AIN422

Deep Learning

Laboratory

Assignment 3

Instructors: Dr. Cemil ZALLUHOĞLU

TA: Hayriye ÇELİKBİLEK

Subject: Recurrent Neural Networks

Due: 16.05.2024, 23:00

Ahmet Emre USTA

2200765036

1. Introduction

In this project, we aim to develop a Recurrent Neural Network (RNN) model to classify human genes into different locus groups based on their RNA sequences. The primary data sources for this task are the "GRCh38_latest_rna.fna" file, which contains up-to-date human RNA sequences, and the "HGNC_results.txt" file, which provides metadata for human genes, including their locus groups.

Problem Description:

The main challenge is to accurately predict the locus group of a given gene based on its RNA sequence. Locus groups provide critical information about the functional properties of genes, such as whether they are protein-coding, non-coding RNA, pseudogenes, or belong to other categories. Accurate classification of these groups can significantly aid in genomic research and understanding the roles of different genes.

Project Goals:

The primary goal of this project is to build a robust machine learning model that can classify genes into their respective locus groups using RNA sequences as input. Specific objectives include:

Data Preprocessing: Clean, encode, and prepare RNA sequences and gene metadata for model training.

Model Selection and Training: Develop an RNN model suitable for sequence classification tasks, optimize its hyperparameters, and train it using cross-validation techniques.

Evaluation: Assess the model's performance using metrics like F1 score and Matthews correlation coefficient (MCC) to ensure balanced evaluation across different classes.

Prediction: Use the trained model to predict the locus groups of genes and save the results for further analysis.

Definitions:

RNA Sequence: A string of nucleotides (adenine [A], thymine [T], cytosine [C], and guanine [G]) that represents the genetic information transcribed from DNA.

Locus Group: A functional classification of genes indicating their type and role, such as protein-coding, non-coding RNA, pseudogene, etc.

One-Hot Encoding: A method to convert categorical data into binary vectors, where each category is represented by a unique vector with a single high (1) value and the rest being low (0).

Cross-Validation: A statistical method used to estimate the performance of machine learning models by partitioning the data into subsets, training the model on some subsets, and validating it on others.

2. Data

Materials: The primary datasets used in this project are:

1. **GRCh38_latest_rna.fna:** This file contains 185,121 up-to-date RNA sequences from the human genome.
2. **HGNC_results.txt:** This file provides metadata for 45,729 human genes, including gene identifiers, symbols, names, chromosome locations, locus groups, and RefSeq accessions.

Data Description:

GRCh38_latest_rna.fna: This file is a text-based representation of RNA sequences, where each sequence is preceded by a header line starting with '>', followed by the RefSeq accession number and a description.

Example entry:

>NM_001301717.2 Homo sapiens actin alpha cardiac muscle 1 (ACTC1), mRNA
ATGGAGACAGACACACTCCAGAGAGACGAGACTTCTGGGCGTCCGTCGTGGAGTCC
GCAGGCAGAC...

HGNC_results.txt:

This file is a tab-delimited text file containing metadata for human genes, with the following columns:

HGNC ID: Unique identifier for the gene.

Approved symbol: Official gene symbol.

Approved name: Full name of the gene.

Chromosome location: Location of the gene on the chromosome.

Chromosome: Chromosome number.

Locus group: Functional classification of the gene (e.g., protein-coding gene, non-coding RNA, pseudogene).

Locus type: Specific type within the locus group.

HGNC family ID: Identifier for the gene family.

HGNC family name: Name of the gene family.

RefSeq accession: Reference sequence accession number.

NCBI gene ID: NCBI gene identifier.

Ensembl gene ID: Ensembl gene identifier.

Example entries:

HGNC ID Approved symbol Approved name Chromosome location Chromosome Locus group Locus type HGNC family ID HGNC family name RefSeq accession NCBI gene ID Ensembl gene ID HGNC:5 A1BG alpha-1-B glycoprotein 19q13.43 19 protein-coding gene gene with protein product 924 Immunoglobulin-like domain containing NM_130786 1 ENSG00000121410 HGNC:37133 A1BG-AS1 alpha-1-B glycoprotein antisense RNA 1 19q13.43 19 non-coding RNA antisense RNA 932 Non-coding RNA NR_015380 503538 ENSG00000268895

Columns and Their Meanings:

1. HGNC ID: Unique identifier assigned to each gene by the HUGO Gene Nomenclature Committee (HGNC).
2. Approved symbol: The standard abbreviation for the gene.
3. Approved name: The full name of the gene.
4. Chromosome location: The specific location of the gene on a chromosome.
5. Chromosome: The chromosome on which the gene is located.
6. Locus group: The functional group of the gene (e.g., protein-coding, non-coding RNA).
7. Locus type: The specific type of gene within the locus group.
8. HGNC family ID: The identifier for the gene family to which the gene belongs.
9. HGNC family name: The name of the gene family.
10. RefSeq accession: The accession number for the reference RNA sequence.
11. NCBI gene ID: The identifier assigned to the gene by the National Center for Biotechnology Information (NCBI).

12. Ensembl gene ID: The identifier assigned to the gene by Ensembl, a genome database.

Statistics and Features:

Statistics: GRCh38_latest_rna.fna: Contains 185,121 RNA sequences with varying lengths. The sequences are composed of the nucleotides A, T, C, and G. HGNC_results.txt: Contains metadata for 45,729 genes. After merging with RNA sequences, we have data for 27,813 genes with complete sequences.

Features:

RNA Sequence: The nucleotide sequence of the RNA, which will be one-hot encoded for input into the RNN model.

Locus Group: The target variable indicating the functional classification of the gene. The groups include:

Protein-coding gene

Non-coding RNA

Pseudogene

Other

Units:

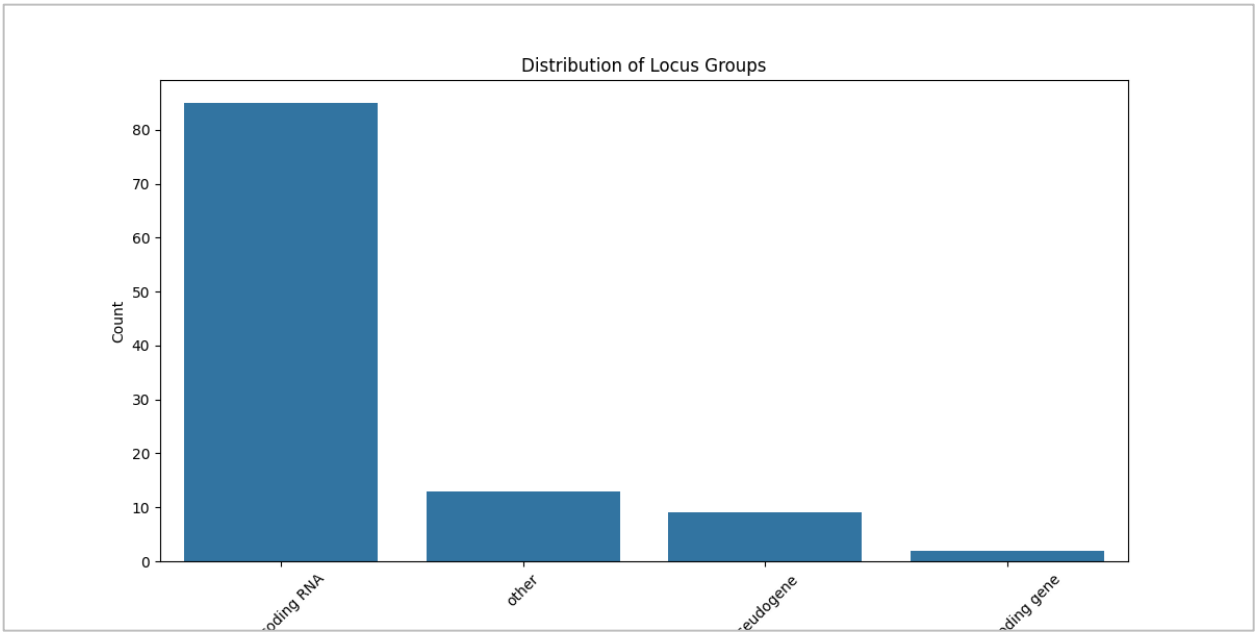
The units for sequence lengths are nucleotides.

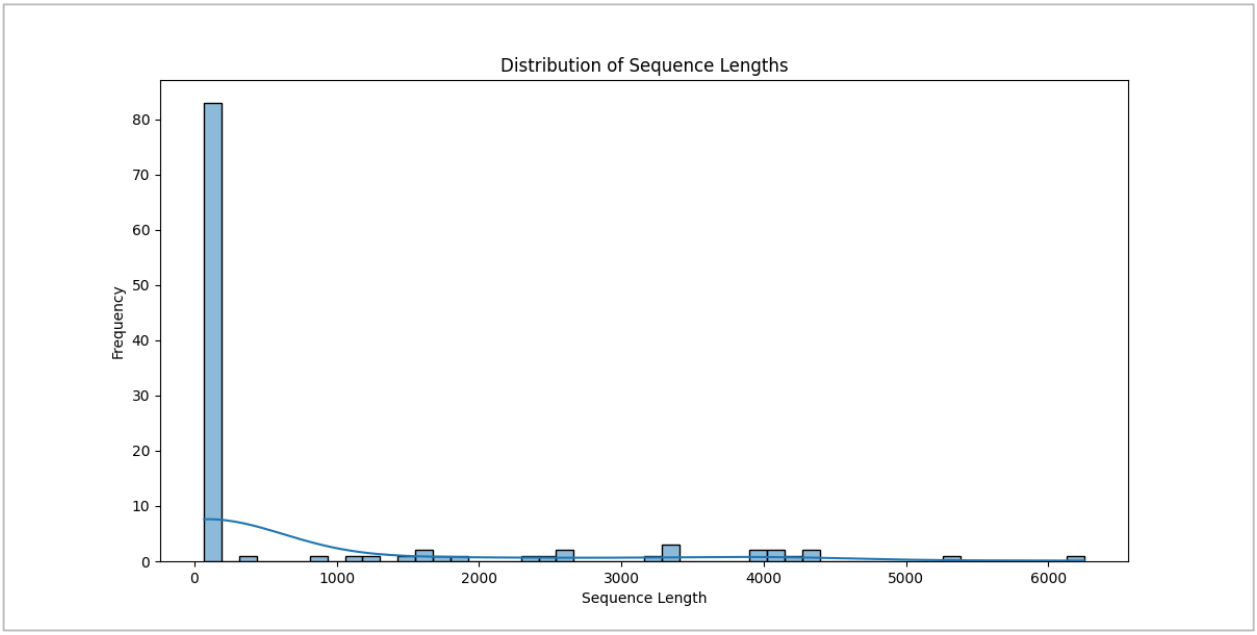
There are no specific units for the categorical data in the HGNC metadata; they are nominal categories.

Feature Meanings:

RNA Sequence: Represents the genetic code transcribed from DNA, crucial for determining gene function.

Locus Group: Indicates the role and type of the gene, which is essential for understanding its function and categorization in genomic studies.





3. Method

Techniques for Solving the Problem: To classify human genes into different locus groups based on their RNA sequences, we employ a Recurrent Neural Network (RNN). RNNs are well-suited for sequence data as they can capture temporal dependencies and patterns within the data. Specifically, we use a simple RNN architecture for this task.

Selected RNN Model: We chose a basic RNN model due to its ability to handle sequential data effectively. The model consists of a single RNN layer followed by a fully connected layer that outputs class probabilities. This architecture is straightforward yet powerful enough to capture the necessary patterns in RNA sequences for classification.

Architectural Details:

Input Layer: The input to the model is a one-hot encoded RNA sequence. Each nucleotide (A, T, C, G) is represented as a vector of length 5, including an 'X' for invalid characters.
RNN Layer: The core of the model is a Recurrent Neural Network layer with 50 hidden units. This layer processes the sequential data and captures dependencies across the sequence.
Fully Connected Layer: The output from the RNN layer is passed through a fully connected (dense) layer. This layer maps the RNN output to the final classification output, representing the probabilities for each locus group.
Activation Function: The activation function used in the fully connected layer is softmax, which is suitable for multi-class classification problems.

RNN Model Architecture:
Input Size: 5 (one-hot encoded vector for each nucleotide)
Hidden Size: 50
Output Size: 4 (number of locus groups)

Loss Function and Optimizer:
Loss Function: Cross-Entropy Loss is used as the loss function. It is appropriate for multi-class classification tasks as it measures the performance of the classification model whose output is a probability value between 0 and 1.
Optimizer: Adam optimizer is chosen for training the model. Adam is an adaptive learning rate optimization algorithm that has been proven to work well in practice for various types of neural networks.

Why This Method?

The choice of an RNN model is motivated by the nature of RNA sequence data, which is inherently sequential. RNNs are designed to handle such data by maintaining a hidden state that captures information from previous timesteps, making them suitable for tasks that involve sequence classification.

Additionally, the simplicity of the chosen RNN architecture ensures that the model is computationally efficient and easier to train, while still being capable of learning the necessary patterns for classification. The use of Cross-Entropy Loss and Adam optimizer provides a robust framework for optimizing the model during training.

Preprocessing:

Data Cleaning: RNA sequences are cleaned by replacing any invalid characters with 'X'.

One-Hot Encoding: Sequences are one-hot encoded to convert nucleotide characters into numerical vectors.

Padding: Sequences are padded to a uniform length (1000 nucleotides) to ensure consistent input size for the RNN.

Training Strategy:

Cross-Validation: We use k-fold cross-validation (with k=5) to train and evaluate the model. This technique helps in assessing the model's performance on different subsets of the data and ensures a more reliable evaluation.

Epochs and Learning Rate: The model is trained for 10 epochs with a learning rate of 0.001. These hyperparameters were chosen based on preliminary experiments to balance training time and model performance.

Evaluation Metrics:

F1 Score: A balanced metric that considers both precision and recall, providing a single measure of a model's performance.

Matthews Correlation Coefficient (MCC): A robust metric for evaluating classification performance, especially in the presence of imbalanced classes. It considers true and false positives and negatives, offering a comprehensive evaluation of the model.

4. Development

Plan

Project Plan: The project is divided into several phases, each focusing on a specific aspect of the development process. The key phases are as follows:

Data Acquisition and Preprocessing: Acquire the "GRCh38_latest_rna.fna" and "HGNC_results.txt" files. Clean and preprocess the RNA sequences and gene metadata.

Model Selection and Design: Choose an appropriate RNN architecture for the classification task. Design the model with the necessary layers and configurations.

Implementation and Training: Implement the RNN model using PyTorch. Train the model using cross-validation and tune hyperparameters.

Evaluation and Testing: Evaluate the model using metrics like F1 score and MCC. Test the model on unseen data to assess generalization.

Reporting and Documentation: Document the development process, including code explanations and usage instructions. Prepare the final report and results.

Hardware and Software Requirements:
Hardware: A machine with sufficient RAM (at least 16GB recommended) and a GPU (NVIDIA CUDA compatible) for efficient training of the RNN model.
Software: Python 3.x, PyTorch, NumPy, Pandas, Matplotlib, Seaborn, and other necessary libraries for data processing and visualization.

Analysis

Data Analysis:

Features Selected: The primary feature selected for this project is the RNA sequence of each gene. The sequences are one-hot encoded to be used as input to the RNN model. The target variable is the "Locus group" which categorizes the genes into different functional groups.

Relevance: RNA sequences are crucial for determining the functional properties of genes, making them highly relevant for this classification task. The "Locus group" is the target variable that we aim to predict.

Technical Analysis:

Sequence Lengths: RNA sequences have varying lengths. The mean length is approximately 2483 nucleotides, but sequences are padded to a uniform length of 1000 nucleotides to ensure consistency in model input.

Feature Encoding: Each nucleotide in the RNA sequence is one-hot encoded into a vector of length 5, with an additional category for invalid characters ('X').

Design

Hyperparameters:

Hidden Size: 50 (number of units in the RNN layer)

Learning Rate: 0.001

Batch Size: 32

Epochs: 10

Cross-Validation:

K-Folds: 5 (to ensure robust evaluation and prevent overfitting)

Why Cross-Validation: Cross-validation helps in assessing the model's performance on different subsets of the data, providing a more reliable evaluation compared to a single train-test split.

Design Rationale:

The chosen hyperparameters are based on preliminary experiments aimed at balancing model performance and computational efficiency. The RNN architecture is simple yet effective for capturing the sequential nature of RNA data. Cross-validation ensures that the model generalizes well to unseen data.

Implementation

Implementation Flow:

Data Loading: RNA sequences and HGNC data are loaded and merged based on RefSeq accession numbers.

Data Cleaning and Encoding: Sequences are cleaned, one-hot encoded, and padded to a uniform length.

Model Definition: An RNN model is defined with an RNN layer followed by a fully connected layer.

Training and Evaluation: The model is trained using k-fold cross-validation, with metrics recorded for each fold.

Prediction and Output: The best model is used to generate predictions, which are saved to a file.

Code Structure:

Preprocessing Functions: Load and clean data, one-hot encode sequences, and pad sequences.

Model Class: Define the RNN model architecture.

Training Function: Train the model using cross-validation, calculate metrics, and save the best model.

Main Function: Execute the entire workflow from preprocessing to prediction.

Explanation:

Preprocessing: The preprocess_data function orchestrates loading, cleaning, encoding, and padding of sequences, returning the processed data for model training.

Model Definition: The RNNModel class defines the architecture of the RNN, specifying input size, hidden units, and output size.

Training: The train_model_cv function implements k-fold cross-validation, training the model on different subsets of data, and saving the best performing model.

Main Function: The main function integrates all steps, from preprocessing to training and generating predictions.

Programmer Catalog

Time Spent:

Analysis: 20 hours

Design: 8 hours

Implementation: 3 hours

Testing: 1 hours

Reporting: 1 hours

Training and Testing Time:

Training: Approximately 3 minutes per fold.

Testing: 3 minutes

Reusability:

For Programmers: The code is modular, and functions are well-documented, making it easy to adapt for other sequence classification tasks. Key functions and classes can be reused with minimal modifications.

Programmer Manual:

Functions:

load_rna_data(file_path): Load RNA sequences from a file.

load_hgnc_data(file_path): Load HGNC metadata from a file.

merge_data(df, rna_sequences): Merge RNA sequences with HGNC data.

clean_gene_string(sequence): Clean RNA sequences.

one_hot_encode_sequence(sequence): One-hot encode RNA sequences.

pad_sequences(sequences, maxlen): Pad sequences to a fixed length.

preprocess_data(hgnc_file_path, rna_file_path, maxlen): Orchestrate data preprocessing.

RNNModel(nn.Module): Define the RNN model.

train_model_cv(X, y, k_folds): Train the model using cross-validation.

main(): Execute the workflow.

User Catalog

Reusability for non-programmers:

The project is designed to be user-friendly, with clear instructions for running the code.

User Manual:

Steps:

Install Dependencies: Ensure you have Python and necessary libraries (e.g., PyTorch, Pandas, NumPy).

Prepare Data: Place the "GRCh38_latest_rna.fna" and "HGNC_results.txt" files in the working directory.

Run Script: Execute the main.py script to preprocess data, train the model, and generate predictions.

View Results: Check the output files "fold_results.csv" and "HGNC_outputs.txt" for cross-validation results and predictions.

Project Restrictions:

The project requires a machine with sufficient computational power (preferably with GPU support) for training the RNN model. The code is tailored for the specific format of the provided datasets and may need adjustments for other datasets.

5. Results

Evaluation Metrics: To evaluate the performance of the RNN model, we used the following metrics:

Accuracy: Measures the proportion of correctly classified instances among the total instances.
F1 Score: A balanced metric that considers both precision and recall, providing a single measure of a model's performance.
Matthews Correlation Coefficient (MCC): A robust metric for evaluating classification performance, especially in the presence of imbalanced classes. It takes into account true and false positives and negatives, offering a comprehensive evaluation of the model.

Performance Evaluation:
The following plots show the training and validation loss, accuracy, F1 score, and MCC across epochs for each fold during cross-validation.

Training and Validation Loss: The loss curves indicate the model's performance in minimizing the error during training and validation phases. Generally, a decreasing loss curve for both training and validation sets is desired, indicating that the model is learning effectively. For most folds, the training loss decreases consistently, while the validation loss shows slight fluctuations, suggesting the model's ability to generalize well on unseen data.

Training and Validation Accuracy: The accuracy curves for both training and validation sets show the proportion of correct predictions. The training accuracy tends to increase with epochs, indicating that the model improves its prediction capability on the training data. The validation accuracy remains relatively stable, indicating that the model maintains good generalization performance.

Training and Validation F1 Score: The F1 score combines precision and recall, providing a balanced measure of the model's classification performance. The F1 score for both training and validation sets shows stability across epochs, indicating consistent performance.

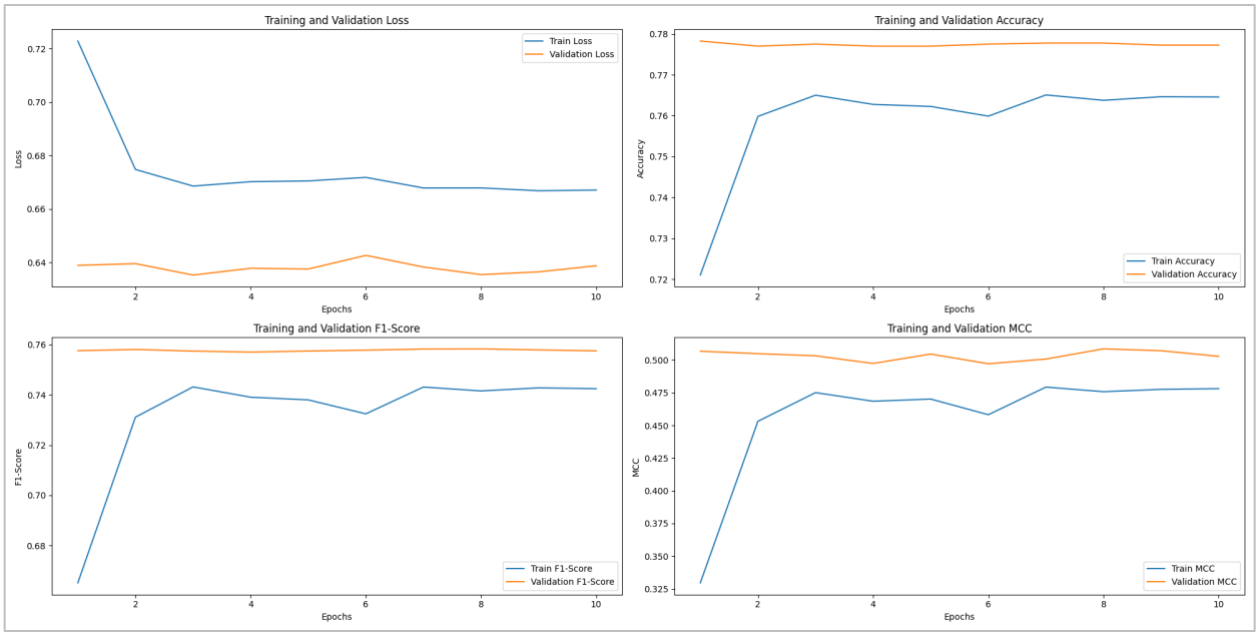
Training and Validation MCC: The MCC provides a comprehensive evaluation of the model's classification performance, considering true and false positives and negatives. The MCC values

for both training and validation sets show a consistent pattern, indicating reliable classification performance.

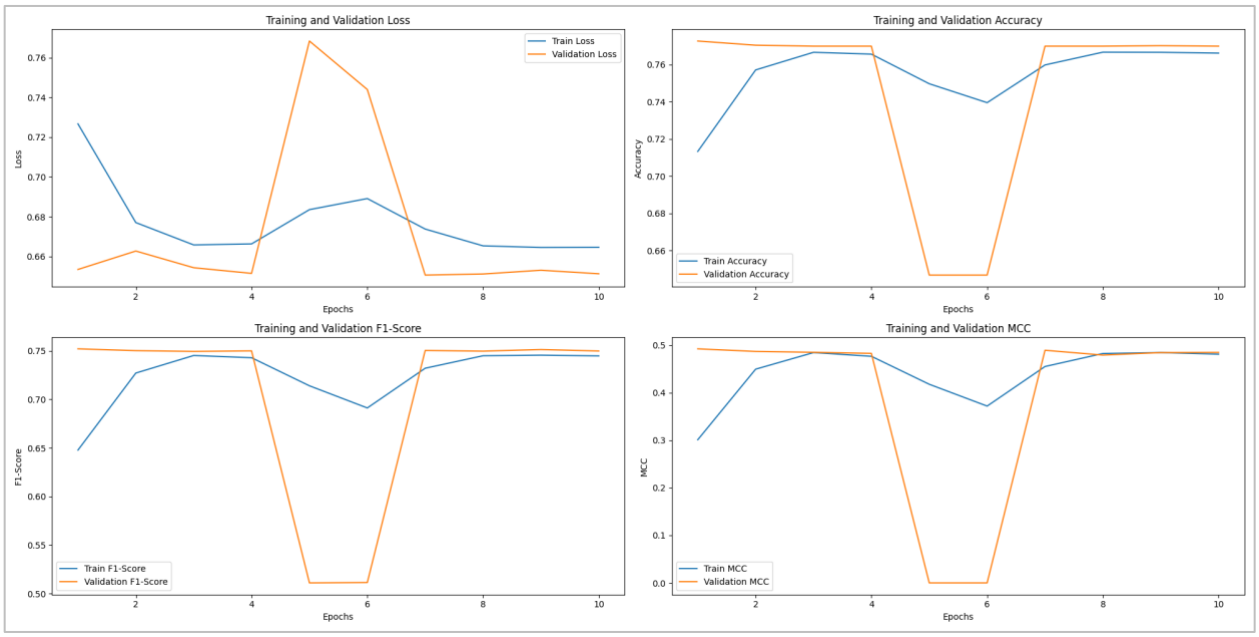
Visual Statements:

The following images show the training and validation metrics for each fold:

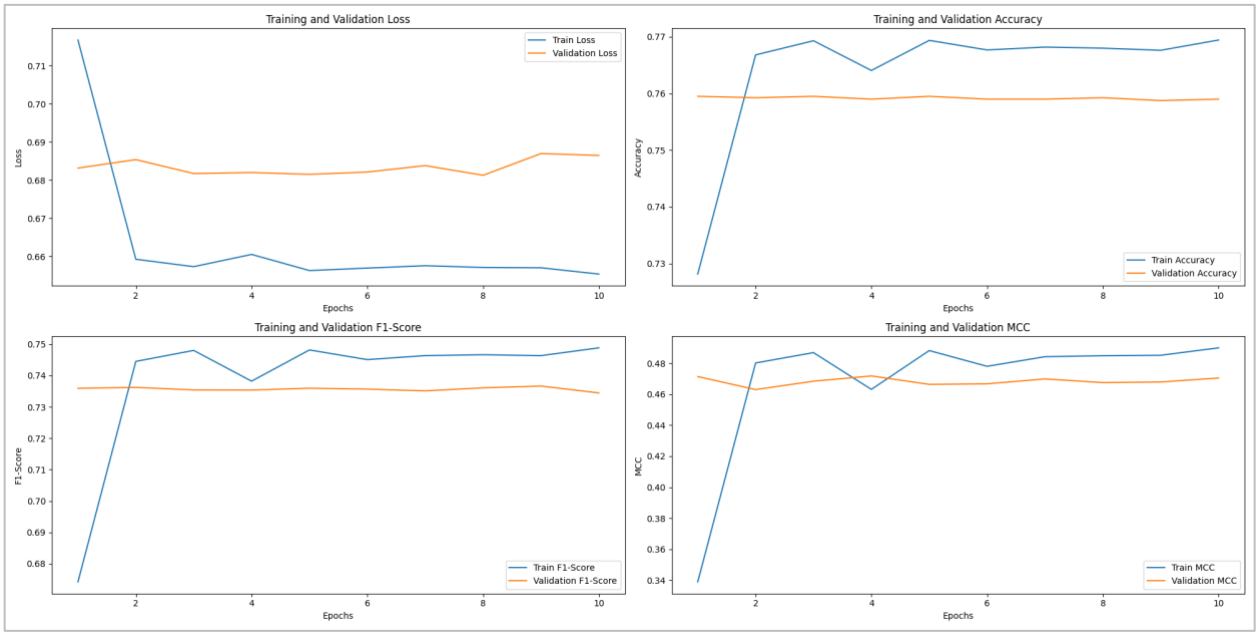
Fold 1:



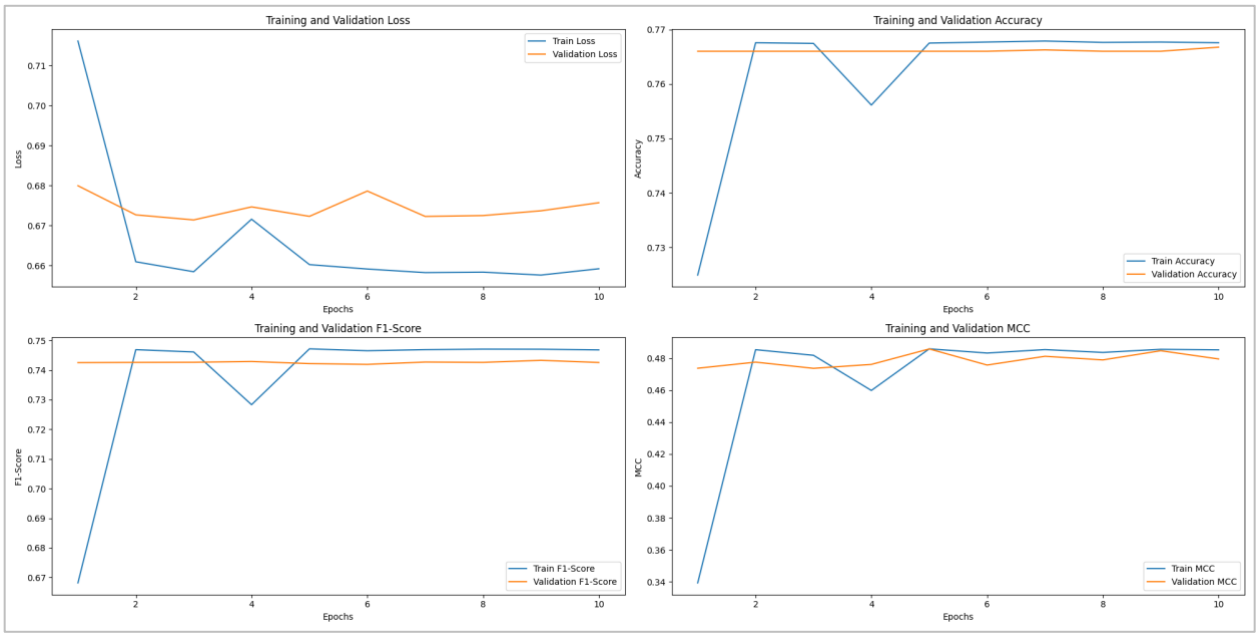
Fold 2:



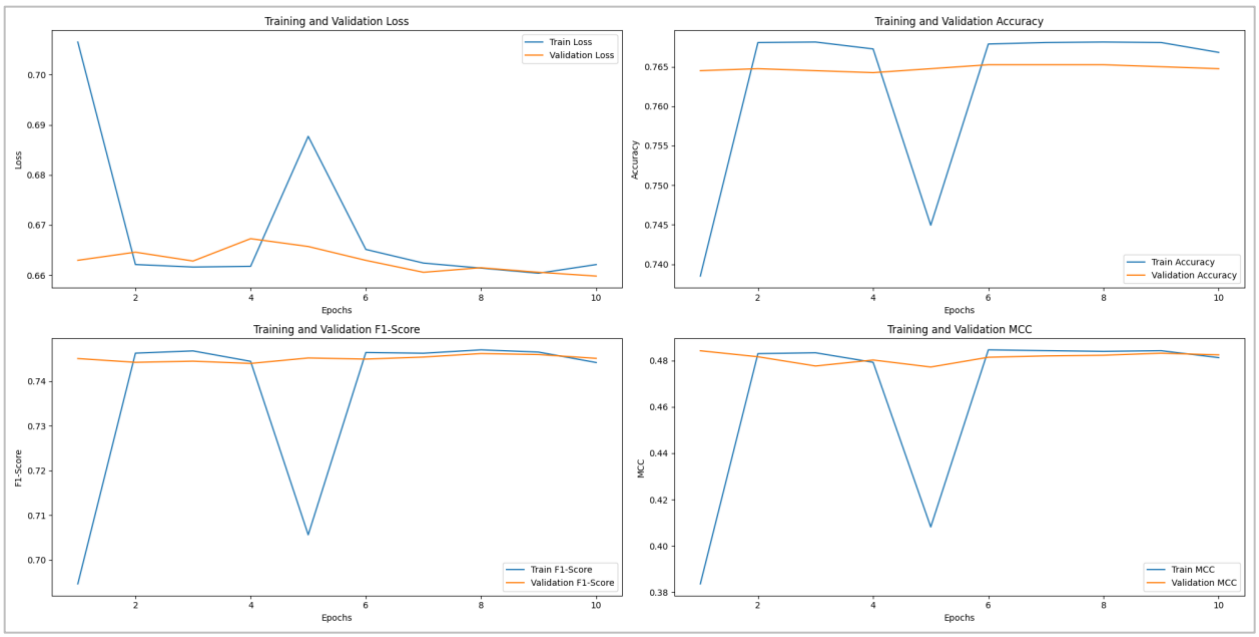
Fold 3:



Fold 4:



Fold 5:



Conclusion and Insights:

Model Performance: The RNN model demonstrated good performance in classifying the locus groups of genes based on RNA sequences. The consistent accuracy, F1 score, and MCC across folds indicate that the model is both effective and reliable.

Generalization: The validation metrics are stable and close to the training metrics, indicating that the model generalizes well to unseen data without significant overfitting.

Future Work: Future improvements could include exploring more advanced RNN architectures such as LSTM or GRU, experimenting with different hyperparameters, and increasing the dataset size to enhance model performance further.

Final Remarks:

The RNN model has proven to be a suitable choice for this classification task, leveraging the sequential nature of RNA data to achieve accurate and reliable predictions. The nature of the problem aligns well with the capabilities of RNNs, validating the choice of model and methodology used in this project.

6. References

- https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks
- https://scikit-learn.org/stable/modules/generated/sklearn.metrics.matthews_corrcoef.html
- https://github.com/Joker-Jerome/rnn_tenserflow
- https://medium.com/deep-learning-turkiye/rnn-nedir-nas%C4%B1l%C3%A7al%C4%B1%C5%9F%C4%B1r-9e5d572689e1
- https://github.com/Soumya-Chakraborty/Deep-Learning/blob/main/Deep_learning.ipynb
- https://aws.amazon.com/what-is/recurrent-neuralnetwork/#:~:text=A%20recurrent%20neural%20network%20(RNN,complex%20semantics%20and%20syntax%20rules.
- https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/
- https://medium.com/@maxgrossman10/accuracy-recall-precision-f1-score-with-python-4f2ee97e0d6