': ['GO:0006139',
  'GO:0019439',
  'GO:0044270',
  'GO:0046700',
  'GO:1901361']}
```

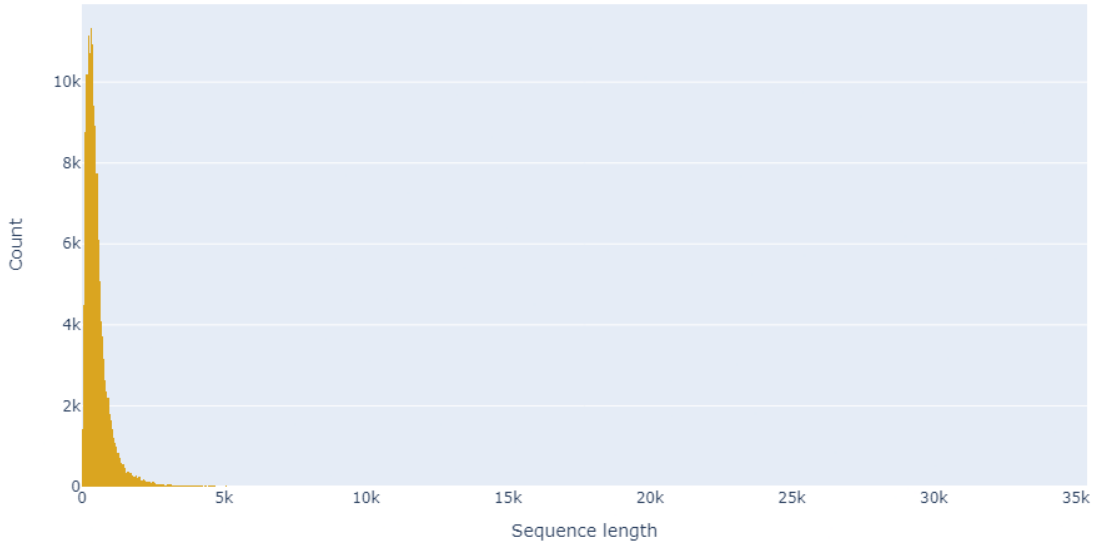

2. Some of the other notable data we observed were:

General:

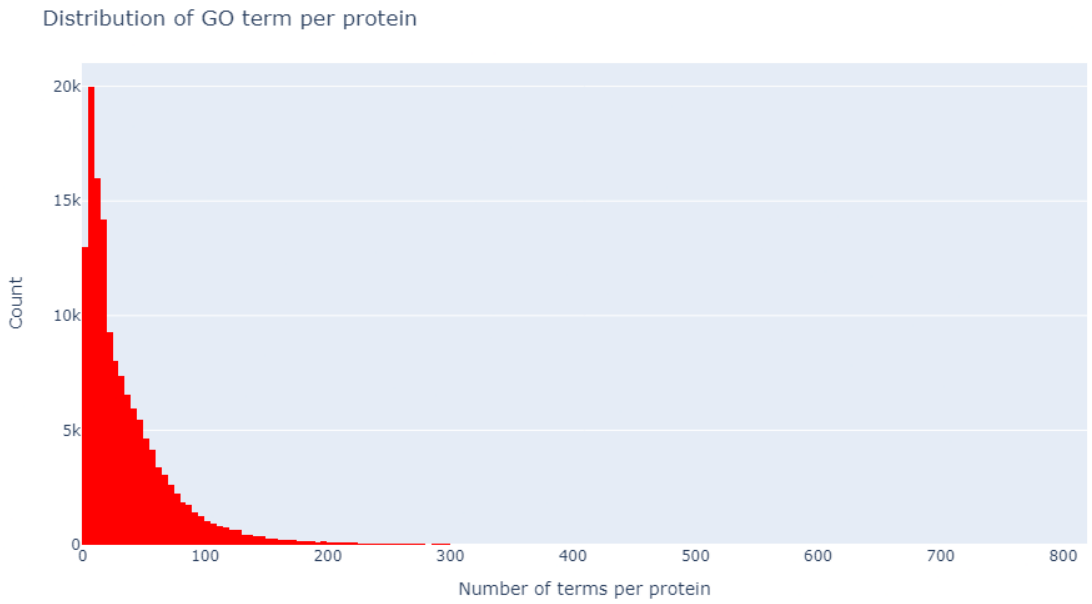
Number of Nodes	43248
Number of Edges	84805
Number of Unique Proteins	142246
Number of Unique GO Terms	31466

Sequence Length:

Number of Sequences	142246
Average Length of Sequences	553.64
Median Length of Sequences	411
90%ile of Length	1054

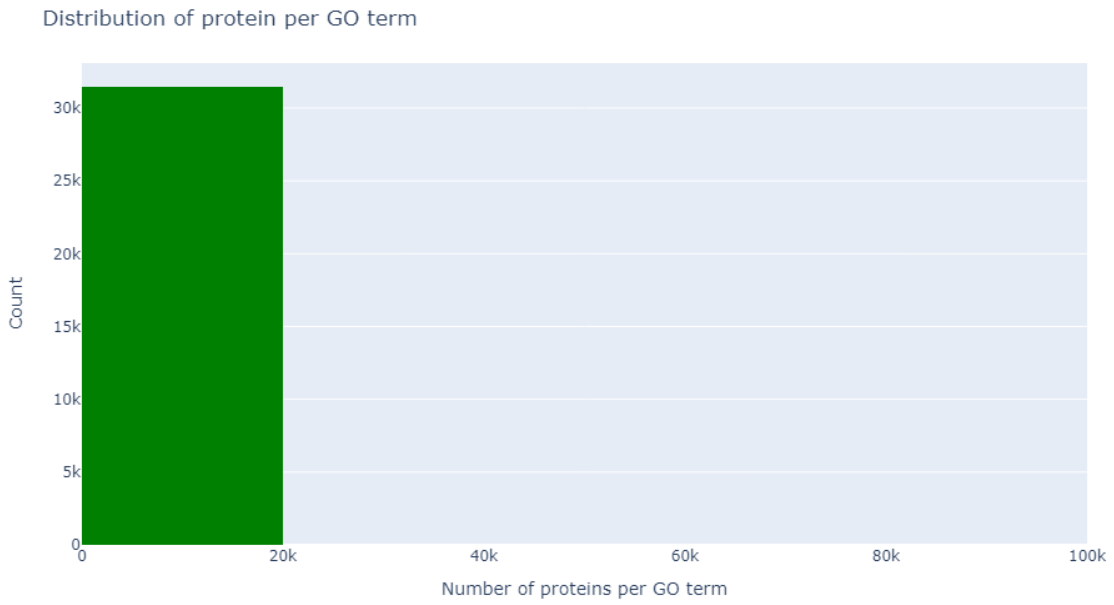


Distribution of GO Term per Protein:



Statistic	Terms per Protein
Minimum	2
Median	24
Mean	38
Maximum	815

Distribution of Proteins per GO Term:



Statistic	Proteins per GO Term
Minimum	1
Median	8
Mean	170
Maximum	92912

Through this EDA we were able to see the relationships between different GO terms as well as how much protein length can we safely **clip** or **pad** if we are to use transformer models. We were also able to see the **dependence** of GO terms on protein and vice versa.

BLAST: Link

Due to the huge size of the Sequence query it was not possible to use BLAST normally using the **Biopython** package. Hence, made use of another python library called **Diamond**, which is known for handling large amount of data

For sequence matching queries, we made use of sequences provided by *testsuperset.fasta* in the CAFA5 PFP dataset.

After performing sequence matching with BLAST, we store the matches in a Pandas dataframe.

	qseqid	sseqid	pident	length	mismatch	gapopen	qstart	qend	sstart	send	evalue	bitscore
0	Q9CQV8	Q9CQV8	100.0	246	0	0	1	246	1	246	1.100000e-167	464.0
1	Q9CQV8	P35213	98.8	246	3	0	1	246	1	246	1.050000e-165	459.0
2	Q9CQV8	P31946	98.8	246	3	0	1	246	1	246	2.120000e-165	458.0
3	Q9CQV8	V9HWD6	98.8	246	3	0	1	246	1	246	2.120000e-165	458.0
4	Q9CQV8	Q5PRD0	91.0	244	22	0	3	246	1	244	1.200000e-150	421.0
5	Q9CQV8	P63104	87.2	242	31	0	3	244	1	242	1.040000e-142	401.0
6	Q9CQV8	Q5ZKC9	86.8	242	32	0	3	244	1	242	2.970000e-142	400.0
7	Q9CQV8	P63101	86.8	242	32	0	3	244	1	242	5.990000e-142	399.0
8	Q9CQV8	P63102	86.8	242	32	0	3	244	1	242	5.990000e-142	399.0
9	Q9CQV8	P68254	81.0	242	46	0	3	244	1	242	4.080000e-133	377.0
10	Q9CQV8	P68255	81.0	242	46	0	3	244	1	242	4.080000e-133	377.0
11	Q9CQV8	P27348	81.0	242	46	0	3	244	1	242	4.080000e-133	377.0
12	Q9CQV8	P29310	78.2	243	53	0	2	244	3	245	6.820000e-128	363.0
13	Q9CQV8	Q20655	78.4	245	53	0	1	245	1	245	3.930000e-127	362.0
14	Q9CQV8	P41932	81.5	227	41	1	7	233	7	232	2.920000e-122	349.0
15	Q9CQV8	P61982	75.8	248	54	3	3	246	2	247	5.400000e-120	343.0
16	P62259	P62258	100.0	255	0	0	1	255	1	255	1.340000e-177	490.0
17	P62259	P62261	100.0	255	0	0	1	255	1	255	1.340000e-177	490.0
18	P62259	P62259	100.0	255	0	0	1	255	1	255	1.340000e-177	490.0
19	P62259	P62260	100.0	255	0	0	1	255	1	255	1.340000e-177	490.0

For finding the probability of the new predictions, We have considered matching with all the terms, as multiple GO terms can be associated with the same function.

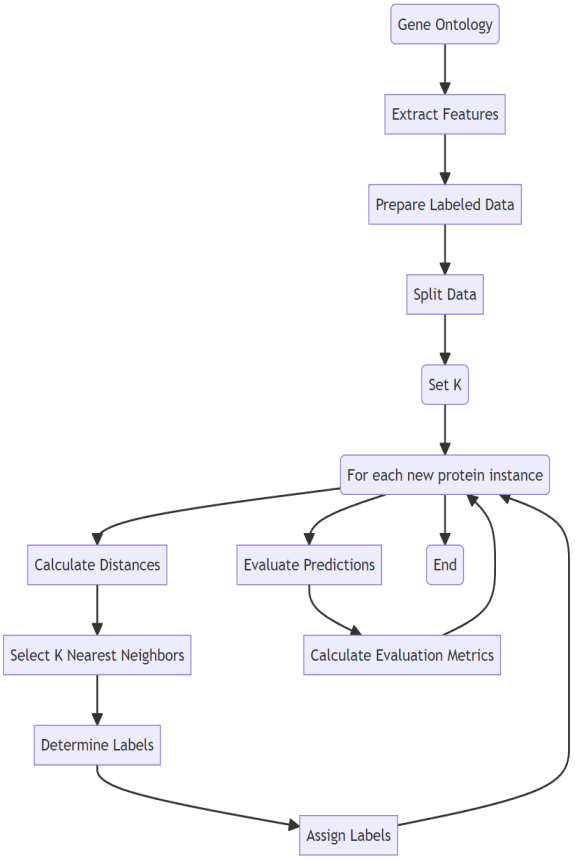
	qseqid	terms	index	ntargets
0	A0A023PXF5	GO:0000722	5019000	0.7
1	A0A023PXF5	GO:0000723	5019000	0.7
2	A0A023PXF5	GO:0003674	3584991	0.5
3	A0A023PXF5	GO:0003678	3584991	0.5
4	A0A023PXF5	GO:0003824	3584991	0.5

KNN: Link

K-Nearest Neighbors (K-NN) is a non-parametric machine learning algorithm used for classification and regression tasks. It predicts the label or value of a new data point by considering the K nearest neighbours from the training set. The algorithm calculates the **distances between the new data point and all the data points** in the training set, selects the K nearest neighbours based on the distance metric, and assigns the new data point to the most common class or averages the target values of its neighbours.

For Multi-Label Classification,

1. We first extracted the relevant features from the protein sequences.
2. Organized the data into feature vectors with corresponding labels.
3. We applied the KNN algorithm to train the model. During training, the KNN algorithm stores the **feature vectors** and labels from the training set.
4. We finally got the predictions from this model. When making predictions for a new protein, it calculates the **distances** between the new protein's features and the training set. The K nearest neighbours are selected, and their labels are used to assign protein function annotations to the new protein.



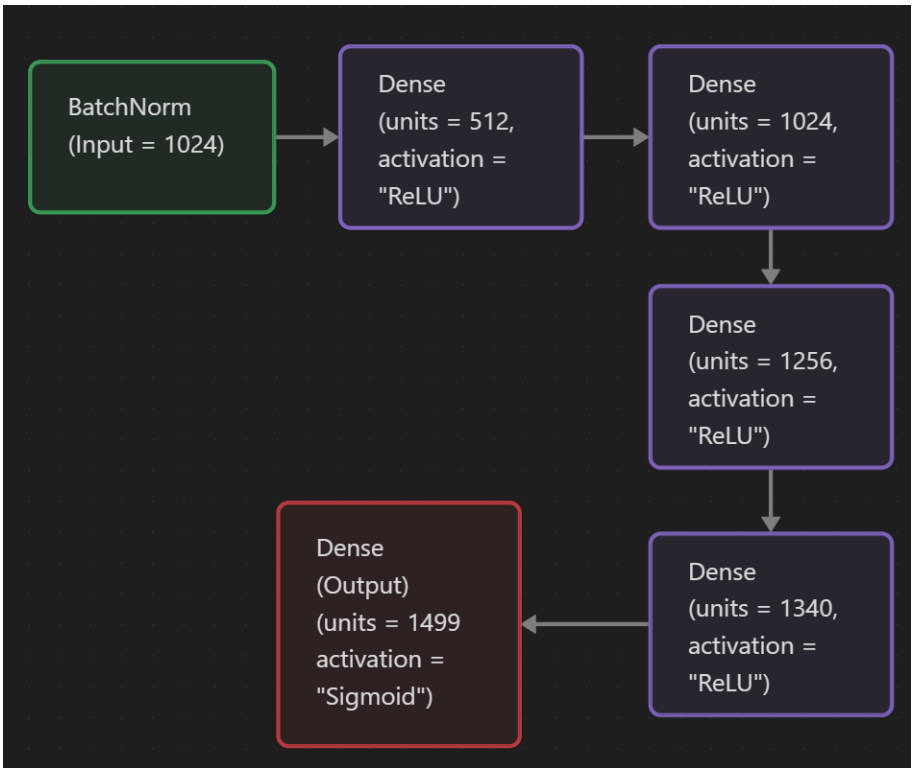
We got a disappointing F1-score of **0.059** in the submitted model and took over **1 hour** to get the predictions, which is undesirable.

Dimension Reductionality

We tried the **Autoencoder method** for dimensional reduction and then used the generated labels with KNN model.

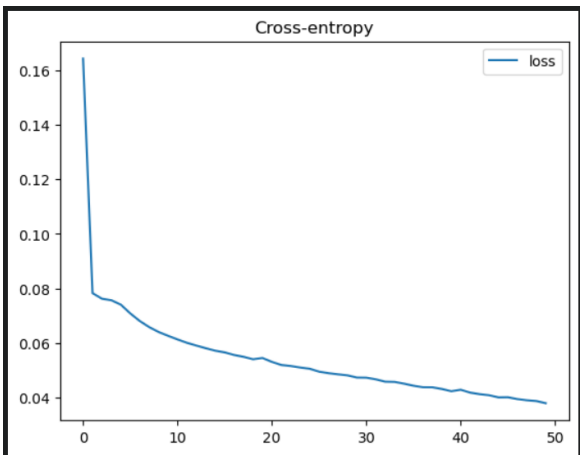
MLP with T5: Link

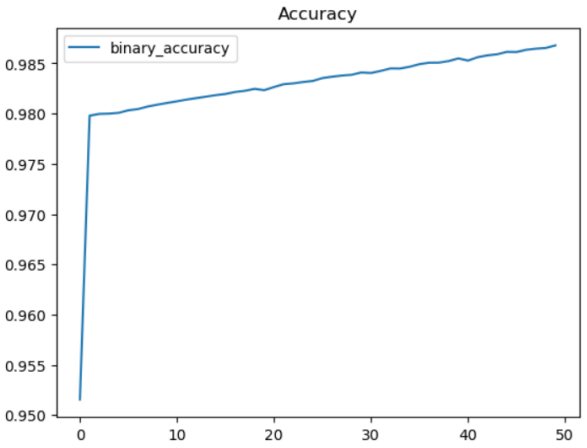
After experimenting with BLAST and standard ML models, we started Deep Learning by implementing a Multi Layer Perceptron with **T5** embeddings.



We used the **Adam** optimizer along with **binary_crossentropy** loss as we are dealing with multilabel classification.

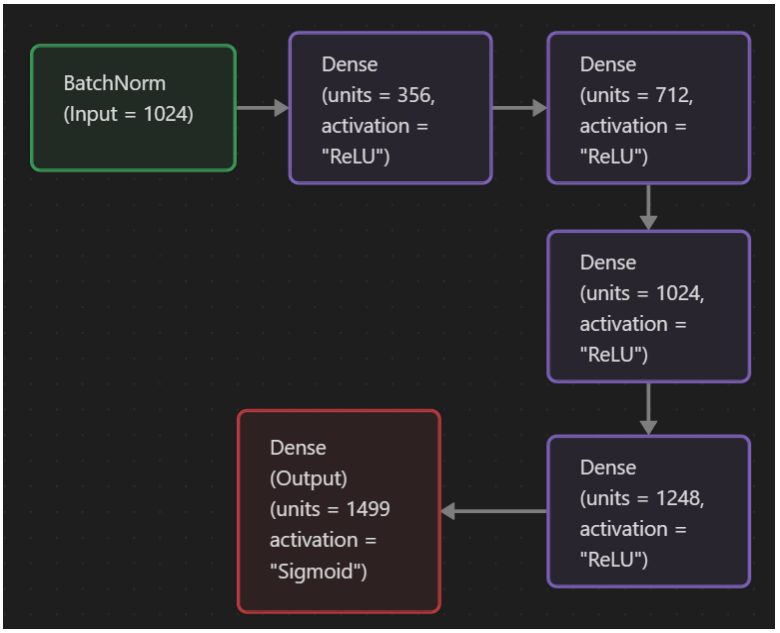
We also used the **AUC** score as well as the **F1-score** to evaluate the model's performance on a test set made from splitting train data into train and validation.



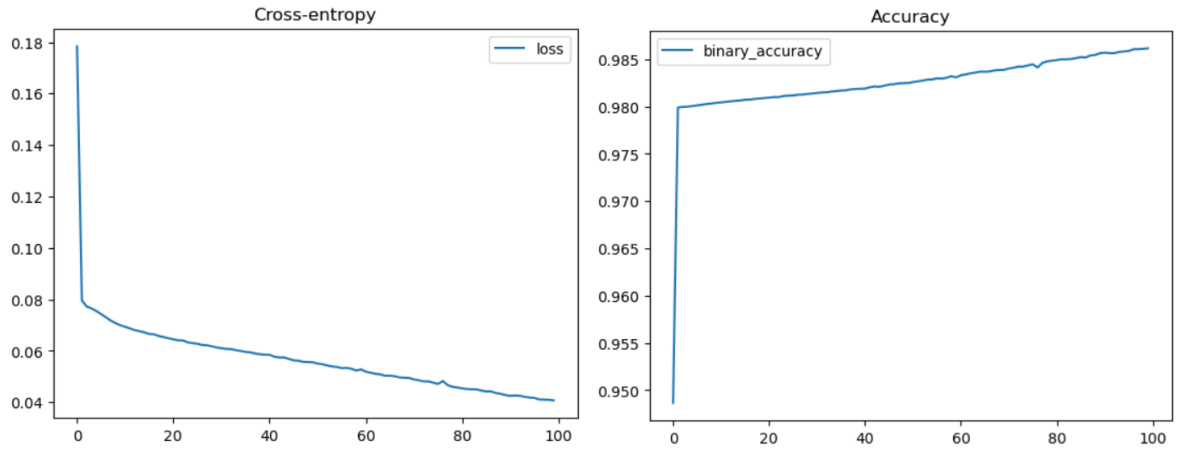


After doing hyperparameter tuning and submitting to the competition for 1 lakh labels we got a score of **0.40** and a hemming loss of **0.018**.

MLP with Protbert: [Link]

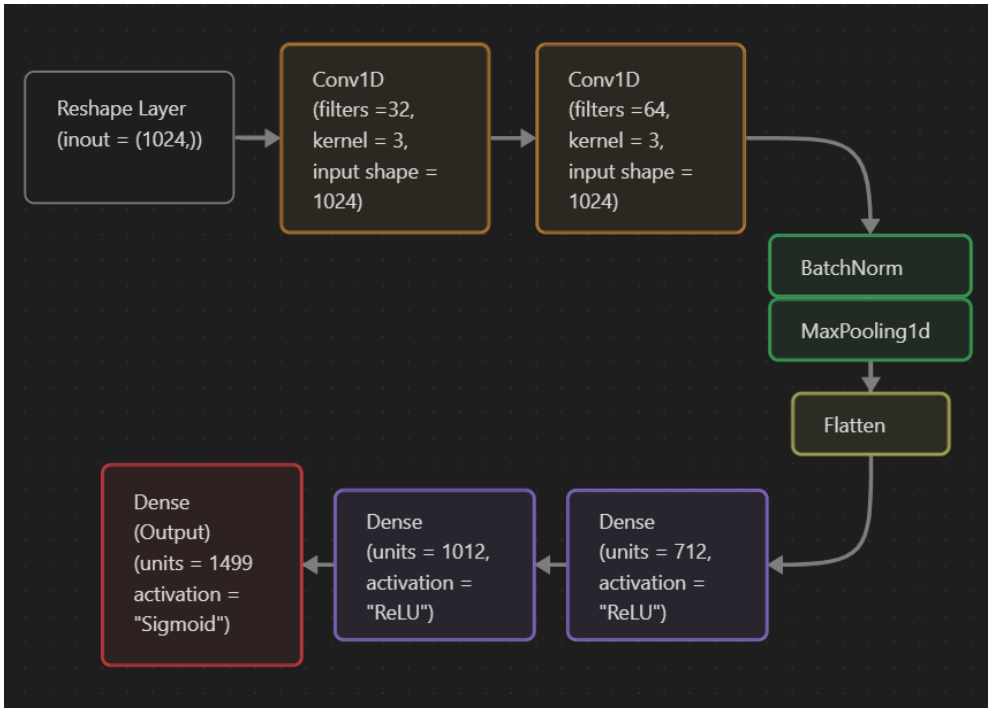


We used the Adam optimizer and Binary cross entropy loss with a learning rate of **0.001**.



We were able to get a F1 score of **0.38** with a hemming loss of **0.019** on the validation dataset.

MLP + Conv1D with Protbert: Link

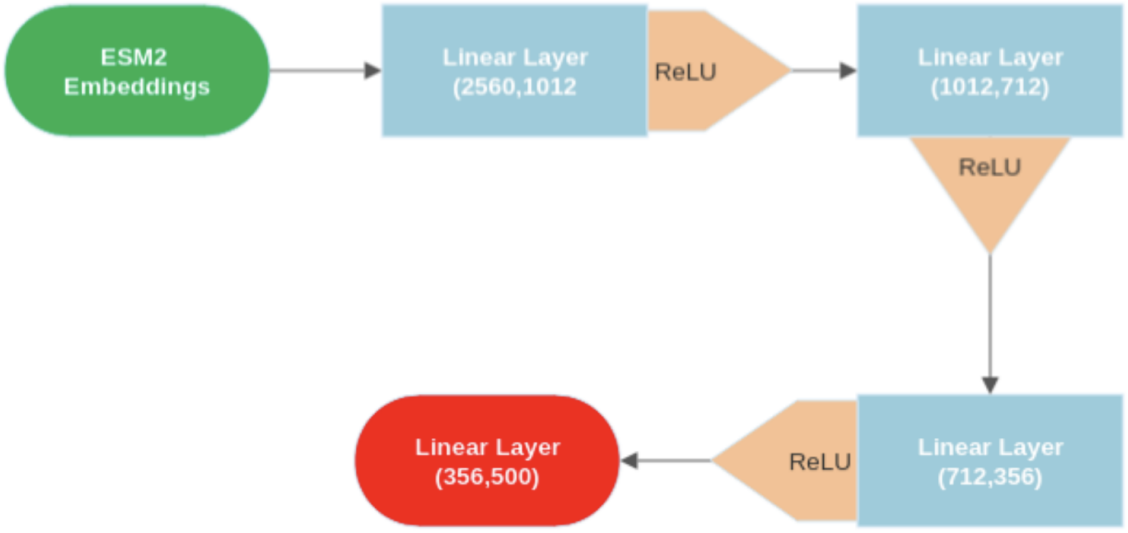


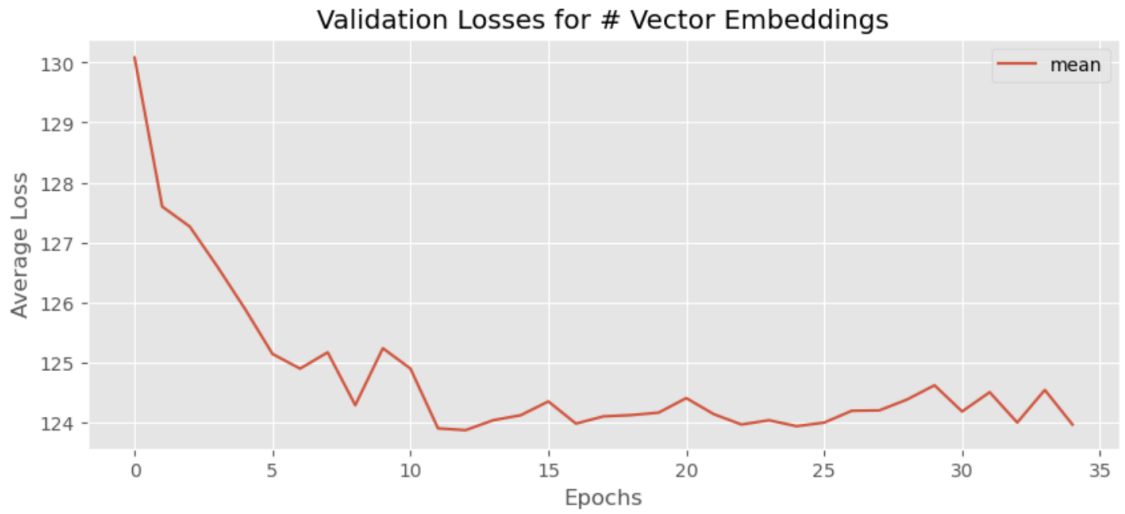
We used the Adam optimizer and Binary cross entropy loss with a learning rate of **0.0015**.

We were able to get a F1 score of **0.37** and a Hemming loss of **0.02** on the validation dataset.

MLP with ESM2: Link

Since, ProtBERT and T5 embeddings were not giving satisfactory F1 scores, we tried using the ESM2 embeddings and fine-tuned only the top **500** labels using a MLP model.





We used the Adam optimizer and Cross Entropy loss with a learning rate of **0.001** to get the best F1 score of **0.50** and an average loss of **123.96** on the validation dataset.

We also tried using a MLP + Conv1D model, but there was no improvement in results.

Conclusion

The table below summarizes the F1 score obtained with different models and embedding combinations:

Model	**F1-Score**
KNN	0.059
MLP + T5	0.40
MLP + ProtBert	0.38

MLP + Conv1D with ProtBert	0.37
MLP with ESM2	0.50

References:

- https://link.springer.com/article/10.1186/1471-2105-14-S3-S14
- https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-14-S3-S8