RNA-binding Project - Deep Learning in Computational Biology

Haim Zisman and Ori Braverman

September 5, 2024

Abstract

In this project, we aim to develop a neural network model to predict RNA-binding intensities for RNA Binding Proteins (RBPs) using high-throughput RNA sequence data. Specifically, given a the HTR-SELEX dataset, with each RBP consist of 1-4 files, with oligonucleotide sequences, we train a binding model for each RBP. This model is then used to score RNAcompete probes, which are sequences ranging from 30 to 41 nucleotides in length, sorted lexicographically.

The primary challenges of this task are achieving high prediction performance and optimizing the training runtime. The goal is to assign binding intensities to RNAcompete probes and compare the predicted values with the real binding intensities using Pearson correlation. We are provided with 38 development sets (HTR-SELEX data combined with RNAcompete probes and their binding intensities) and 38 evaluation sets (HTR-SELEX data with RNAcompete probes without binding intensities). The model's performance will be assessed based on the accuracy of predicted binding intensities and the computational efficiency during training.

This work builds on the foundation of previous models like DeepSELEX, which demonstrated the effectiveness of deep neural networks in inferring DNA-binding preferences.

1 Baseline Method

The baseline algorithm provides a benchmark to evaluate the effectiveness of our methods in correlating RNAcompete scores with binding intensities for RNA-binding proteins (RBPs).

Algorithm 1 Compute Correlations for RNA-Binding Proteins (RBPs)

Input: *htr_selex_dir* (Directory containing sequence data) *sequences_file* (File containing RNA sequences) *intensities_dir* (Directory containing binding intensity data) *K_mer_len* (optional-Length of the k-mers - default 7) *num_rbps* (optional-Number of RBPs to process - default 38)
Output: A file with correlation values for each RBP.

Initialize an empty list *correlations*.
for each RBP i from 1 to *num_rbps* **do**
    Locate the sequence file paths for the last cycle in *htr_selex_dir*.
    Compute k-mer counts for SELEX sequences and sum them to get a score for each k-mer.
    Read RNAcompete sequences from *sequences_file*.
    Compute RNAcompete scores by dot product between RNAcompete k-mer counts and SELEX k-mer scores.
    Normalize RNAcompete scores by sequence length.
    Load binding intensities from *intensities_dir*.
    Calculate Pearson correlation between RNAcompete scores and binding intensities.
    Add the computed correlation.
end for
return *correlations*

2 Methodology

In this section we will present some of the models we used and the results of our experiments. All the methods are using cross-entropy except for the n-gram selection projuction. The output size of 5 presented in the pictures below is for illustrative purposes only (can be different - e.g. binary case).

2.1 Negative Examples

In this study, three methods were used to sample negative examples:

1. **Random Sampling (RANDOM)**: Negative examples are generated by creating random sequences of 40 nucleotides, each nucleotide being randomly chosen from {A, C, G, T}. This method produces sequences uncorrelated with the actual data, providing a baseline for comparison.

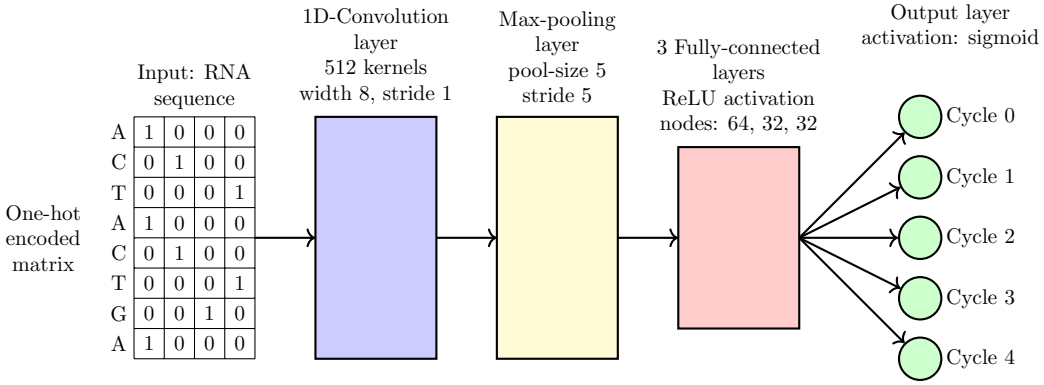
2. **Sampling from Another Cycle (FROM_CYCLE_1)**: Negative examples are sampled from sequences found in Cycle 1 files of other RBPs (RBPj_1.txt where $j \neq i$). These sequences are assumed to be less associated with the current RBP, making them suitable as negative examples.

3. **Markov Chain Sampling (MARKOV)**: A Markov chain model, built from Cycle 1 sequences, is used to generate negative examples. The model captures transition probabilities between nucleotides, producing sequences that are statistically similar to Cycle 1 data but distinct enough to serve as negative examples.

We mainly used methods 1 and 2 due to high time computation demand from the last approach.

2.2 Model 1: DeepSELEX

In this experiment, we simply aimed to replicate the results of the DeepSELEX model on the RNA-binding dataset.



The prediction procedure begins with an RNA sequence of length (L). During pre-processing, if (L) is less than (k), the sequence is padded, then the sequence divided into subsequences of length (k). For prediction, each subsequence pased through the network, and the results of all the subsequence are aggregated using the formula: score = $\max(\text{cycle}_4) + \max(\text{cycle}_3) - \min(\text{cycle}_0)$. The final output is the aggregated binding score for the RNA sequence.

2.3 Model 2: DNN-ngrams

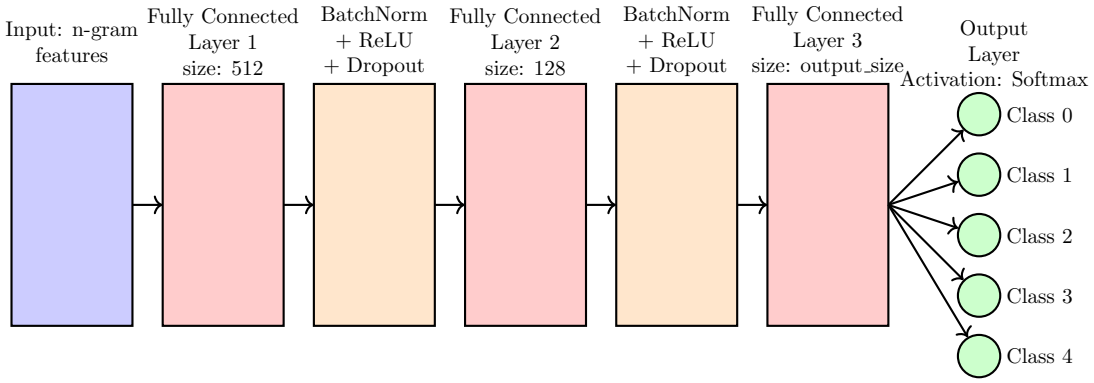
Feature Selection phase:

Algorithm 2 DNN-ngram Data pre-process of Feature Selection

Input: Sequences and their labels $S = \{(s_1, y_1), (s_2, y_2), \ldots, (s_m, y_m)\}$ where $\forall s_i : s_i \in \{A, C, T, G\}^T$
Output: The selected features $F \in R^k$
Initialize an empty set F_{all} to store all possible features
for $z = z_1$ to z_t **do**
    Compute all possible 4^z subsequences for length z
    Add these subsequences to F_{all}
end for
Total Features = $4^{z_1} + 4^{z_2} + \ldots + 4^{z_t}$
Initialize a `CountVectorizer` with the n-grams from all F_{all} and Count occurrences of F_{all} (`FitTransform`) in S.
SelectKBest selector: Select the top k features based on the chi-square test (evaluate independence between each feature in F_{all} and target labels Y) and store them in F
Return the selected features $F \in R^k$

This process ensures that the most relevant N-gram features are selected for the classification task, improving the model's performance by focusing on the most informative subsequences.

The classification task:



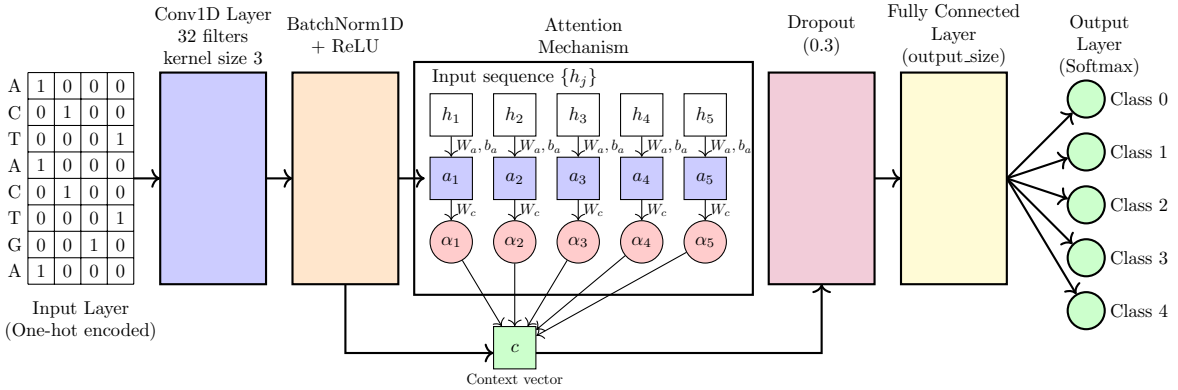
2.4 Model 3: CNN-Attention

We developed a CNN-Attention model to enhance performance on DNA sequence classification. This model combines the local pattern recognition capabilities of CNNs with the selective focus of attention mechanisms.

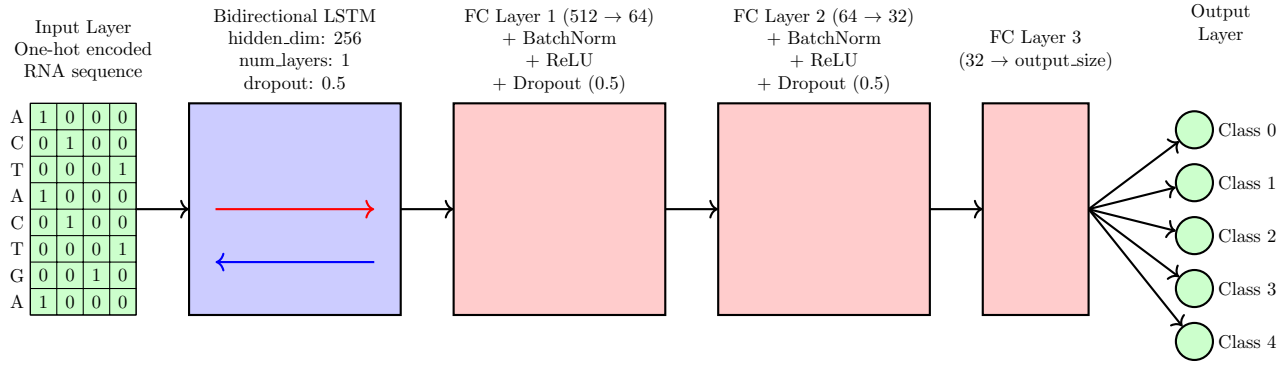
Attention Mechanism Explanation: Define h_j to be the output of the model after convolution, batch-norm + ReLU.

- Transform convolutional output: $a_j = \tanh(W_a \cdot h_j + b_a)$
- Calculate attention weights: $\alpha_j = \frac{\exp(W_c \cdot a_j)}{\sum_k \exp(W_c \cdot a_k)}$
- Compute context vector: $c = \sum_j \alpha_j \cdot h_j$

This model combines CNN's ability to capture local patterns with attention's focus on relevant sequence parts, enhancing DNA sequence classification.



2.5 Model 4: LSTM



2.6 Model 5: ngram-Selection-projection

Algorithm 3 N-Gram RNA Sequence Dataset Creation with Feature Selection

1: **Input:** HTR-SELEX Sequences and their labels $S = \{(s_1, y_1), (s_2, y_2), \ldots, (s_m, y_m)\}$ where $s_i \in \{A, C, T, G\}^T$, RNAcompete sequences $T = \{t_1, t_2, \ldots, t_N\}$
2: **Parameters:**

- K - number of selected features
- *binary* - whether binary vectors are used
- n_1 to n_2 - range of n-gram lengths

3: **Output:** RNA scores r
4: Convert all HT-SELEX sequences S into BOW feature matrix $\mathbf{X}$ using n-grams of lengths ranging from n_1 to n_2
5: Apply chi-squared feature selection on $\mathbf{X}$ to select top K features
6: Convert RNAcompete sequences T into BOW n-gram feature matrix T_{features} using the same features
7: Compute HTR-distribution features aggregation $\mathbf{f} = \sum \mathbf{X}$
8: Project T_{features} onto the HTR-SELEX feature manifold to calculate RNA scores: $r = \mathbf{f} \cdot T_{\text{features}}^{\top}$
9: **Return** RNA scores r

* The steps 3,4 of the detailed algorithm are similar to the DNN-ngram algorithm.

In our Optuna optimization for the n-gram selection model, we tested various parameters, including k-mer lengths (4-9), top-k values (1024-16348), and binary embeddings. The surprising result was that binary embedding consistently outperformed non-binary embedding. The highest average correlation of 0.2185 was achieved with binary embedding, k-mer lengths of 6-8, and a top-k value of 1055.

2.7 Normalization of Predicted RNA Scores

$$y = \left(\frac{x - \min x}{\max x - \min x} \right) \times (\max y - \min y) + \min y$$

We are mapping the results to a known range of intensities based on the RBPs known to us. x is the predicted_rna_score by any model, and y is the scaled and biased term.

2.8 Models configurations

Model	Hyperparameters				k	Dropout Rate
	Attention Dim	Input Size	Kernel Size	Learning Rate	Output Size	
cnn_attention	512	40	8	4.0e-05	-	-
deepselex_k_14	-	-	-	0.0001	14	-
deepselex_k_20	-	-	-	0.0001	20	-
deepselex_k_30	-	-	-	0.0001	30	-
deepselex_k_40	-	-	-	0.0001	40	-
dnn	-	2048	-	4.0e-05	-	0.25
lstm	-	-	-	0.0001	-	0.3

Table 1: Hyperparameter configurations for different model architectures

3 Results

Our study evaluated several neural network models for predicting RNA-binding intensities of RNA Binding Proteins (RBPs) using the HTR-SELEX dataset. We compared the performance of different architectures and parameter configurations across 38 RBPs.

3.1 Model Comparison

We compared the average correlation across all RBPs for different model architectures:

Model	Average Correlation	Average Time Consumption per RBP
ngram_selection	0.2185	2 min
deepselex_k_30	0.1424	25 min
deepselex_k_20	0.1421	25 min
deepselex_k_14	0.1267	25 min
deepselex_k_40	0.1226	25 min
dnn	0.1204	15 min
cnn_attn	0.1195	25 min
lstm	0.0798	55 min

Table 2: Average correlation and time consumption across RBPs for different model architectures

The ngram selection model outperformed other architectures, achieving the highest average correlation of 0.2185.

4 Code

All deep learning models were trained using a GeForce GTX 3090 GPU. In order to run our model, you should go to the folder where the `main.py` file is located. Before running the model, ensure that all necessary dependencies are installed by using the attached `requirements.txt` file. You can install the required packages by running the following command in the terminal:

```
pip install -r requirements.txt
```

After installing the dependencies, you can run the best model using the following command:

```
python3 main.py rnacompete rbns_input ... rbns_highest_concentration
```

Please note that the data files `htr-selex/`, `RNAcompete_intensities/`, and `RNAcompete_sequences_rc.txt` must be located in the project directory.

To run any deep learning model, this command can be used.

```
python3 train.py rnacompete cycle1 cycle2 .. cyclek --model model_name
```

References

[1] M. Asif and Y. Orenstein, *DeepSELEX: inferring DNA-binding preferences from HT-SELEX data using multi-class CNNs*, School of Electrical and Computer Engineering, Ben-Gurion University of the Negev, Beer-Sheva 8410501, Israel.