Grace Nye
RBP Binding Prediction

Introduction:

RNA-binding proteins (RBPs) are common throughout eukaryotic genomes (around 5-10% of the proteome), and they have been found to have many key roles in biological processes. Experimental methods to detect which RNA sequences bind to given RBPs are costly and time-consuming. Therefore, I constructed a CNN using as data a large-scale CLIP-seq dataset and corresponding base-pairing data calculated by Eternafold. Accuracy of the final model was 0.87.

Data Wrangling:

There were 96 datasets examined for this study, comprising four datasets per RBP, with 24 RBPs total. Since each model built is specific to a single RBP, I decided to move forward with four datasets in particular, and the final model was built and trained on a single model. The datasets were downloaded from GraphProt. Each RBP had four fasta files, each containing a list of RNA sequences. The sequences were obtained from high-throughput CLIP-seq experiments which quantified the binding of RNA sequences to an RBP of interest. The four fasta files were as follows: 1) training sequences that *do* bind the RBP ("positive" sequences); 2) training sequences that do *not* bind the RBP ("negative" sequences); 3) a small external dataset containing positive sequences; 4) a small external dataset containing negative sequences. All the sequences were formatted into a dataframe with three columns: sequence ID, dataset of origin, and sequence. Finally, the length of all sequences were calculated and plotted (Figure 1).

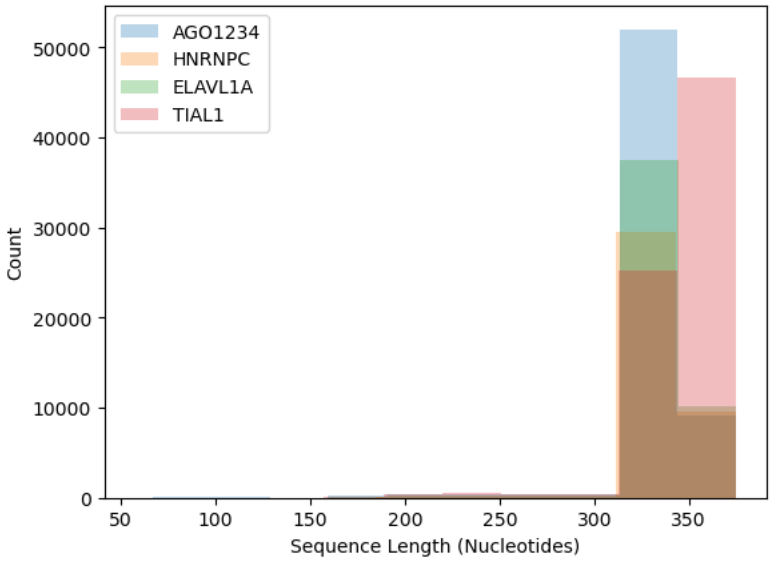

Figure 1: Sequence length distributions throughout the datasets examined. The majority of sequences for all datasets are around 300-350 nucleotides in length, although there are a few outliers as short as 50 nucleotides in all datasets.

In order to prevent redundancy in the dataset, the program cd-hit-est was used to cluster all sequences and filter out those with over 80% sequence similarity (this also helps prevent data leakage between training/testing datasets).

Exploratory Data Analysis

First, because all sequences must be the same length for running a CNN, sequences were padded to same length by adding an 'n' onto each sequence. Sequences were all padded to 400 nucleotides.

Next, sequences were all run through Eternafold to calculate their base pairing probability. As RNA sequences are known to form quite complex structures, various bases pairing with one another, and this structure undoubtedly plays a role in protein binding, I hypothesized that having that data would enhance the predictive power of the model. Eternafold is a model that can calculate base pairing probabilities of RNA sequences.[i] Figure 2 shows an example of the data that is gathered by this calculation.

Figure 2: Base pairing probabilities calculated for each nucleotide of an RNA sequence. This graph in particular is showing the probability that each nucleotide in the RNA sequence is unpaired. Higher values (darker orange) indicate a higher probability that this nucleotide is freely floating in solution, whereas smaller values (lighter) indicate a higher probability that this base is pairing with another and is therefore structured.

Preprocessing

In order for the data to be run through the CNN in the correct structure the sequences need to be converted to numerical data. Therefore, each nucleotide is represented as a 1D vector. Adenine (A) is represented as [1,0,0,0]. Cytosine (C) is represented as [0,1,0,0]. Guanine (G) is represented as [0,0,1,0]. Uridine (U) is represented as [0,0,0,1]. The extra nucleotides 'N' used for padding are represented as [0.25, 0.25, 0.25, 0.25]. Therefore, each RNA sequence is a 2D array.

Eternafold data was incorporated in the same manner, except that instead of using the integer 1 to represent the existence of a nucleotide at a certain location, the 1 was replaced with the base-pairing probability of that nucleotide. For example, if a given A has a base-pairing probability of 0.75, it is represented as [0.75,0,0,0]. In this situation, Ns are represented by [0,0,0,0].

Since the data already exists in the scale between 0-1, it does not need to be standardized.

Data from the external datasets is reserved for additional validation after model training. The other sequences, I divided into 90% training, 10% testing sets.

Modeling

I built my convolutional neural networks on a Google Colab notebook since it allows for the use of GPUs to accelerate training. I tested three types of CNN architecture for this purpose: LeNet architecture, VGG architecture, and my own improvised flavor of LeNet architecture. The

initial LeNet architecture performed well, with an overall validation accuracy of 0.83. However, I noticed that the model was overfitting since training accuracy was 0.9952. The VGG architecture did not perform very well, with both training and validation accuracy at 0.53. Therefore, I chose to focus on improving the LeNet architecture and reduce overfitting. To improve accuracy, I added an additional convolutional layer with more nodes and a finer filter and an additional pooling step after that. To reduce overfitting, I added dropout layers between all convolutional and pooling layers. The accuracy of this model was 0.85. However, as shown in Figure 3, it was clear that over the epochs, training and validation accuracy diverged, leading me to believe that the additional epochs were leading to overfitting of the data.

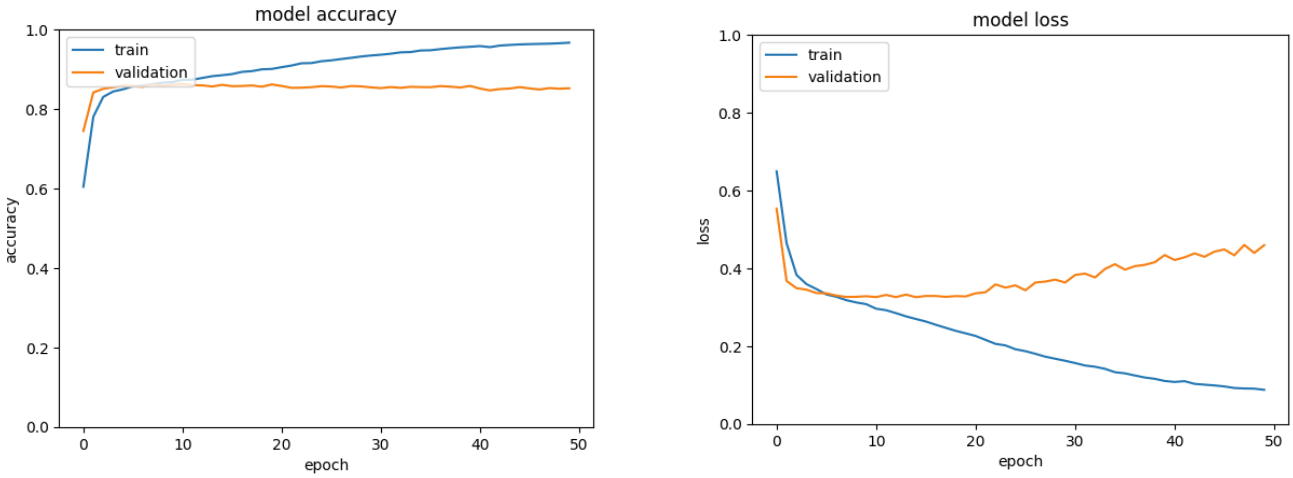

Figure 3: Improved LeNet training and validation accuracy and loss.

For the model, I chose to move forward with my improved LeNet architecture and tune the parameters and hyperparameters. I utilized GridSearchCV to tune the following parameters/hyperparameters: batch size, epoch number, learning rate, optimizer, and dropout size. The final model had the following parameters: batch size = 128, epochs = 10, learning rate = 0.0001, optimizer = Adam, and dropout rate = 0.25.

Finally, I compared the final model's performance on the sequence data and the Eternafold base pairing data (Figure 4). Surprisingly, the model trained on the sequence-only data outperformed the model trained on the Eternafold data.

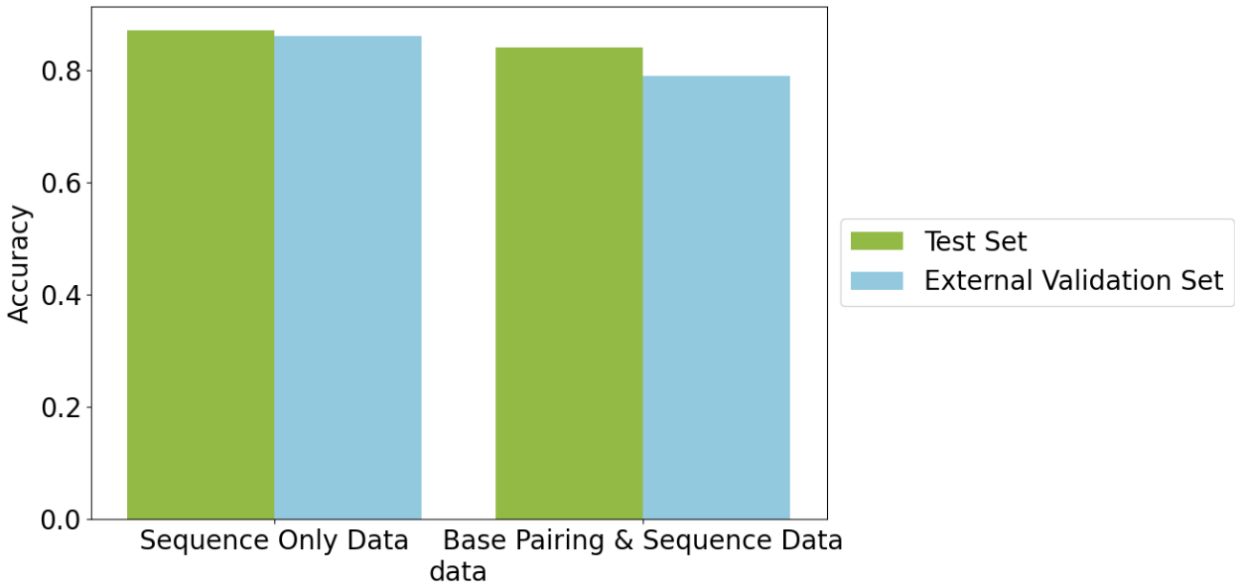

Figure 4: Performance of final model on different data sets. Overall, both were comparable, with the sequence only data performing slightly better on both test sets.

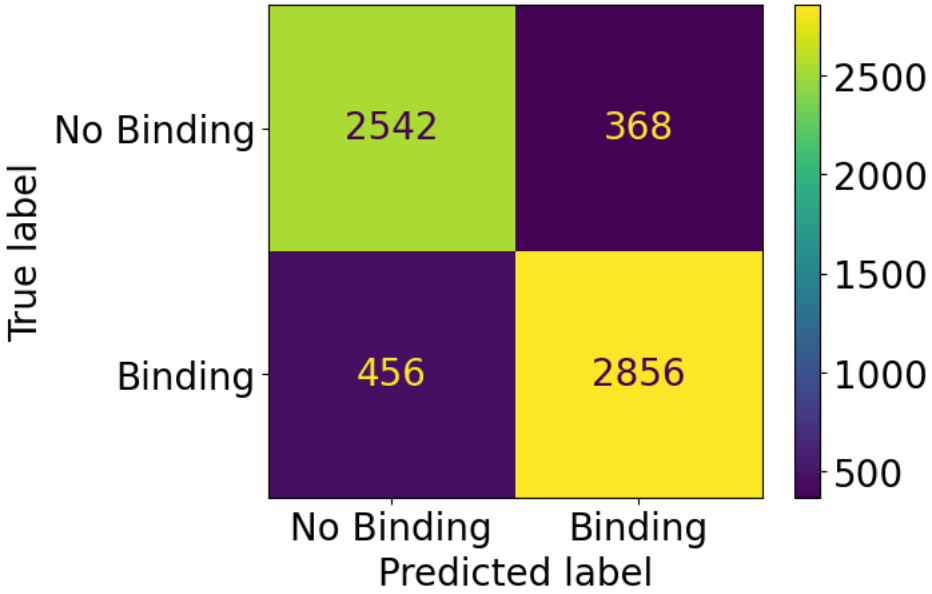

Figure 5: Confusion matrix for best model. Overall accuracy is 0.87. F1-score for both negative and positive classes is 0.86 and 0.87, respectively.

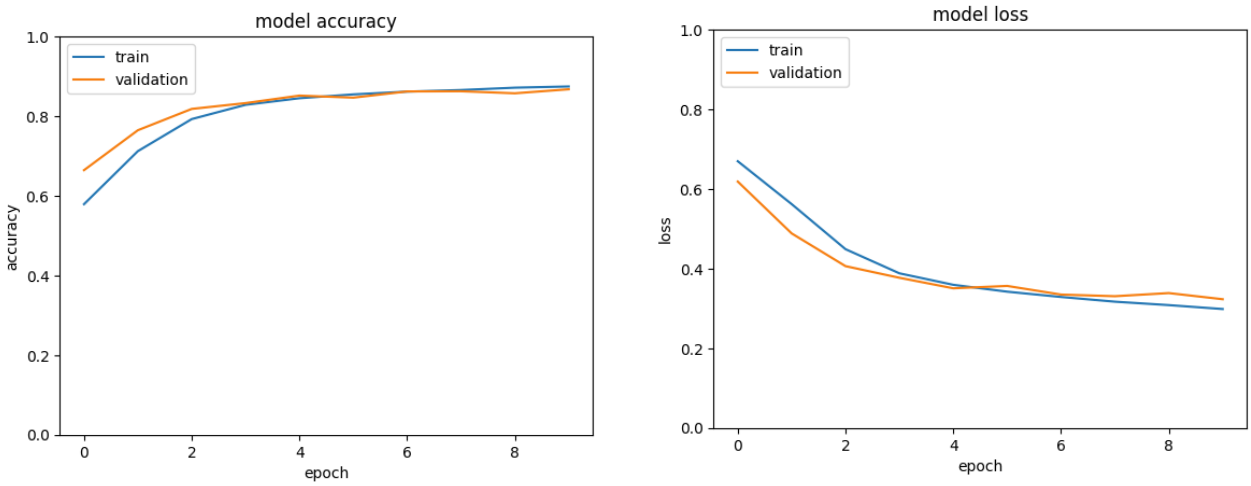

Figure 6: Model accuracy and loss for training and validation sets over epochs of training.

Conclusion & Future Directions

By using large-scale CLIP-seq datasets of RNA sequences with positive and negative examples of binding sites for certain RBPs, as well as secondary structure information derived through running Eternafold on all RNA sequences, I built a CNN that was able to classify RNA sequences into groups depending on whether or not they would bind a given RNA-binding protein (RBP). The final model runs with an overall accuracy of 0.87.

Surprisingly, including the Eternafold data did not improve the prediction task, but actually seemed to worsen it somewhat. Future directions for this work include investigating the possible cause of this, as well as using the other 23 RBP binding datasets to continue to validate the developed model.

---

[i] Wayment-Steele, H.K. *et al.* RNA secondary structure packages evaluated and improved by high-throughput experiments. *Nature Methods,* **19,** 1234-1242 (**2022).**