MULTI LABEL PROTEIN CLASSIFICATION

K.V.S.Teja 220001038

D.Kalyana Sriram 220001023

OVERVIEW

- Introduction
- Dataset collection
- Model
- Evaluation
- Results
- Acknowledgement
- Conclusion

INTRODUCTION

Protein Function Prediction Using Machine Learning (GO Term ID)

Our project focuses on predicting protein functions by identifying their corresponding Gene Ontology (GO) term IDs through machine learning techniques. Each protein sequence can have multiple functions and may therefore be associated with several GO terms. These terms are uniquely identified using GO Term IDs.

As a result, our model needs to predict all relevant GO terms for a given sequence, making this task a multi-label classification problem.

INTRODUCTION

Advantages of using this model:

- Automates protein annotation, saving time and reducing manual effort.
- Improves accuracy and consistency compared to rule-based or manual methods.
- Can generalize to unseen proteins based on learned patterns.
- Speeds up biological research and discovery processes.

Applications:

This model can be used in drug discovery, functional genomics, and personalized medicine, where understanding protein roles is critical for designing targeted therapies or identifying disease markers.

INTRODUCTION

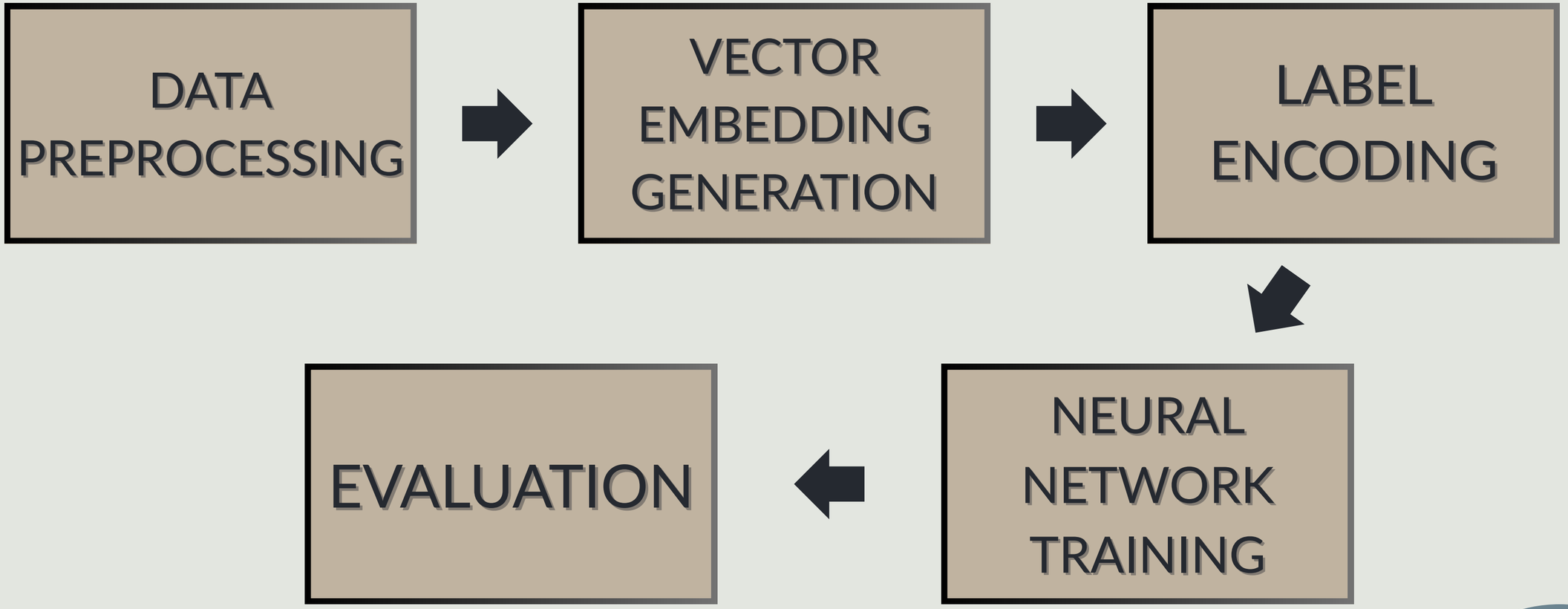
Gene Ontology (GO) is a key initiative in bioinformatics aimed at standardizing how gene and gene product characteristics are described across different species.

The project defines a gene product traits, into three main categories:

- **Cellular Component (CCO)**: Locations within the cell or in the extracellular space.
- **Molecular Function (MFO)**: Specific molecular-level roles performed by gene products, such as binding or enzymatic activity.
- **Biological Process (BPO)**: Sequences of molecular events with a start and end, essential for the functioning of cells, tissues, organs, and whole organisms.

Each GO term falls under one of these three categories.

WORKFLOW OVERVIEW



DATASET COLLECTION

- Our work is based on the **CAFA Protein Function Prediction** dataset, released as part of a Kaggle competition

DATASET

- To train a machine learning model, we cannot directly use raw protein sequences from FASTA files because they are in an alphabetical format.
- These sequences must be converted into a numerical format that the model can understand.

Our Approach

- We use embeddings of protein sequences to feed into the model.
- We will train a custom model on the protein sequences to generate these embeddings (vector representations).

T5 MODEL

1. Foundation Training & Adaptability

- T5 leverages large-scale pre-training followed by custom task-specific tuning.
- It captures general linguistic patterns and fine-tunes to specialized domains.
- This dual training makes it robust for diverse NLP applications.

2. Versatile Learning Objective

- The model learns to convert any input string into a desired output format.
- This makes it task-agnostic and ideal for various downstream applications.
- Its uniform training goal helps in easy generalization.

MODEL DESIGN

3. Text-to-Text Conversion Logic

Everything from translation to summarization is treated as text in, text out. This unified interface simplifies learning and boosts task performance.

- It enables consistent formatting and easier debugging across use cases.

4. Embedding Real-World Inputs

T5 can embed data types beyond text, such as biological sequences.

Using tokenizers and encoders, sequences become usable by deep learning models.

- This bridges the gap between raw data and model understanding.

FINAL DATASET

We extracted all GO Term IDs from the training data.
There are over 50,000 unique labels.
To manage complexity, we selected the top most frequent GO terms as labels.

Building the Label Dataset

We created a new dataframe with:

- 1,500 columns (GO Term IDs)
- 142,246 entries (protein samples)

Each entry is labeled with 1 (present) or 0 (absent) for each GO Term.

This is a multi-label classification problem, meaning one protein can be assigned multiple GO terms.

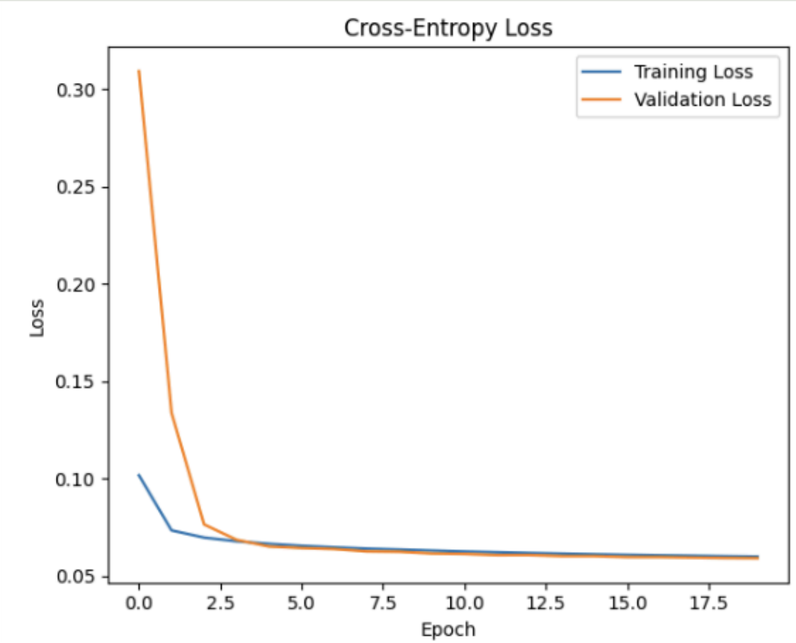
TRAINING SETUP

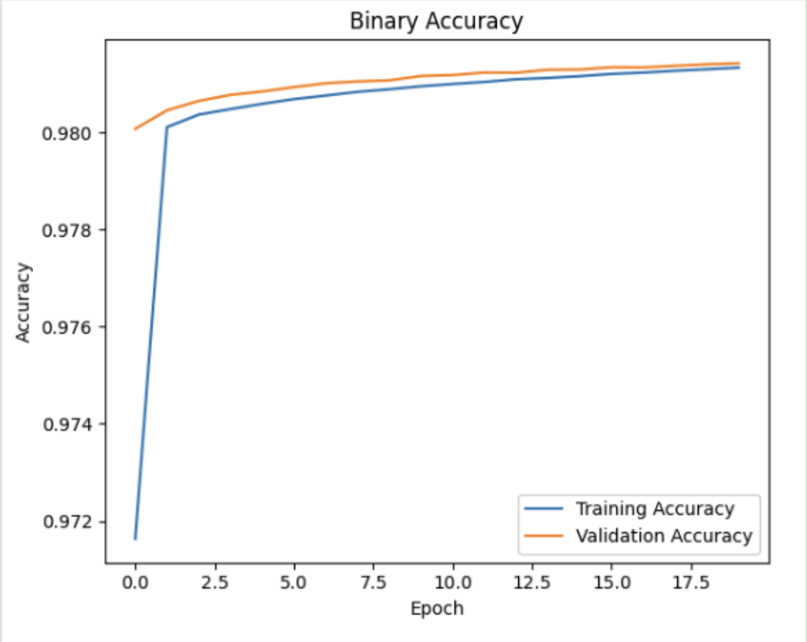
- BATCH SIZE: SPEEDS UP TRAINING USING PARALLEL COMPUTATION WHILE MAINTAINING STABLE GRADIENTS.
- BATCH NORMALIZATION: ADDED AT INPUT TO STABILIZE AND ACCELERATE CONVERGENCE.
- DROPOUT (0.3): PREVENTS OVERFITTING BY RANDOMLY DISABLING NEURONS DURING TRAINING.
- LOSS FUNCTION: BINARY_CROSSENTROPY – SUITABLE FOR INDEPENDENT BINARY LABELS.
- METRICS: TRACKING BINARY_ACCURACY AND AUC FOR PERFORMANCE INSIGHTS.
- OPTIMIZER: ADAM WITH LEARNING RATE 0.001 – ADAPTS LEARNING RATE DURING TRAINING.

RESULT

The Accuracy is around 95%

AUC is 98% , Loss will be 5%





RESULT

- Input files provided

```
# just run it so you will get both decimal file and modified binary fil
predict('/kaggle/input/fastaexample/example.fasta')
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/normalization/batch_normal
```

example.fasta ×

example.fasta (342 B) /kaggle/input/fastaexample/example.fasta

```
>P20536 sp|P20536|UNG_VACCC Uracil-DNA glycosylase OS=Vaccinia virus (strain C
MNSVTVSHAPYTITYHDDWEPVMSQLVEFYNEVASWLLRDETSPIPDKFFIQLKQPLRNK
RVCVCGIDPYPKDGTGVPFESPNFTKKSIKEIASSISRLTGVIDYKGYNLNIIDGVIPWN
```

- Output csv files of GOs

predictions.csv

decimal.csv

ACKNOWLEDGEMENT

We are grateful for valuable guidance of
Dr. Aruna Tiwari and T.A’s for assisting us
in this Computer Intelligence Project

Thank You

For your attention