Project Report

Indian Institute of Technology Indore

Report for Multi-label protein function prediction

Course Instructor: Dr. Aruna Tiwari

Team Members:
Gourav Ahlawat (210001019)
Jilagam Dinesh Venkat Kumar (210001026)
Yash Vashistha (210001082)

Contents

1 Introduction

1.1 Background and Overview

Our project is centered around protein function prediction, specifically focusing on predicting the functional annotations (GO Term IDs) for protein sequences using machine learning techniques. Each protein sequence may exhibit multiple functions in this context, making the task a multi-label classification problem.

GO is a directed acyclic graph. The nodes in this graph are functional descriptors (terms or classes) connected by relational ties between them (is-a, part-of, etc.). For example, the terms 'protein binding activity' and 'binding activity' are related by an is-a relationship; however, the edge in the graph is often reversed to point from binding towards protein binding. This graph contains three subgraphs (subontologies): Molecular Function (MF), Biological Process (BP), and Cellular Component (CC), defined by their root nodes. Biologically, each subgraph represents a different aspect of the protein's function: what it does on a molecular level (MF), which biological processes it participates in (BP), and where in the cell it is located (CC).

This multi-label classification task is crucial in understanding the diverse roles and functionalities of proteins, aiding in biological research, drug discovery, and bioinformatics analyses. By accurately predicting the GO Term IDs for protein sequences, we can gain insights into the underlying biological processes and mechanisms, facilitating advancements in various domains such as personalized medicine and protein engineering.

‘

1.2 Context

Proteins are responsible for many activities in our tissues, organs, and bodies and play a central role in the structure and function of cells. Proteins are large molecules composed of 20 building blocks known as amino acids. The human body makes thousands of proteins, each composed of dozens or hundreds of amino acids linked sequentially. This amino-acid sequence determines the protein's 3D structure and conformational dynamics, which determines its biological function. Due to ongoing genome sequencing projects, we are inundated with large amounts of genomic sequence data from thousands of species, which informs us of the amino-acid sequence data of proteins for which these genes code. The accurate assignment of biological function to the protein is key to understanding life at the molecular level. However, assigning functions to any specific protein can be difficult due to the multiple functions many proteins have and their ability to interact with multiple partners. More knowledge of the functions assigned to proteins—potentially aided by data science—could lead to curing diseases and improving human and animal health and wellness in areas as varied as medicine and agriculture.

2 Problem

2.1 Problem Description

Our goal is to predict a protein's various go-terms or functions given a fasta file. The FASTA file format is a standard text-based format for representing nucleotide or protein sequences. It's widely used in bioinformatics for storing and sharing biological sequence data.

A FASTA file typically consists of one or more sequence entries, each of which includes two main parts: Header: The header line begins with a greater-than symbol ("¿") followed by a unique identifier for the sequence, optionally followed by a description or additional information about the sequence. The identifier can be any string of characters, but it's often chosen to be descriptive to provide information about the source or nature of the sequence.

Sequence Data: The sequence data follows the header line and consists of the actual nucleotide or protein sequence. The sequence can span multiple lines if necessary, but it's typically written as a single uninterrupted line of characters representing the sequence's residues. Nucleotide sequences are represented by the characters A (adenine), T (thymine), G (guanine), and C (cytosine) for DNA sequences, or A, U (uracil), G, and C for RNA sequences. Protein sequences are represented by single-letter amino acid codes such as A (alanine), R (arginine), G (glycine), etc.

For our problem, we needed to use the variable length sequence data and predict its various go-terms.

2.2 Dataset

We decided to use a dataset from Kaggle since it had a large dataset covering various protein sequences.

Gene Ontology: The ontology data is in the file `go-basic.obo`. This structure corresponds to the 2023-01-01 release of the GO graph. The nodes in this graph are indexed by the term name. For example, the roots of the three ontologies are:

```
subontology_roots = {'BPO': 'GO:0008150',
                     'CCO': 'GO:0005575',
                     'MFO': 'GO:0003674'}
```

Training Sequences: The `train-sequences.fasta` file contains the protein sequences for the training dataset. The headers in the `train-sequences.fasta` file indicate from which database the sequence originates. For example, `sp|P9WHI7|RECN-MYCT` in the FASTA header indicates the protein with UniProt ID P9WHI7 and gene name RECN-MYCT was taken from Swiss-Prot (`sp`). Any sequences taken from TrEMBL will have `tr` in the header instead of `sp`.

Labels: The `train-terms.tsv` file contains the list of annotated terms (ground truth) for the proteins in `train-sequences.fasta`. The first column indicates the protein's UniProt accession ID, the second is the GO term ID, and the third indicates in which ontology the term appears.

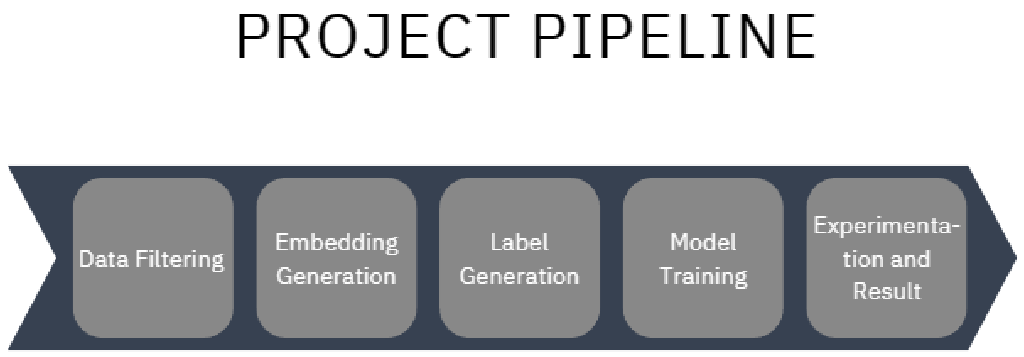
Taxonomy: The `train-taxonomy.tsv` file contains the list of proteins and the species to which they belong, represented by a "taxonomic identifier" (taxon ID) number. The first column is the protein UniProt accession ID, and the second is the taxon ID.

The dataset contains the following files:

- `go-basic.obo`: ontology graph structure.
- `train-sequences.fasta`: amino acid sequences for proteins in the training set.
- `train-taxonomy.tsv`: taxon ID for proteins in the training set.
- `train-terms.tsv`: the training set of proteins and corresponding annotated GO terms.

Our training set had 5,363,863 rows, which meant over 5 million proteins to train our model on, and `train-terms.tsv` mentioned the GO-term and aspect of every one of them.

3 Problem Overview



4 Methodology

4.1 Data-Preprocessing

- We needed to convert the protein sequences into a format we could run machine learning models so we tried to find ways to convert it into an embedding. We processed the fasta file and used the sequence string to generate tokens or tokenize the string.

- To generate vector embeddings. The best way to generate embeddings currently is to use transformer based models to generate word embeddings from the tokens we generated from the sequence strings.

- We also needed to limit our target classes since it would not be computationally feasible for us to train separate models for each class either we decided to use the top k classes and tried to vary k to get a mix of a good model which would also be computationally viable for us to train. Then we modeled the problem as a multiclass binary labeled problem where our inputs would be embeddings and outputs would be a 0 or 1 label for each of the k go-terms we are trying to predict.

4.2 Model

- To generate the vector embeddings, we used transformer based models to generate vector embeddings from protein sequences. We used different models we found were currently best in class. Word embedding models work by trying to model the words into a *N-D* dimension space, and the relationship of the words is modeled by their distances in various dimensions. Transformer models we used were trained on very large datasets generated by web crawling, and transformers allowed them to have a larger context to model the relationships better.

- After we got the embeddings, we created a multilayer Deep Neural Network (DNN) architecture to predict the various go-terms. We used DNNs since they are flexible and powerful in finding abstract relationships between input and output data. They are theoretically proven capable of mapping any relationship between input data and output classes.

5 Experimentation and Results

5.1 Experimentation

- We tried various embedding models since they were a key part of our process, and better embeddings would lead to better results. We tried transformer-based word embedding models such as ProtBERT,T5, and EMS2. We found T5 to be the best embedding model for us it gave us a vector embedding of size 1024 for every sequence.
- We tried different k sizes where k is the number of top most frequent go-terms we are trying to predict. We ultimately decided to use 1500 as the k size as a decent tradeoff between the number of terms we are trying to predict and computational complexity.
- We tried different numbers of layers and tried to balance between overfitting and underfitting by using a higher number of epochs and more complex models by adding more layers and dropout layers in between. Ultimately we used a DNN with one batch normalization layer after the input layer, then 5 hidden layers with 512 neurons, each having a dropout layer between each one and then an output layer.

5.2 Performance Metrices

Performance metrics are essential in machine learning for evaluating models, guiding decisions, and driving business outcomes. They enable model comparison, selection, and optimization by quantifying performance and highlighting strengths and weaknesses. These metrics impact business decisions like fraud detection or customer churn prediction. By monitoring metrics over time, practitioners can identify performance trends and trigger necessary updates, ensuring continuous improvement in model effectiveness. Performance metrics are crucial in assessing, optimizing, and understanding model performance, ultimately driving better decision-making in machine learning applications.

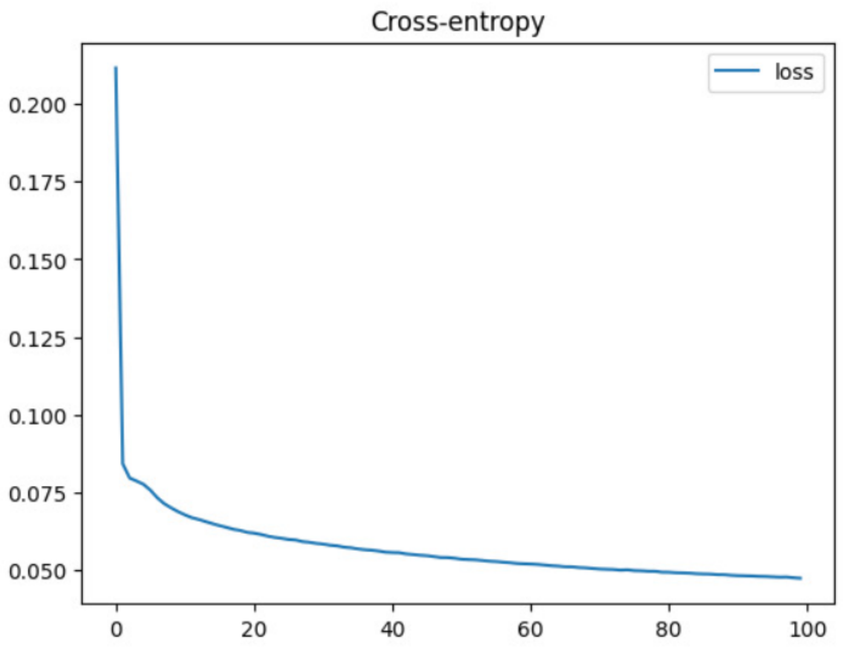
For our problem, we used the following performance metrics:

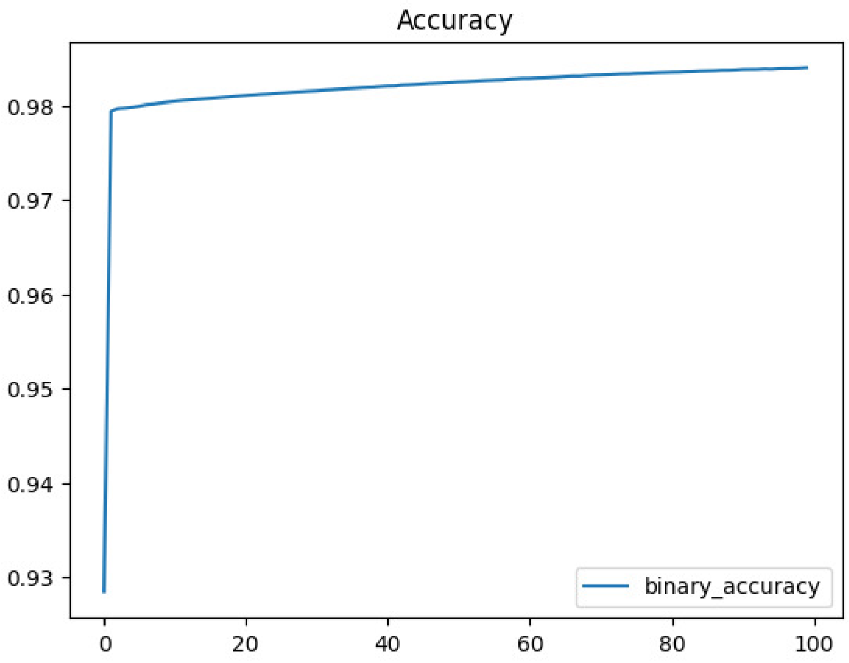
- **Binary Accuracy**: Binary accuracy is a simple and commonly used metric for evaluating classification models. It measures the proportion of correct predictions made by the model among all predictions made. In binary classification problems, where there are only two possible outcomes (e.g., true or false, positive or negative), binary accuracy calculates the ratio of correct predictions to the total number of predictions.
- **Binary Crossentropy**: Binary crossentropy, also known as log loss, is a loss function used in binary classification tasks to measure the difference between predicted probabilities and true binary labels. It quantifies the difference between the predicted probability distribution and the actual distribution of the labels. Lower values of binary crossentropy indicate better model performance.
- **Area Under the ROC Curve (AUC)**: AUC is a metric used to evaluate the performance of binary classification models. It represents the area under the receiver operating characteristic (ROC) curve, which is a graphical plot that illustrates the

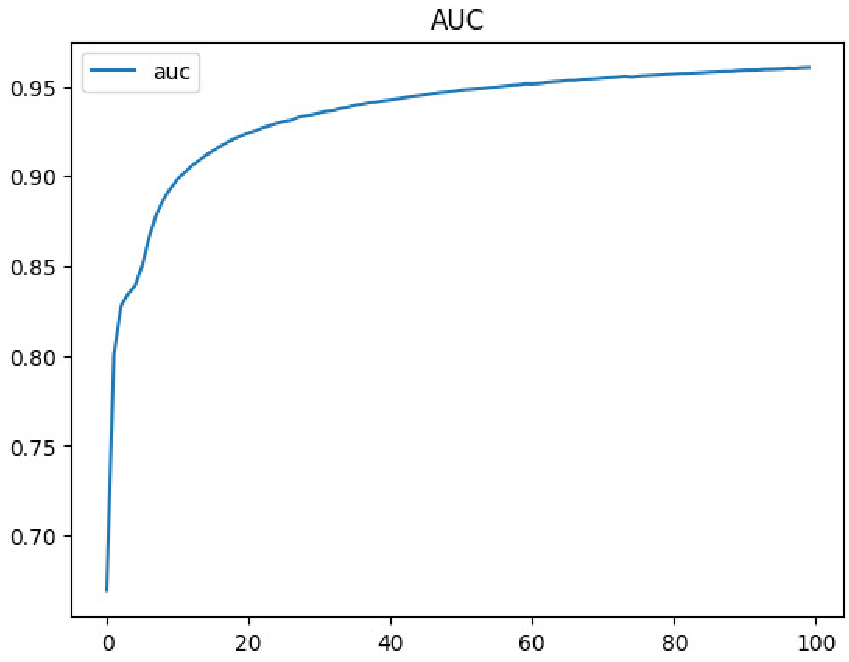
true positive rate (sensitivity) against the false positive rate (1-specificity) at various threshold settings. AUC provides a single scalar value representing the model's overall performance, with higher values indicating better discrimination between positive and negative classes.

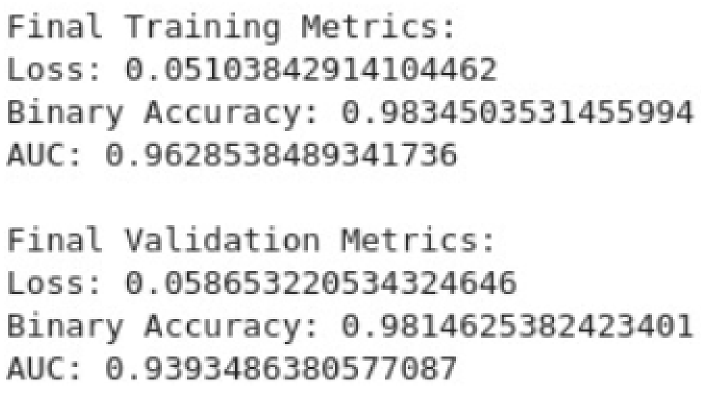
- **Precision and Recall**: Precision and recall are two important metrics used in binary classification tasks, especially when the class distribution is imbalanced. Precision measures the accuracy of positive predictions, i.e., the ratio of true positive predictions to the total number of positive predictions made by the model. Recall, also known as sensitivity, measures the ability of the model to correctly identify all positive instances, i.e., the ratio of true positive predictions to the total number of actual positive instances in the data. Precision emphasizes the accuracy of positive predictions, while recall emphasizes the completeness of positive predictions. These two metrics are often traded against each other, and the F1 score, the harmonic mean of precision and recall, is commonly used to balance them.

5.3 Results

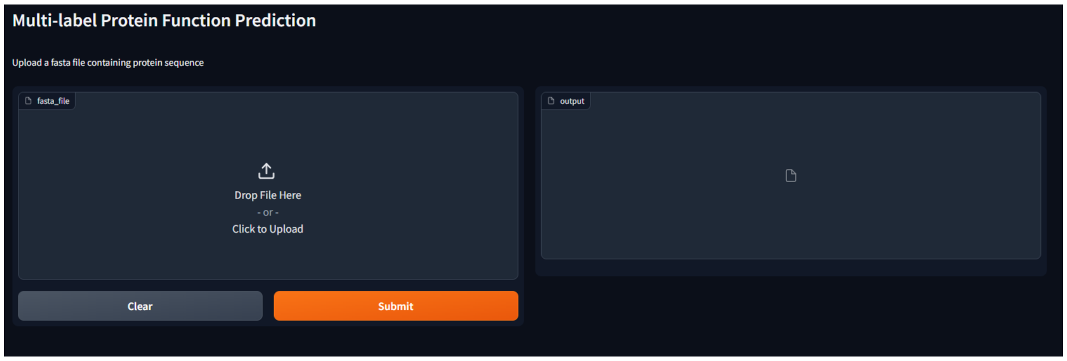



Accuracy
0.98
0.97
0.96
0.95
0.94
0.93
binary_accuracy
0
20
40
60
80
100


AUC
auc
0.95
0.90
0.85
0.80
0.75
0.70
0
20
40
60
80
100



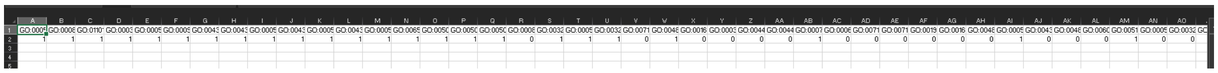
```
Final Training Metrics:
Loss: 0.05103842914104462
Binary Accuracy: 0.9834503531455994
AUC: 0.9628538489341736

Final Validation Metrics:
Loss: 0.058653220534324646
Binary Accuracy: 0.9814625382423401
AUC: 0.9393486380577087
```



Multi-label Protein Function Prediction

Upload a fasta file containing protein sequence

Interface deployed on hugging face



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	GO:0005	GO:0008	GO:0110	GO:0003	GO:0005	GO:0009	GO:0043	GO:0043	GO:0005	GO:0043	GO:0005	GO:0043	GO:0005	GO:0065	GO:0050	GO:0050	GO:0050	GO:0008	GO:0032	GO:0005	GO:0032
2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
3																					
4																					
5																					

	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN	AO	
1	GO:0071	GO:0048	GO:0016	GO:0003	GO:0044	GO:0044	GO:0007	GO:0006	GO:0071	GO:0071	GO:0019	GO:0016	GO:0048	GO:0005	GO:0043	GO:0048	GO:0060	GO:0051	GO:0005	GO:0032	GC
2	0	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	
3																					
4																					
5																					

Predictions output

5.4 Code Implementation

- Our Implementation of Multi-label protein function prediction.
- Interface to easily use it: Huggingface.
- GitHub repository link.

6 Acknowledgment

We thank Dr. Aruna Tiwari for teaching the course: COMPUTATIONAL INTELLIGENCE with utmost finesse. As a computer science student, this course is crucial for any career path we wish to pursue and to have learned it with this level of clarity is our pleasure. Moreover, this project and many other lab assignments have enhanced our theory application and given us hands-on coding experience with real-life task-based questions. So, once again we show our gratitude towards our professors and TAs, who ensured our learning went smoothly.

7 References

- Dataset
- T5: a detailed explanation.
- Text-To-Text Transfer Transformer(T5).
- T5 embedding dataset.
- The Gene Ontology and the Meaning of Biological Function.