Protein Function Prediction Using Deep Learning

Professor:Aruna Tiwari

K.V.S Teja 220001038
Kalyan Sriram 220001023

1. Introduction

1.1 Motivation and Objective

This project addresses the challenge of predicting protein functions by identifying their corresponding Gene Ontology (GO) term annotations using deep learning. A protein can be associated with multiple biological functions, making this a multi-label classification problem.

Gene Ontology (GO) provides a structured vocabulary in the form of a Directed Acyclic Graph (DAG). Each node in this graph represents a functional descriptor (GO term), and the edges represent biological relationships such as is-a or part-of. For instance, "protein binding activity" is a more specific subclass of "binding activity". These relationships are directionally structured but biologically hierarchical.

GO is divided into three distinct subontologies:

- **Molecular Function (MF)**: Describes the basic biochemical activity of a protein.
- **Biological Process (BP)**: Captures the biological objectives the protein contributes to.
- **Cellular Component (CC)**: Indicates the location within the cell where the protein is active.

The goal of this project is to develop a machine learning pipeline capable of accurately predicting the relevant GO terms for a given protein sequence. Such predictions are valuable for understanding protein function, supporting biological research, drug discovery, disease diagnostics, and applications in bioengineering and personalized medicine.

1.2 Biological Background

Proteins are essential biomolecules responsible for most structural and functional processes in living organisms. They are composed of sequences of 20 standard amino acids, which fold into complex 3D structures that determine their functional roles.

Advancements in genome sequencing have led to a massive influx of amino acid sequence data from various organisms. While we can easily determine a protein's sequence from genetic data, understanding its biological function remains a significant challenge—especially because many proteins perform multiple functions or interact with multiple biological partners.

Assigning functions to unknown proteins is critical for decoding life at the molecular level. Machine learning, particularly deep learning, offers a promising solution to automate and improve the accuracy of protein function prediction. By leveraging data-driven models, we aim to bridge the gap between protein sequence and biological insight, ultimately aiding efforts in healthcare, agriculture, and molecular biology.

2. Problem Definition and Dataset Description

2.1 Problem Statement

The objective of this project is to develop a model that can predict the functional annotations of proteins—specifically, their associated Gene Ontology (GO) terms—based solely on their amino acid sequences provided in FASTA format. This task is inherently a multi-label classification problem, as a single protein may be associated with multiple GO terms.

The FASTA file format is a widely adopted standard in bioinformatics for storing biological sequences. Each entry in a FASTA file includes:

- A header line beginning with > that contains a unique identifier and optional description.
- One or more lines containing the amino acid sequence, represented by standard single-letter amino acid codes (e.g., A for alanine, R for arginine, G for glycine, etc.).

Our model is designed to process these variable-length protein sequences and predict one or more relevant GO terms that describe the protein's molecular function, biological process, or cellular component.

2.2 Dataset Overview

We utilized a well-curated dataset released for the CAFA-5 (Critical Assessment of Function Annotation) challenge, hosted on Kaggle. The dataset includes millions of protein sequences along with their taxonomic data and annotated GO terms. Below are the key components of the dataset:

- **Ontology Structure – go-basic.obo**: GO ontology graph (DAG) released on 2023-01-01.
- **Protein Sequences – train_sequences.fasta**: Contains amino acid sequences for training.

- **GO Term Annotations – train_terms.tsv**: Associates protein IDs with GO terms.
- **Taxonomic Metadata – train_taxonomy.tsv**: Maps proteins to species using taxon IDs.

2.3 Dataset Summary

File Name	Description
go-basic.obo	GO ontology graph (DAG)
train_sequences.fasta	Protein sequences (training set)
train_terms.tsv	GO term labels for proteins
train_taxonomy.tsv	Species info via taxon ID

Table 1: Dataset Files and Descriptions

Note: Although the raw dataset contains millions of entries, for this project we used a cleaned and curated subset consisting of ∼70,000 proteins and the top 1,500 most frequent GO terms to reduce computational complexity and ensure efficient model training.

3. Project Workflow

This section outlines the major components of the project pipeline for protein function prediction, from raw sequence input to GO term prediction using deep learning techniques.

3.1 Data Preprocessing

The raw protein sequence data is provided in FASTA format, containing amino acid sequences along with their headers. During preprocessing:

Non-standard amino acids (e.g., U, Z, B, O) are replaced with "X" to ensure compatibility with the tokenizer.

Sequences are cleaned and formatted for input into the embedding model.

3.2 Vector Embedding Generation

Each protein sequence is passed through a pretrained T5 encoder model (prot_t5_xl_half_uniref50-enc) to extract semantic embeddings.

Tokenization is performed using the model's custom tokenizer, converting amino acids into token IDs.

The T5 encoder produces contextual embeddings (shape: sequence length $\times$ 1024).

A final 1024-dimensional fixed-size vector is generated by averaging the embeddings across the sequence (mean pooling).

3.3 Label Encoding

GO term labels are extracted from the train_terms.tsv file. Since there are over 40,000 terms in total, only the top 1,500 most frequently occurring GO terms are selected for classification.

Each protein is mapped to its set of GO terms.

A multi-hot encoded matrix is constructed (proteins $\times$ 1500 labels), where:

1 indicates presence of a term,

0 indicates absence.

3.4 Neural Network Training

A custom Deep Neural Network (DNN) is constructed using TensorFlow Keras. The model architecture includes:

5 hidden layers, each with 512 neurons and ReLU activation.

Dropout layers for regularization (dropout rate = 0.2).

Final output layer with 1500 neurons and sigmoid activation for multi-label output.

The model is trained using:

Loss function: Binary cross-entropy

Optimizer: Adam

Metrics: Binary accuracy and AUC

Training/Validation split: 80% training, 20% validation

3.5 Evaluation and Prediction

The model's performance is evaluated on validation data using accuracy and AUC scores.

Predictions are made on unseen protein sequences by passing them through the same embedding + classification pipeline.

Output is a 1500-dimensional vector of probabilities, which are thresholded to obtain final GO term predictions.

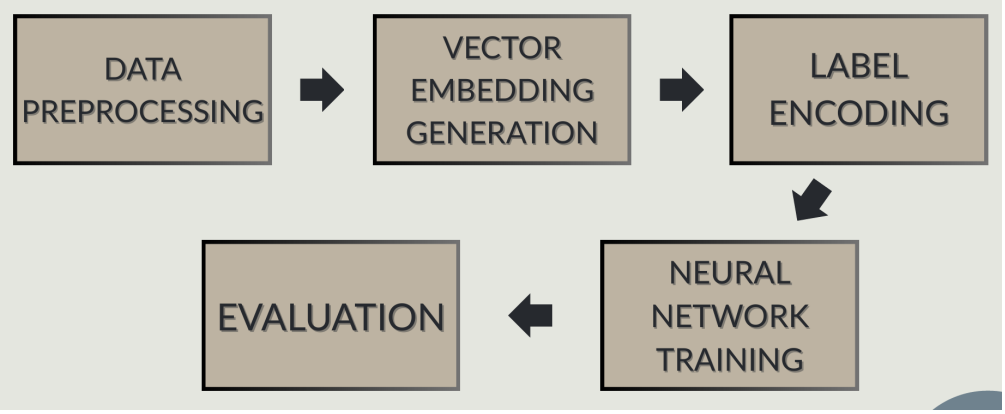

Figure 1: Figure 1: Complete pipeline screenshot illustrating the step-by-step architecture. (Insert your screenshot here, and label it clearly.)

4. Methodology

This section outlines the preprocessing steps and embedding generation from protein sequences, followed by the architecture of the deep learning model used for functional prediction.

4.1 Sequence Encoding and Label Selection

To utilize protein sequences in a deep learning model, we first needed to convert them into a numerical format. The sequences from the FASTA file were parsed and cleaned by replacing non-standard amino acids with 'X'.

We then used a transformer-based encoder model, specifically ProtT5, to generate embeddings. Protein sequences were tokenized (each amino acid separated by a space) and passed into the model. The model produced per-residue embeddings, which were averaged across the sequence to generate a fixed-length 1024-dimensional vector for each protein.

Since the full GO label space is extremely large (over 40,000 terms), we limited the prediction scope to the top 1,500 most frequent GO terms to make the problem computationally feasible while retaining biological relevance.

This transformed the task into a multi-label classification problem, where each protein embedding is mapped to a binary vector of length 1,500, indicating which GO terms apply to that protein.

4.2 Model Design

After generating embeddings, we constructed a Deep Neural Network (DNN) to predict the relevant GO terms.

The DNN architecture consists of multiple fully connected layers with non-linear activation functions, designed to learn complex patterns between the input embedding and output labels.

A total of five hidden layers, each containing 512 neurons and using ReLU activation, were used. These layers help the model capture higher-level relationships within the data.

Dropout layers were inserted after each dense layer to improve generalization and reduce overfitting during training.

The output layer consists of 1,500 neurons, one for each GO term being predicted. Each neuron outputs a probability score indicating the likelihood that the corresponding GO term applies to the input protein.

5. Experimentation and Results

5.1 Experimental Setup

In this phase, we systematically explored and optimized various components of the model pipeline to enhance the accuracy and generalization of GO term predictions.

Embedding Model Selection: We experimented with several transformer-based models including ProtBERT, EMS2, and ProtT5 for generating protein sequence embeddings. Among these, ProtT5 yielded the best performance in terms of embedding quality and downstream classification accuracy. It produces a 1024-dimensional vector representation for each protein, capturing meaningful biological context.

GO Term Selection (Top-k Filtering): Given the vast label space of over 40,000 GO terms, we varied the value of k—the number of most frequent GO terms—to strike a balance between prediction depth and computational feasibility. After evaluation, we fixed k = 1500, which ensured good biological coverage while keeping training complexity manageable.

Model Architecture Tuning: Multiple deep learning architectures were evaluated to balance model complexity with performance. We tested various depths and dropout rates to minimize overfitting and underfitting. The final model architecture includes:

- An input batch normalization layer
- Five hidden layers, each with 512 neurons and ReLU activation
- Dropout layers (0.2) inserted between hidden layers
- A sigmoid-activated output layer for multi-label prediction over 1500 GO terms

This architecture provided a robust structure for capturing non-linear patterns between input embeddings and GO term labels.

5.2 Evaluation Metrics

To evaluate the performance of our multi-label classifier, we adopted several standard classification metrics that offer a comprehensive view of the model's effectiveness:

Binary Accuracy: Measures the overall correctness of predictions across all labels. It computes the ratio of correctly predicted GO terms (both present and absent) over the total predictions made.We have obtained it to be around 0.95 i.e, 95 percentage.

Binary Cross-Entropy (Log Loss): A loss function used to measure the divergence between predicted probabilities and actual binary labels. Lower values indicate better model calibration and confidence alignment.We have obtained it to be around 0.43 i.e, 4 percentage.

Area Under the ROC Curve (AUC): Evaluates the model's ability to distinguish between positive and negative classes at various thresholds. A higher AUC reflects better discrimination and classification reliability.We have obtained it to be around 0.97 i.e, 97 percentage.

These metrics collectively helped guide model selection, tuning, and validation.

5.3 Results

We evaluated our model on the validation dataset and observed its performance using various metrics, including accuracy, AUC, and F1 score. Below are visualizations summarizing the experimental results.

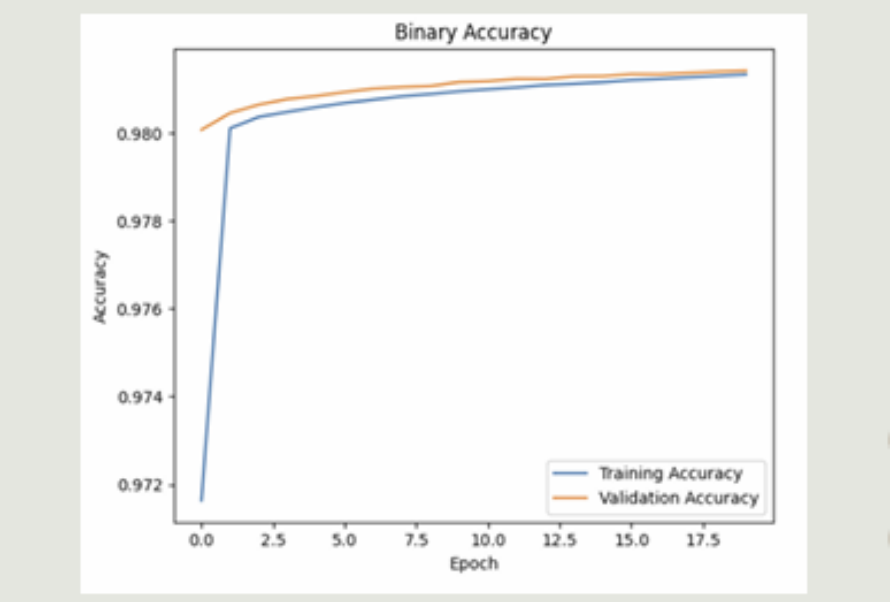

Figure 2: Training and validation accuracy over epochs.

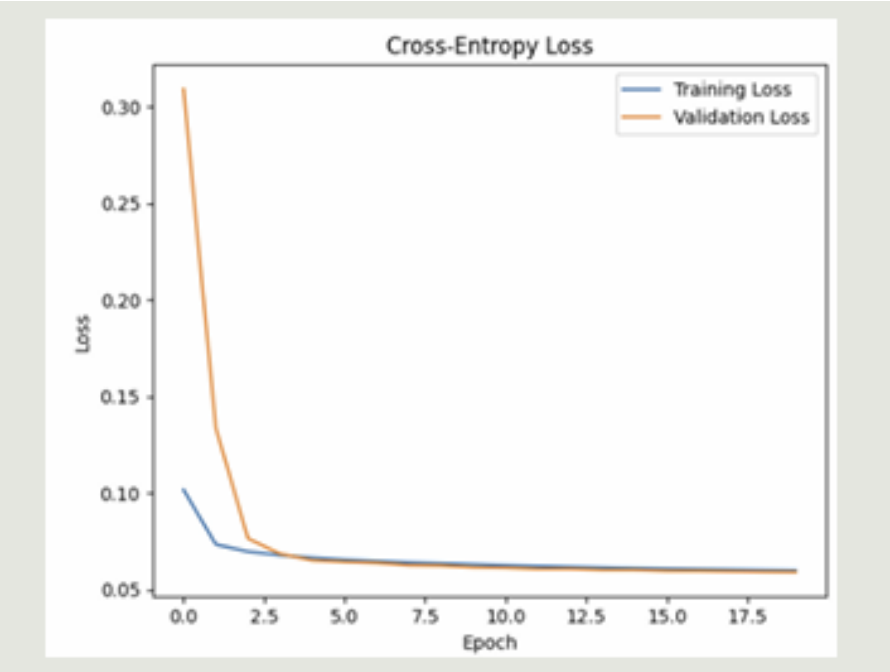

Figure 3: Binary cross-entropy loss during training.

5.4 Code Implementation

The entire pipeline for multi-label protein function prediction has been implemented using Python, integrating tools such as TensorFlow, PyTorch, BioPython, and Transformers from the Hugging Face library.

The model processes protein sequences from FASTA format, generates 1024-dimensional ProtT5 embeddings, and predicts associated GO terms using a custom-trained deep neural network.

The codebase is modular and well-documented, allowing easy extension or reuse for other sequence-based prediction tasks.

All source files, including preprocessing scripts, model training routines, embedding generation, and prediction utilities, are available in the project repository.

GitHub Repository: `[https://github.com/kalyanxyz]`

6. Acknowledgments

We would like to thank Dr. Aruna Tiwari for her excellent teaching of the Computational Intelligence course. As computer science students, the knowledge gained from this course has been invaluable to our academic growth and future careers.

This project, along with other lab assignments, has significantly enhanced our ability to apply theoretical concepts through hands-on coding experience. We also extend our gratitude to the professors and teaching assistants for their continued support throughout the learning process.

7. References

T5 transformers: link
Deep nueral Networks: link
Information about GO terms : link
Information about Protein Sequences and Amino acid breakdown(Download the pdf for more information): link