AFRICAN INSTITUTE FOR MATHEMATICAL SCIENCES

(AIMS RWANDA, KIGALI)

Name: Assignment Number: 2
Course: Introduction to Neural Networks Date: April 1, 2025

Groupe Name On Kaggle : Kristeva - Aubrey
Ranking (Score) On Kaggle : 1 (0.742)

1 Overview

This report analyzes the machine learning pipeline developed for classifying DNA sequences based on binding affinity. The project implements two distinct approaches: a standalone Convolutional Neural Network (CNN) model, and a hybrid model that combines a CNN with a Multi-Layer Perceptron (MLP) using the KMeans features. By integrating these models, we aim to enhance prediction accuracy and capitalize on their unique strengths for a more effective analysis of DNA sequences.

2 Data

- Training sequences (Xtr): DNA sequences containing 2000 sequences of 101 proteins
- Training labels (Ytr): Binary binding indicators that take **label 1** whether the transcription factor will bind or **label 0** if not
- Training KMeans features (Xtr_mat): 100 features per sequence for each sequence, we compute the average of the representations of all its subsequences to obtain the feature vector of this sequence.
- Testing equivalents (Xte, Xte_mat) for final prediction

3 Data Preprocessing

The process begins with DNA Sequence Encoding, where one-hot encoding is applied to convert DNA sequences into a numerical format. We save it in the X_seq_tensor of shape (N, 4, maxlen) a 3D numerical representation, where the first dimension indexes the sequences, the second dimension indexes the DNA base (A, C, G, T), and the third dimension indexes the position within the sequence. Following this, Feature Normalization is performed on the KMeans features, utilizing the StandardScaler and save in X_cluster_tensor (N, K)to ensure that all features contribute equally to the model training. This normalization step is crucial for maintaining balanced feature influence during the learning process. The shape of X_seq_tensor is (N, 4, maxlen), where N is the number of training samples. The shape of X_cluster_tensor is (N, K), where K is the number of KMeans clusters. Similar shapes apply to the test tensors.

After that, the data was divided into training and validation sets, allocating 70% for training and 30% for validation purposes, with a fixed random state of 42 to ensure reproducibility. A custom DualInputDataset class was developed to efficiently manage both the DNA sequences and KMeans features. Furthermore, a batch size of 32 was implemented to enhance training efficiency, facilitating stable and effective learning throughout the training process.

4 Model

After many experiments using either sequence data alone or KMeans-derived features alone, tested with MLP, CNN, KMeans, and KNN models, proved inconclusive, an ensemble approach was adopted. This ensemble averages the predictions of two models: a CNN-only model, excel at capturing fine-grained sequence motifs, and a CNN+KMeans model, designed to incorporate broader contextual information from the clustering features. The aim of this ensemble strategy is to achieve more robust and reliable results by leveraging the complementary strengths of both perspectives.

4.1 CNN-Only Model

This model processes one-hot encoded DNA sequences of size 4×101. It features two 1D convolutional layers with kernel sizes of 7 and 11, followed by ReLU activation, adaptive max pooling, and two fully connected layers. A sigmoid activation function is used for binary classification.

4.2 CNN+KMeans Fusion Model

This architecture includes dual input streams: a CNN branch with two 1D convolutional layers and batch normalization, and a MLP of three fully connected layers. In essence, the model learns features from DNA sequences using a CNN, takes pre-computed K-Means features as another input, concatenates these two feature sets, and then uses an MLP to process the combined information and produce a final prediction (a probability between 0 and 1).

4.3 Final Prediction Model

An ensemble approach was employed, generating predictions from both models. The probability outputs were averaged, and a threshold of 0.5 was applied for final binary predictions. This method capitalizes on the strengths of both models to improve accuracy.

4.4 Selection of Hyperparameters

Hyperparameters were selected after many experimentations. The CNN-only model utilized kernel sizes of 7 and 11, while the CNN+KMeans model used sizes 9 and 15. Learning rates of 0.001 for the CNN-only model and 0.002 for the CNN+KMeans model were assigned to enhance training effectiveness. Dropout rates were set at 0.3 for the CNN-only model and 0.4 and 0.3 for the CNN+KMeans model to address overfitting.

5 Results

Overall, these plots illustrate the effectiveness of integrating KMeans clustering features alongside CNN to enhance model performance in terms of both reducing training loss and improving validation accuracy. The CNN+KMeans Fusion Model shows improved predictive ability and

stability, making it a strong candidate for tasks requiring the analysis of complex data, such as biological sequence classification.
On the test data we have obtained these following results:

- The validation score on our validation score is : 70.69 %
- The public score on Kaggle is : 72.4 %
- The private score on Kaggle is : 74.2 %

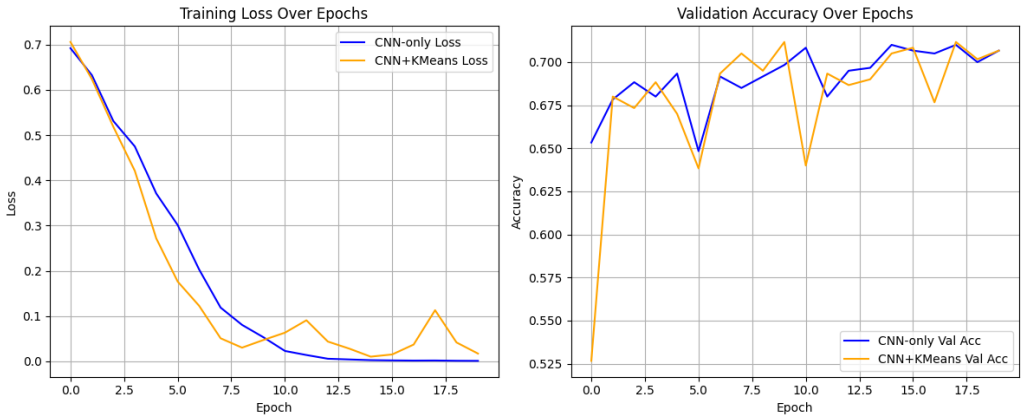

Figure 1: Training and validation graphs

6 Conclusion

Finally, by utilizing the sequence information with pre-computed KMeans features and combining predictions from both models, we have been able to maximize the accuracy and ensure the robustness in our predictions.

7 Future Improvements

- Explore attention mechanisms to focus on key binding regions
- Test deeper architectures with residual connections
- Test additional sequence encodings (k-mer frequency, position-specific scoring matrices)