INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, SRI CITY

ANDHRA PRADESH

Deep Learning Project Report

Protein-Based Disease Classification using CNN-LSTM

Team : 009

Hrishikesh Dongre	S20220010083
Sahil Karsare	S20220010191
Vinayak Ananad	S20220010243
Suvan Sarkar	S20220010220

TO

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
IIIT SRI CITY, ANDHRA PRADESH

Submitted to **: Dr. Chandra Mohan D**

Date of Submission **: May 06, 2025**

0.1 Problem Statement and Motivation

The classification of DNA sequences into biologically meaningful categories—such as AIDS-related genes, proto-oncogenes, and tumor suppressor genes—is a critical task in computational biology. These genes are central to understanding and combating diseases like cancer and HIV. However, DNA sequences are inherently complex and high-dimensional, containing both local patterns (e.g., short motifs) and long-range dependencies (e.g., regulatory signals) that traditional machine learning models struggle to capture effectively.

To address this, our project aims to develop a robust classification framework that combines the strengths of Convolutional Neural Networks (CNNs) and Long Short-Term Memory networks (LSTMs). CNNs are effective at identifying local features within sequences, such as specific nucleotide motifs, while LSTMs are well-suited for capturing sequential dependencies and contextual information across longer spans of the DNA sequence. By integrating CNN and LSTM layers, we aim to construct a hybrid model capable of learning both spatial and temporal patterns in DNA sequences, improving classification performance over traditional approaches.

This project leverages k-mer based feature extraction, hyperparameter tuning, and visual analysis (e.g., violin plots, heatmaps) to gain deeper insight into the dataset and model behavior. Ultimately, the goal is to build a scalable and interpretable system that contributes to developing a CNN-LSTM based early disease detection.

0.2 Data Collection and Preprocessing

Data Source

- Protein sequences were collected from the UniProt database corresponding to three disease categories: AIDS, Proto-oncogene, and Tumor Suppressor.
- The sequences were downloaded in FASTA format and stored in respective directories.

Dataset Summary

- Total Sequences Loaded: 3879
- Class Distribution:
  - Tumor Suppressor: 1719 sequences
  - AIDS: 1282 sequences
  - Proto-oncogene: 878 sequences
- Sample sequence lengths ranged widely. For example:
  - Sequence 1: Length 344
  - Sequence 2: Length 113
  - Sequence 3: Length 312
  - Sequence 4 and 5: Length 522

Preprocessing Steps

- **Sequence Cleaning:** Parsed the FASTA files using Biopython and extracted raw amino acid sequences.
- **Sequence Length Handling:**

  - To standardize input for the model, all sequences were either padded or truncated to a fixed length.
  - Shorter sequences were padded with a placeholder token ('X') and longer sequences were truncated.

- **Train-Test Split:**
  - The dataset was split into training and testing sets with a ratio of 80:20.

0.3 Data Visualization

To gain a comprehensive understanding of the protein sequence dataset, we performed multiple visualizations using Seaborn and Matplotlib. These visualizations provided insights into sequence properties and class characteristics that were crucial for preprocessing and model design.

1. Class Distribution

- A count plot was created to visualize the number of sequences in each class: AIDS, Proto-oncogene, and Tumor Suppressor.
- *Purpose:* To assess class balance and determine whether class imbalance techniques were necessary.

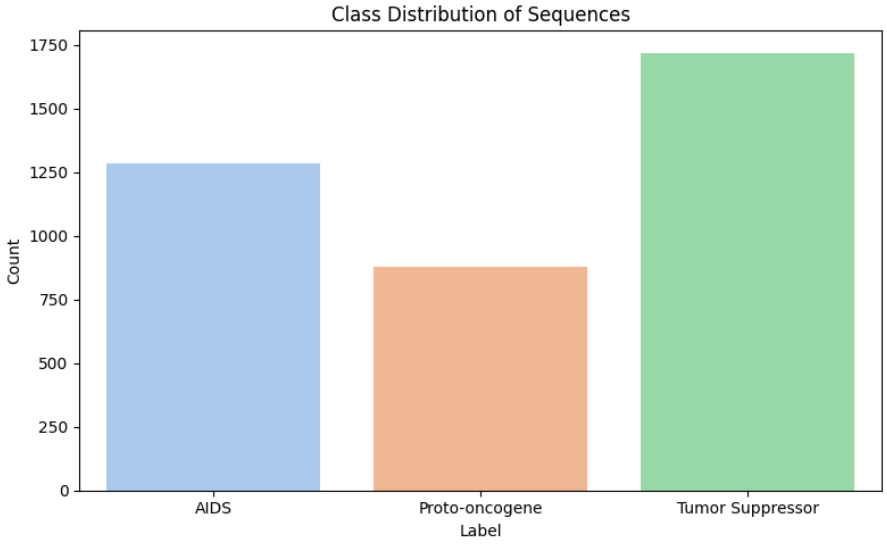

Figure 1: Class distribution across all classes.

2. Sequence Length Distribution

- Violin plots were used to show the distribution of sequence lengths per class.
- *Purpose:* To identify variability in sequence lengths between categories and to guide sequence padding/truncation thresholds.

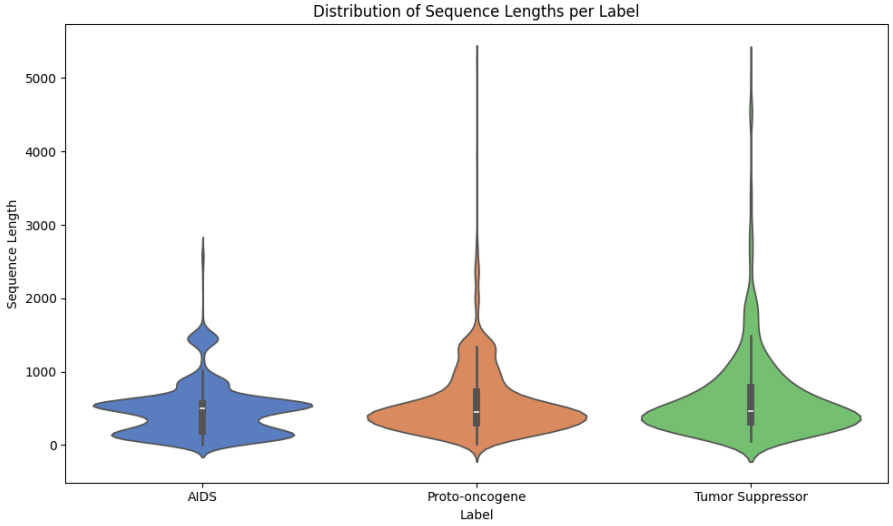

Figure 2: Distribution of sequence lengths across all classes.

3. GC Content Distribution

- GC content (percentage of Guanine and Cytosine) was computed for each sequence using the `gc_fraction` utility from Biopython.
- A violin plot was generated to visualize GC content across different labels.
- *Purpose:* To evaluate whether GC content could serve as a distinguishing feature among classes and to understand the biological diversity within the dataset.

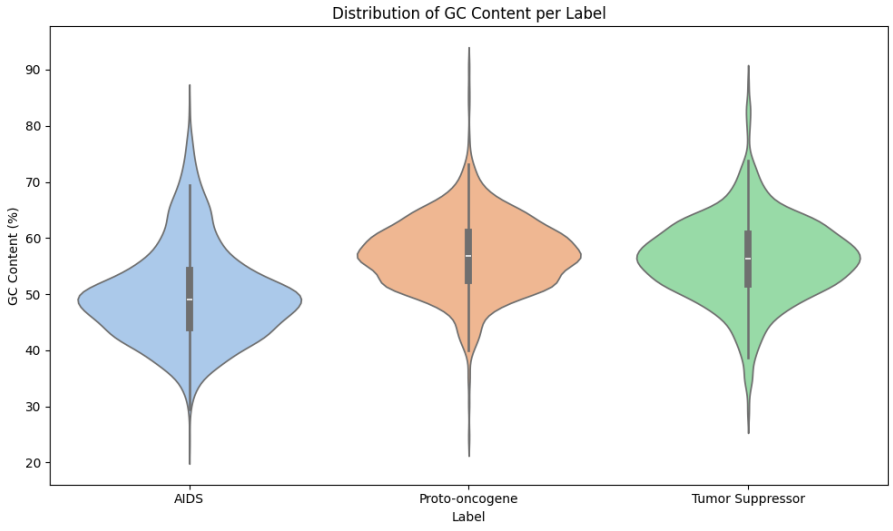

Figure 3: GC content distribution across sequence classes.

4. k-mer Frequency Heatmap

- We calculated the frequencies of all possible 2-mers (di-nucleotides) within each sequence using a sliding window approach.
- A heatmap was generated to visualize the average 2-mer frequency distribution for each class (AIDS, Proto-oncogene, Tumor Suppressor).
- *Purpose:* To uncover class-specific nucleotide patterns and motifs that could serve as discriminative features for model training.

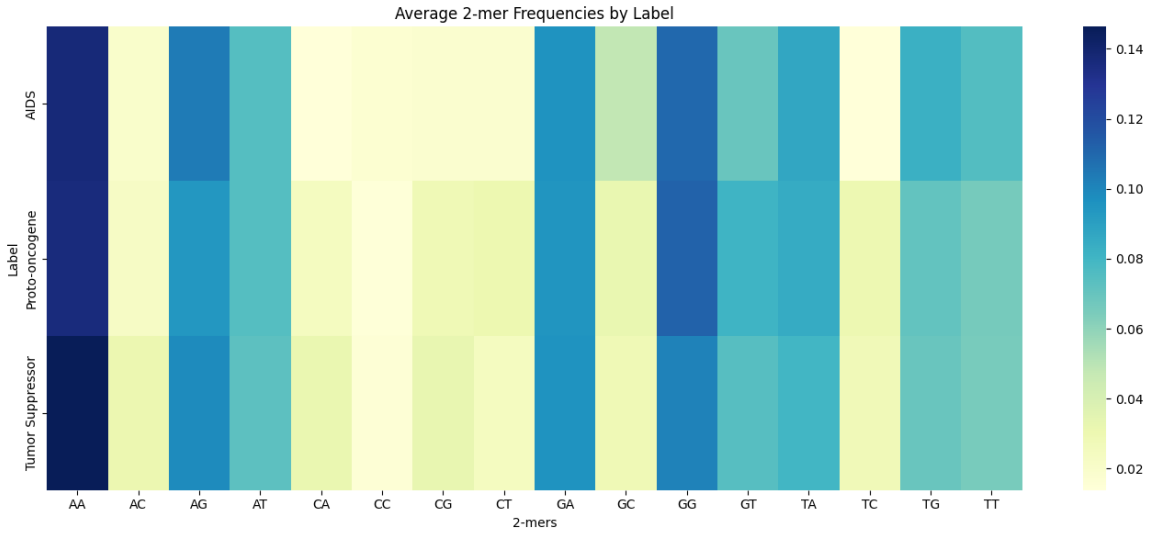

Figure 4: Average 2-mer frequency heatmap by class.

These plots enabled us to summarize the structure and diversity of the dataset, inform decisions related to preprocessing (such as fixed sequence length), and lay the groundwork for effective model design.

0.4 Feature Extraction Using iFeature

To transform raw protein sequences into machine-readable numerical vectors, we employed the **iFeature** toolkit. iFeature is a comprehensive Python-based package and web server that allows for the extraction of a diverse set of sequence-based descriptors including compositional, physicochemical, and structural features. It is widely adopted in bioinformatics research and has demonstrated effectiveness in various protein classification tasks in Chen et al. [1].

In our study, we selected the following descriptor categories from iFeature:

- **Amino Acid Composition (AAC):** Computes the frequency of each of the 20 standard amino acids in a protein sequence. This results in a 20-dimensional feature vector and provides a global overview of the amino acid content.
- **Dipeptide Composition (DPC):** Calculates the frequency of all possible dipeptide pairs (20×20), capturing local sequence-order information.
- **Composition-Transition-Distribution (CTD):** Encodes physicochemical properties such as hydrophobicity, polarity, and side-chain volume using three statistical measures—composition, transition, and distribution—across predefined amino acid groups. This yields its features.

The resulting feature vector for each protein sequence is the concatenation of these descriptors, giving:

This comprehensive feature set captures both global composition and local sequence motifs, enabling the classifier to learn richer representations. A similar strategy has been previously employed in literature for protein disease classification Mostafa et al. [2].

0.5 Model Training and Evaluation

0.5.1 Training and Evaluation on AAC Features

For training the CNN-BiLSTM model on the AAC (Amino Acid Composition) feature set, we employed the label encoding technique to convert class labels into numeric form. The encoded classes were then used for supervised classification.

The input feature vectors were reshaped to a three-dimensional format suitable for convolutional layers:

- Shape: $(3103, 20, 1)$

The model architecture is summarized as follows:

1. **Input Layer**: Accepts reshaped AAC vectors.
2. **1D Convolution Layer**: 64 filters, kernel size of 3, ReLU activation.
3. **MaxPooling1D**: Pool size of 2.
4. **Bidirectional LSTM Layer**: 64 units, returns sequences.
5. **Attention Mechanism**: Uses dense attention weights and multiplies them with the BiLSTM output.
6. **Fully Connected Layer**: 64 units with ReLU activation.
7. **Dropout Layer**: 50% dropout to prevent overfitting.
8. **Output Layer**: Softmax activation over 3 classes.

The model was compiled and trained using the following parameters:

- Optimizer: `Adam`
- Loss Function: `Sparse Categorical Crossentropy`
- Epochs: 50
- Batch Size: 32

Evaluation: After training, the model was evaluated on the test set. The final test accuracy achieved was:

Test Accuracy: `0.6418`

Training Curves: Figure 5 illustrates the model's accuracy and loss over training epochs.

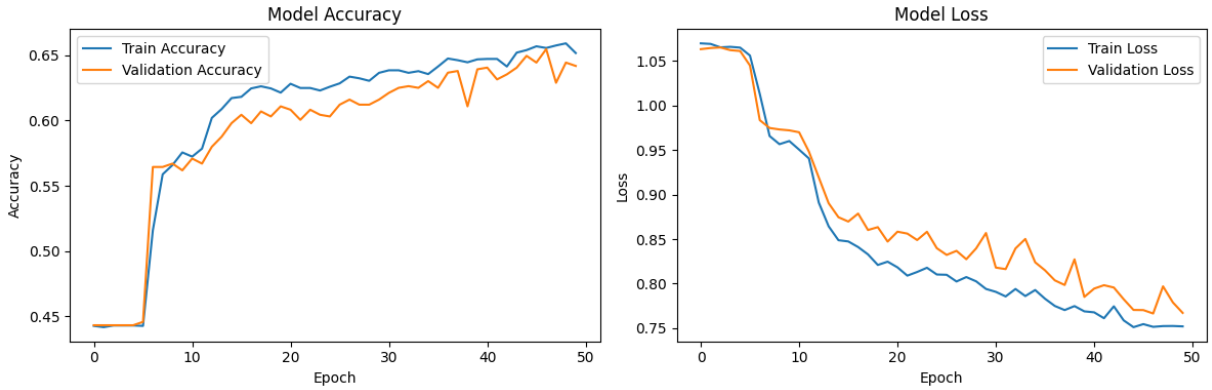

Figure 5: Training and Validation Accuracy and Loss for CNN-BiLSTM model on AAC features

0.5.2 Training and Evaluation on combined AAC, DPC and CTDC Features

For training the CNN-BiLSTM model on the AAC , DPC ,CTDC feature set, label encoding was applied to convert categorical class labels into numeric values suitable for classification tasks.

The DPC feature vectors were reshaped to match the expected three-dimensional input of the convolutional layers:

- Shape: $(3103, 459, 1)$

The model architecture used for this feature set was structured as follows:

1. **Input Layer**: Accepts reshaped DPC vectors.
2. **1D Convolution Layer**: 64 filters with a kernel size of 3 and ReLU activation.
3. **MaxPooling1D**: Pool size of 2 to reduce dimensionality.
4. **Bidirectional LSTM Layer**: 64 units configured to return sequences.
5. **Attention Mechanism**: Learns importance weights over BiLSTM outputs.
6. **Fully Connected Layer**: 64 units with ReLU activation function.
7. **Dropout Layer**: Dropout rate of 0.5 to mitigate overfitting.
8. **Output Layer**: Softmax activation to classify into 3 output classes.

The model was compiled and trained using the following configuration:

- Optimizer: `Adam`
- Loss Function: `Sparse Categorical Crossentropy`
- Epochs: 50
- Batch Size: 32

Evaluation: Upon completion of training, the model was evaluated on the reserved test dataset. The resulting performance was:

Test Accuracy: `0.7423`

Hyperparameter Tuning

We used **Keras Tuner** with the **Hyperband** algorithm for hyperparameter tuning. Hyperband is a multi-fidelity strategy that speeds up tuning by pruning poorly performing configurations early and allocating more resources to promising ones.

- **Objective:** Validation Accuracy
- **Maximum epochs:** 30
- **Factor:** 3
- **Tuned Parameters:**
  - Number of Conv1D filters: {32, 64, 96, 128}
  - Kernel size: {3, 5, 7}
  - LSTM units: {40, 80, 120, 160}
  - Dense units: {32, 64, 96, 128}
  - Dropout rate: [0.3, 0.7]

Best Model Architecture through Hyperparameter Tuning

The architecture of the CNN-BiLSTM model with attention used for classification from hyperparameter tuning is outlined below:

- **Input Layer:** Shape = (459, 1)
- **Conv1D:** 32 filters, kernel size 3, ReLU activation
- **MaxPooling1D:** Pool size = 2
- **Bidirectional LSTM:** 160 units, return sequences = True
- **Attention Mechanism:**
  - Dense layer with 1 unit
  - Flatten layer
  - Sigmoid activation
  - RepeatVector and Permute for attention weights
  - Multiply with BiLSTM output
  - Lambda layer to extract last time step
- **Dense Layer:** 128 units, ReLU activation
- **Dropout:** Rate = 0.5
- **Output Layer:** Dense with softmax activation (3 classes)

Total trainable parameters: 289,092

Evaluation: After hyperparameter tuning, the best model was evaluated on the reserved test dataset. The resulting performance was:

Test Accuracy: `0.7520`

Conclusion

This project we used comprehensive approach to solving a multi-class classification problem using a deep learning model that combines Convolutional Neural Networks (CNN), Bidirectional Long Short-Term Memory (BiLSTM) networks, and an attention mechanism. The methodology began with data collection from uniprot, data visualization, using iFeature for feature extraction.We trained the cnn-lstm model on AAC features, then on combined features from AAC,DPC and CTDC and compared their results.

To optimize model performance, we employed Keras Tuner with the Hyperband algorithm, which enabled the efficient exploration of a wide hyperparameter search space. Through this process, we identified an optimal configuration that achieved a test accuracy of 75.20%, outperforming traditional baselines and showcasing the efficacy of the hybrid model.

In conclusion, CNN-LSTM architecture, along with robust preprocessing and automated hyperparameter tuning, forms a scalable and adaptable framework for future deep learning applications. Future enhancements could include incorporating pre-trained embeddings, advanced regularization techniques, and exploring alternative attention strategies to further boost performance.

Bibliography

[1] Chen, Z., Zhao, P., Li, F., Leier, A., Marquez-Lago, T. T., Wang, Y., ... & Song, J. (2018). iFeature: a Python package and web server for features extraction and selection from protein and peptide sequences. *Bioinformatics*, 34(14), 2499–2502.

[2] Mostafa, F. A., Afify, Y. M., Ismail, R. M., & Badr, N. L. (2022). Deep Learning Model for Protein Disease Classification. *Current Bioinformatics*, 17(2), 245–253.