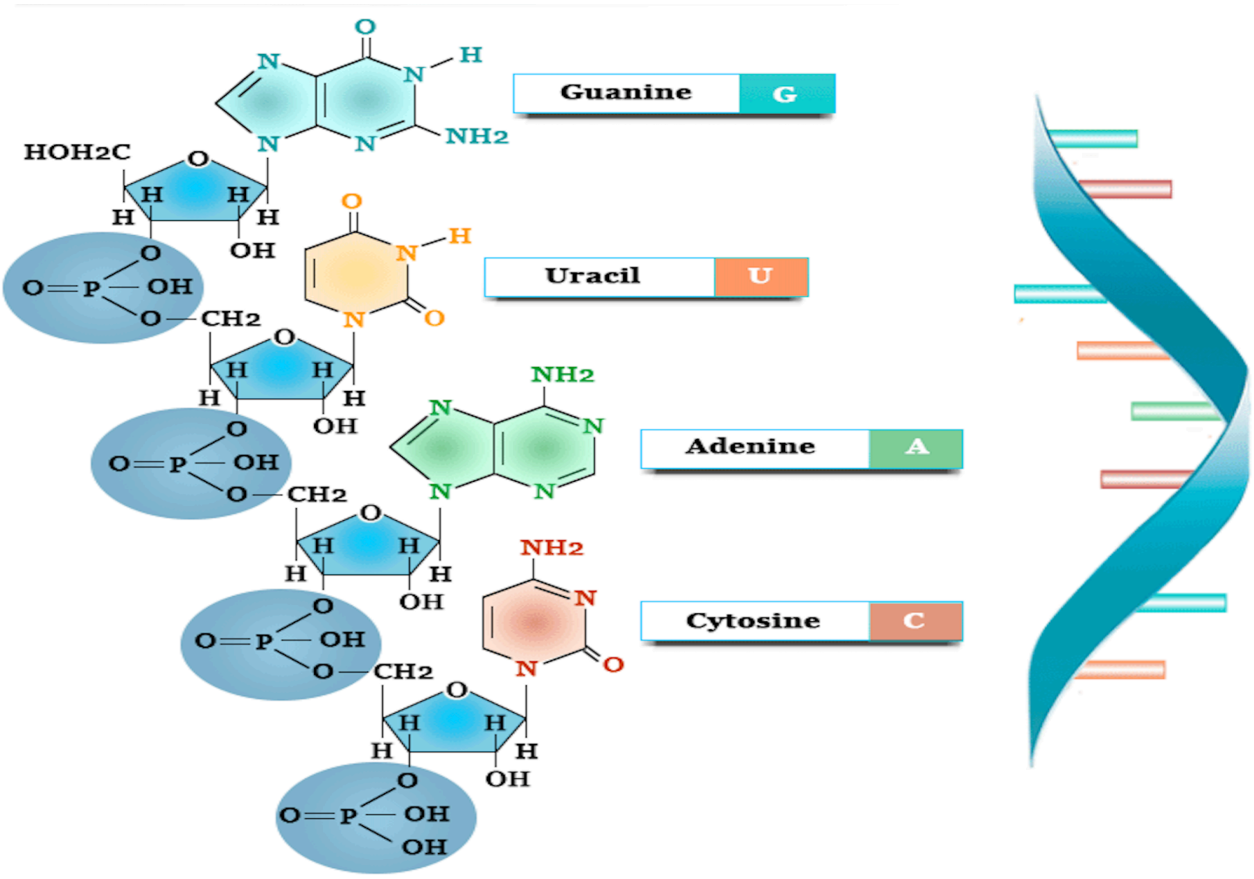
Introduction to Machine Learning - CSC-535B

Beyond the Alphabet: Decoding the Hidden Language of RNA Structure



Submitted By:

Shashank M. Tripathi

Marianela Landi

University Of Illinois at Springfield

Instructor:

Elham K. Buxton

Abstracts:

RNA structure prediction is critical for advancing programmable medicine and treating diseases such as Alzheimer's and pancreatic cancer. The Ribonanza RNA Folding model attempts to solve this problem. Scientists and programmers are working together to find a way around training data and computer resource constraints. To comprehend the architecture of RNA molecules, the solution offers experimental data measuring chemical reactivity in RNA molecules. By using a mutational profiling (MaP) experiment, the model seeks to predict an RNA sequence's responsiveness to the chemical modifiers DMS and 2A3. An effective algorithm has the potential to transform medical research by locating RNA-based drug targets and making it easier to create medications like CRISPR gene therapies and mRNA vaccinations. This endeavor advances our entire understanding of living processes, which goes beyond medicine.

Problem Definition and Goals:

The objective of this project is to predict the structure of any RNA structure and the resultant chemical mapping profile, which can be compared to data collected for each position in the RNA using deep learning algorithms. An implicit understanding of the RNA structure is needed in order to accurately predict the chemical reactivities. The dataset contains information about the experimental measurements of the chemical reactivity at each position of an RNA molecule. The primary objective is to use the available features to predict the reactivities.

Dataset Details:

As part of a Kaggle Research Prediction competition, the Stanford Ribonanza RNA Folding dataset provided for this project includes sensitive data on experimental measurements showing 419 variables, including the chemical reaction of 806,573 distinct RNA sequences of lengths ranging from 115 to 206 in the file “train_data.csv”[1].

The dataset may present difficulties with sequence-specific features as it is intended to predict RNA folding properties. The variables are:

- **Sequence_id**: Unique Arbitrary identifier for each RNA sequence.
- **Sequence**: String describing the RNA sequence with characters A, C, G, and U.
- **Experiment_type**: Indicates the chemical mapping experiment type (DMS_MaP or 2A3_MaP).
- **Dataset_name**: Arbitrary name of the high-throughput sequencing dataset.
- **Reads**: Number of reads in the sequencing experiment assigned to the RNA sequence.
- **Signal_to_noise**: Signal/noise value for the profile, calculated based on measurement values and statistical errors.
- **SN_filter**: Boolean (0 or 1) indicating whether the profile meets specific signal-to-noise and reads criteria.
- **Reactivity_0001, reactivity_0002, ...**: Array of floating-point numbers defining the reactivity profile for the RNA sequence.
- **Reactivity_error_0001, reactivity_error_0002, ...**: Array of floating-point numbers representing errors in experimental values obtained in reactivity.

Related work:

It is very important for this project to understand what Ribonucleic acid (RNA) is and its structure to be able to create an effective model. RNA is an important molecule that builds and maintains the cells in the human body. This is done by taking genetic information from the deoxyribonucleic acid (DNA) to the parts of your cells that make proteins, which is what keeps the human body functioning. In other words, RNA transfers genetic information from the DNA to different cells to keep the body working properly. The primary structure of an RNA molecule is a linear sequence of nucleotides,

which consist of sugar, which is called ribose that forms the backbone of the RNA strand; phosphate group, which is links the ribose sugars together to form the sugar-phosphate backbone of the RNA molecule; and nucleotide base, which is the third component of a nucleotide is a nitrogenous base. The four bases in RNA are Adenine (A), Cyosine (C), Guanine (G) and Uracil (U). The secondary structure of an RNA is formed when molecules produce various structural elements based on the interactions between its constituent nucleotides including the complementary base pairing between the four bases, which means that the bases are bounded [2]. In this project, we are given a dataset of RNA sequences and their corresponding secondary structures, there are 1,118,513 RNA sequences of lengths between 115 and 206. The knot theory is important to understand for this project, it is a branch of mathematics about closed curves in three-dimensional space. Knots can be classified by number of crossings, symmetry or knot invariants. The RNA molecules are typically single-stranded, but they can also form secondary structures by base-pairing between complementary nucleotides. They can form open or closed knots [3].

Data Exploration and Preprocessing:

This Ribonanza RNA Folding dataset contains a sequence column, which contains sequences with specifications of the experiment type and an additional column (same sequence with different reactivity values). This diversity in sequence length presents an interesting aspect of the dataset. While 177 is the most common length, there is a range of sequence lengths present, with a minimum of 115 nucleotides and a maximum of 206 nucleotides as shown below in Fig 1.1. This diversity in length adds complexity to the data analysis, as different algorithms may be better suited for sequences of different lengths.

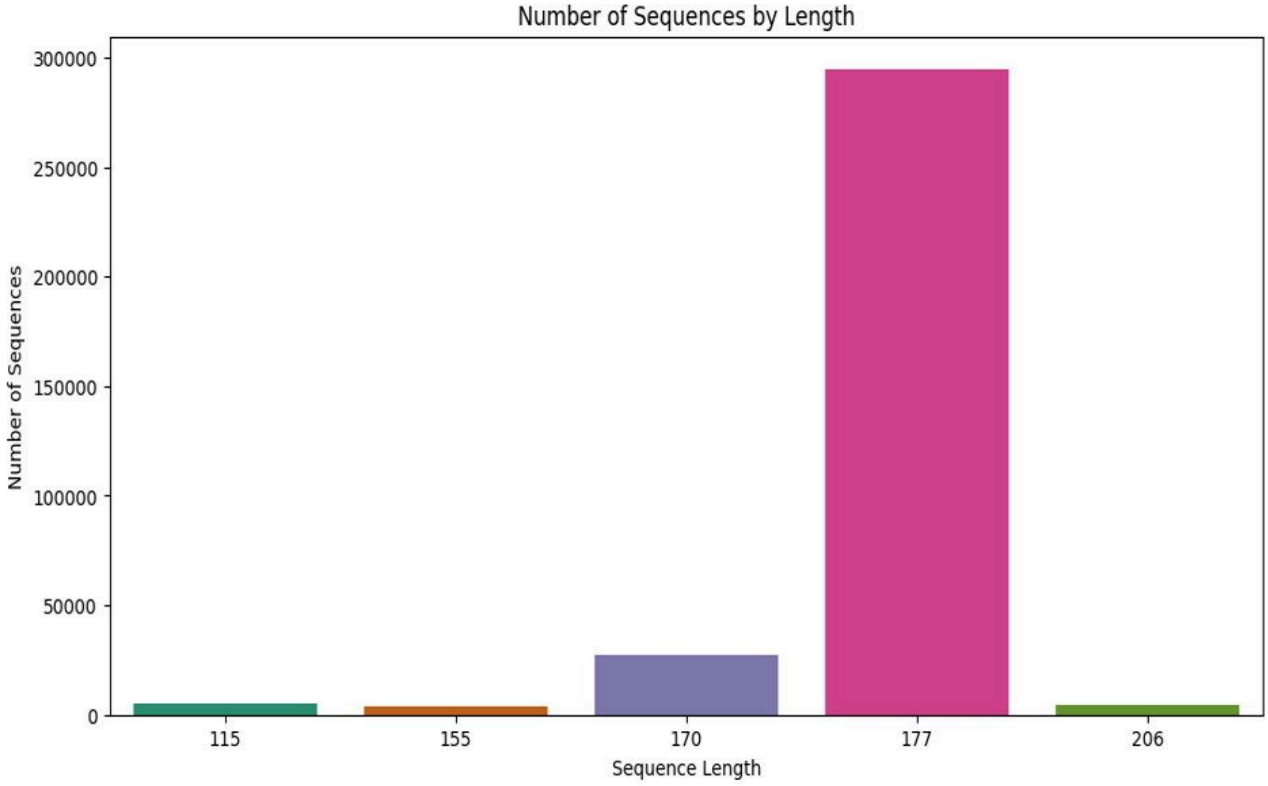


Fig 1.1

The Ribonanza RNA Folding dataset reveals fascinating insights into the variability of reactivities across different experiment types and sequences. This diversity suggests that the reactivity profile of an RNA molecule is not solely determined by its sequence but is also influenced by specific experimental conditions. The graphs clearly illustrate the distinct reactivity distributions observed for different experiment types. Each experiment type exhibits a unique pattern, highlighting the influence of experimental conditions on the reactivity landscape. The distribution of reactivities for both experiment types varies for different sequences. Even for the same sequence, it's different for different reactivity as shown in the graphs Fig 1.2 and Fig 1.3.

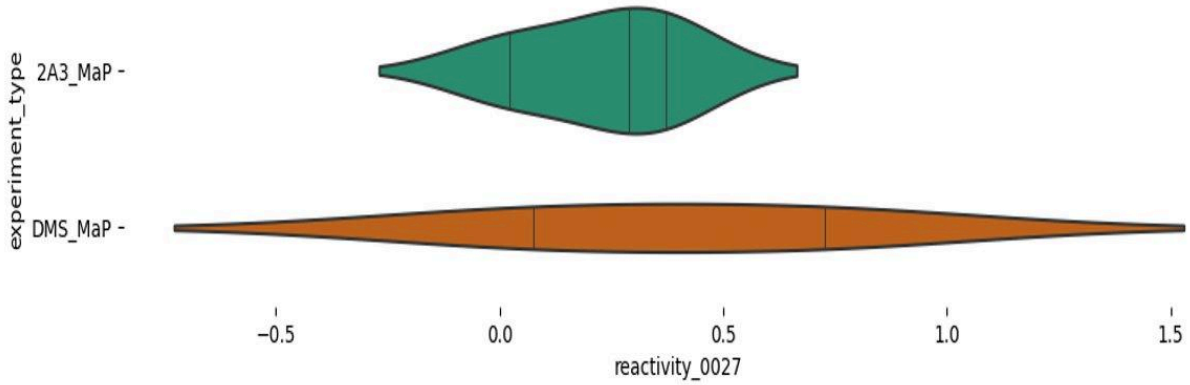


Fig 1.2

These graphs show that the range of values for experiment types for different reactivity.

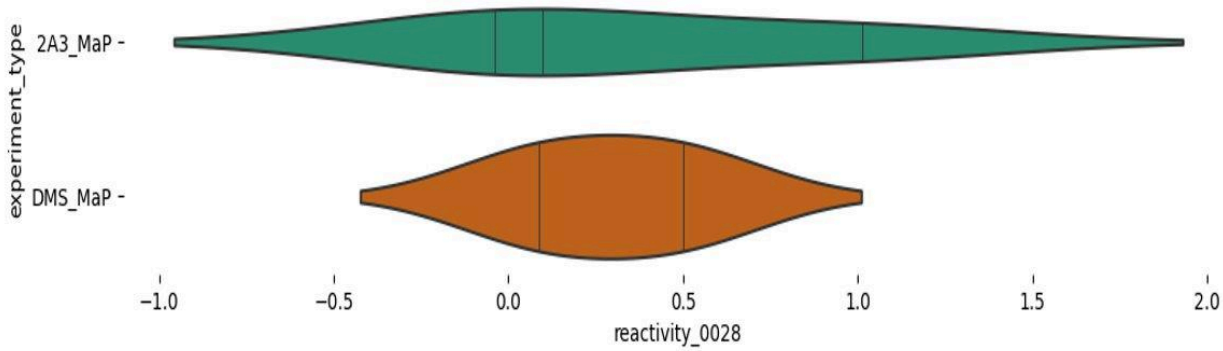


Fig 1.3

Our extensive analysis of the Ribonanza dataset, utilizing statistical tools and graphical visualization, revealed the presence of missing values (NaNs). To address this issue, we opted to replace these Nans with zeros, ensuring data integrity and facilitating subsequent processing steps.

Next, we focused on preparing the sequence data for training. We employed the Keras Tokenizer library to convert the sequences composed of A, C, G, and U characters into numerical representations. This transformation mapped each character to an integer between 1 and 4, significantly simplifying the data format for model training.

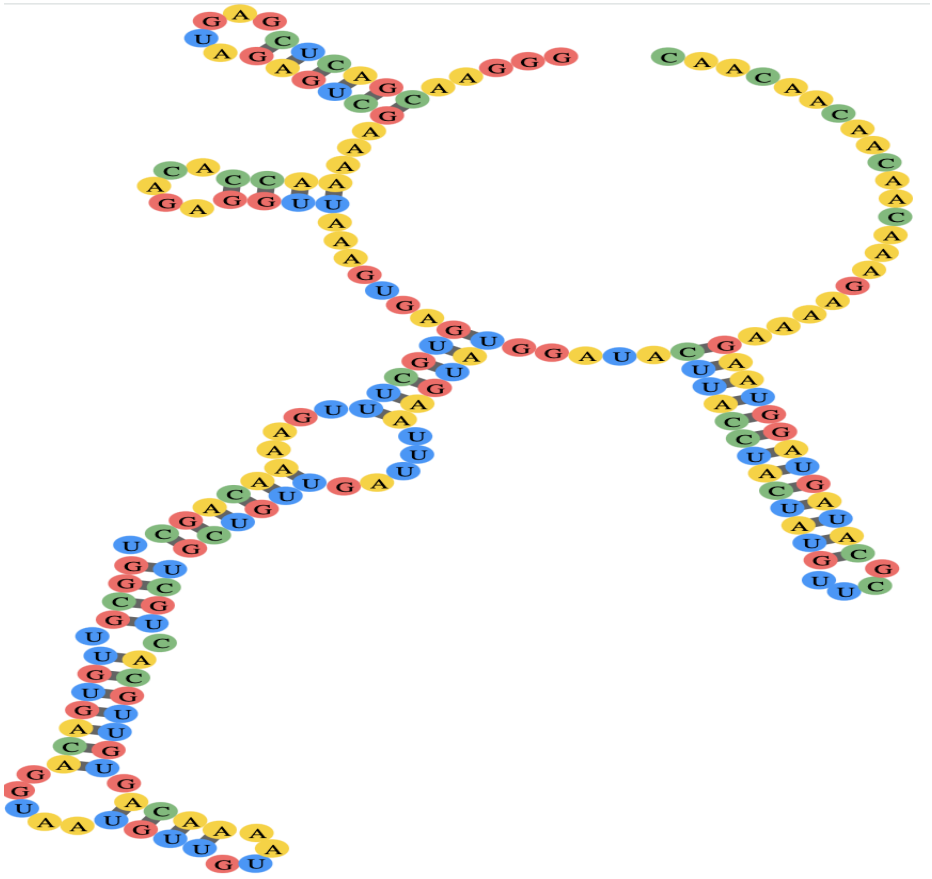


Fig 1.4

However, upon tokenization, we observed inconsistencies in the sequence lengths. To address this, we implemented the **`pad_sequences`** function. This function intelligently padded each sequence with zeros to achieve a uniform length of 206, the maximum sequence length found in the dataset. This standardization ensured equal representation for all sequences during model training.

Finally, we employed the **`train_test_split`** library to partition the preprocessed data into distinct training, validation, and test sets. This division enables us to effectively evaluate the performance of our trained models and assess their generalizability to unseen data.

Data Analysis:

After preprocessing and splitting our dataset, as explained in the previous section, we proceed by creating the model using the first layer as the embedding layer to convert the tokenized and padded input sequences into **dense vectors** of fixed size, with input dimension of 5, 206 as length of it, output of 256 and masked the zeroes so the model ignores the reactivities that are equal to zero. The second layer is the bidirectional LSTM layer with 128 neurons, which is very important to be able to capture the dependency in the data from past and future values, including the return_sequence=True to make sure that the output of the layer is the full sequence of outputs for each input sequence. The third layer is the time-distributed layer with 64 neurons and 'relu' activation, which is important to apply the activation to each time step independently, ensuring that the temporal dependencies are preserved.

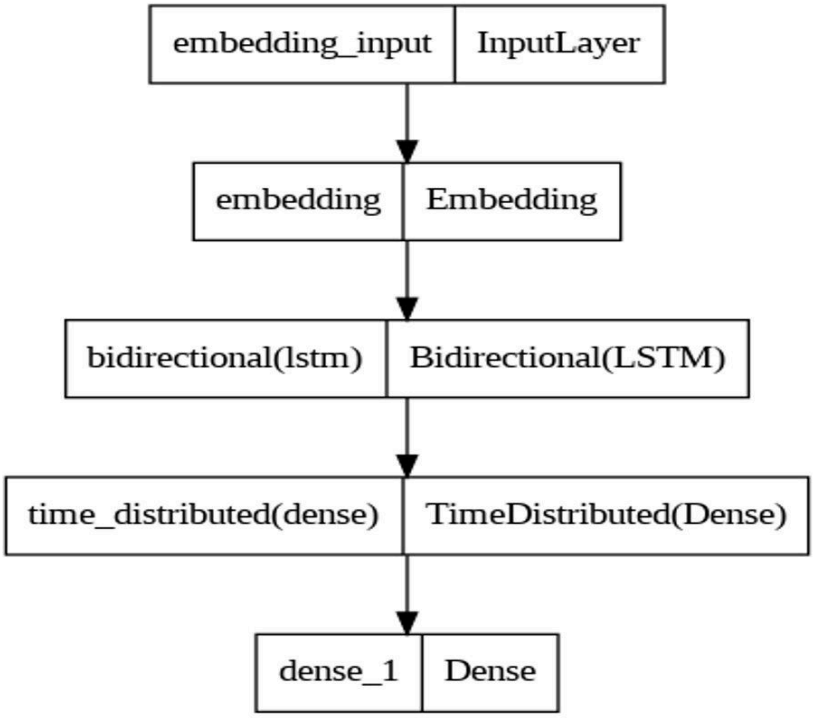


Fig 2.1

The last layer is the dense output layer with no activation, since it is a regression model where continuous values are predicted and they are not being restricted to only non-negative numbers. The summary of our model is shown in figure 2.1, where we can observe each layer applied. Also, in figure 2.2, we can see the layers of the model, the shape of the outputs and the parameters of each one of them.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 206, 256)          1280

 bidirectional (Bidirection  (None, 206, 256)          394240
 al)

 time_distributed (TimeDist  (None, 206, 64)           16448
 ributed)

 dense_1 (Dense)             (None, 206, 1)            65

=================================================================
Total params: 412033 (1.57 MB)
Trainable params: 412033 (1.57 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
```

Fig 2.2

The absolute mean error loss function is applied, since it is mentioned in the kaggle competition that submissions are scored using this function, and it allows the model to learn the best parameters by minimizing the squared differences between predicted and

true values. The individual losses are aggregated to compute the final loss automatically (reduction), depending on the context. The optimizer used is Adamw with 1e-2 as the learning rate, which is similar to the Adam optimizer that is effective for most deep learning models, but Adamw handles the weight decay in a better way, getting a better generalization of the model on the validation and test sets. After applying all of the above, we fit the model using the train and validation data, and plot the validation and train losses in a graph, which is part of the evaluation process. The results expected from the model are the values of the reaction for each position inside each sequence.

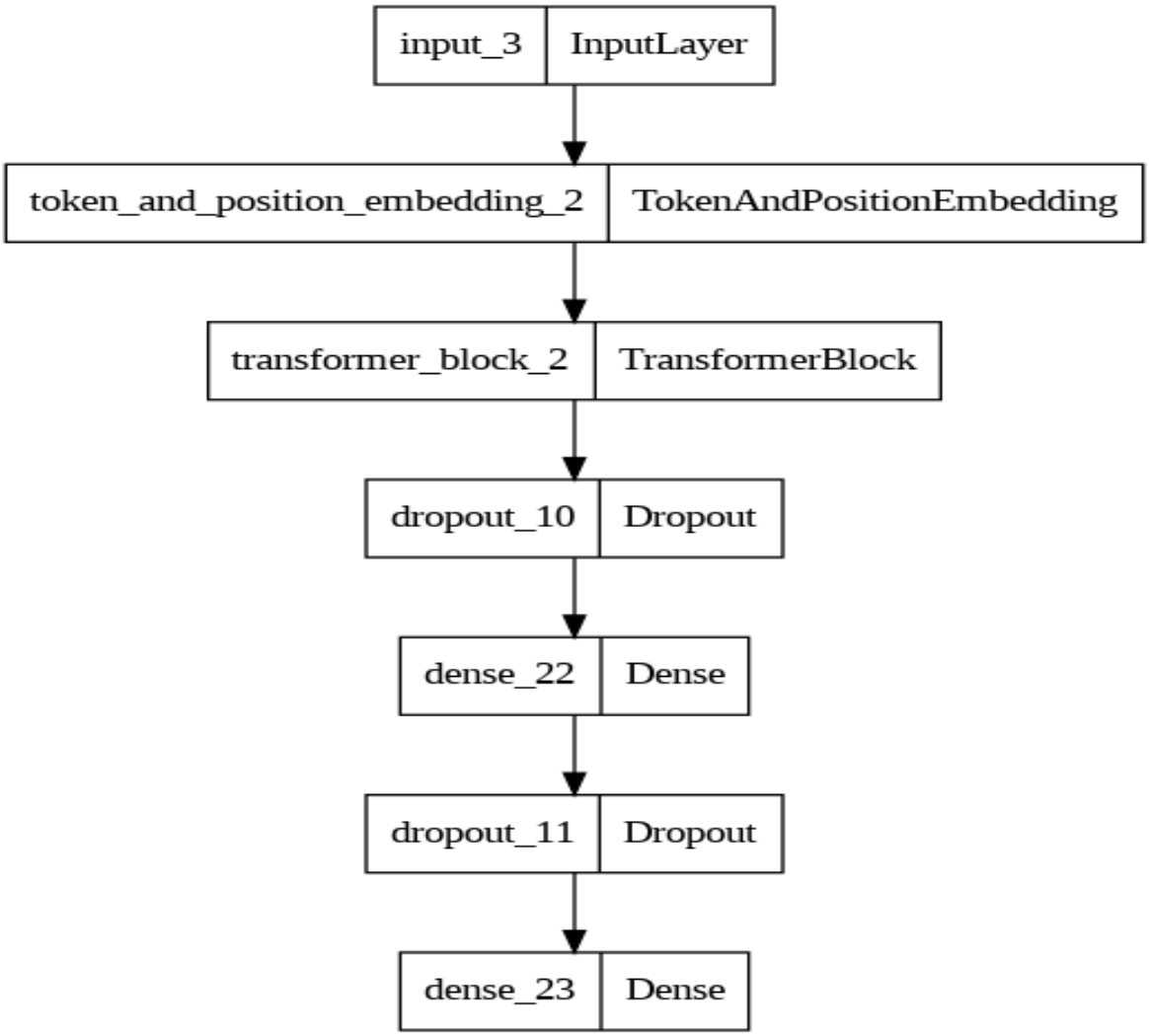
Expanding our investigation beyond Recurrent Neural Networks (RNNs), we explored the potential of transformer-based architectures for sequence analysis tasks. As illustrated by the provided block diagram, the model leverages a transformer architecture specifically tailored for sequence-to-sequence tasks, likely regression due to the absence of an activation function in the output layer.

The model begins by receiving input sequences of a defined maximum length. These sequences are then processed through a specialized embedding layer, TokenAndPositionEmbedding, which combines token and positional information into a unified representation. This enriched representation serves as the foundation for the subsequent analysis.

The core of the model lies in a powerful transformer block, TransformerBlock. This block employs multi-head self-attention and feedforward layers to capture complex contextual relationships and patterns within the sequence. Dropout layers are strategically placed throughout the architecture to prevent overfitting and enhance the model's ability to generalize to unseen data.

Following the transformer block, a dense layer equipped with a ReLU activation function further extracts intricate patterns from the transformed representation. This is followed by another dropout layer to ensure model stability. Finally, the architecture culminates in a single-unit output layer, well-suited for regression tasks.

The choice of a transformer architecture highlights the model's focus on capturing long-range dependencies within sequences. Additionally, the inclusion of dropout layers demonstrates a commitment to preventing overfitting and enhancinggeneralizability. This combination of architectural features makes the model a compelling option for a wide range of sequence-based applications.



```
Model: "model_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_3 (InputLayer)        [(None, 206)]             0

 token_and_position_embeddi  (None, 206, 256)          54016
 ng_2 (TokenAndPositionEmbedding)

 transformer_block_2         (None, 206, 256)          2236160
 (TransformerBlock)

 dropout_10 (Dropout)        (None, 206, 256)          0

 dense_22 (Dense)            (None, 206, 256)          65792

 dropout_11 (Dropout)        (None, 206, 256)          0

 dense_23 (Dense)            (None, 206, 1)            257

=================================================================
Total params: 2356225 (8.99 MB)
Trainable params: 2356225 (8.99 MB)
Non-trainable params: 0 (0.00 Byte)
```

The above block provides a comprehensive overview of the model's architecture, detailing each layer with its respective shape and parameter count. This information offers valuable insights into the model's complexity and resource requirements.

In the next section of the document, we will delve deeper into the model's performance and explore potential optimization strategies through parameter tuning. This analysis will provide further insights into the model's effectiveness and uncover opportunities for improvement.

Evaluation, Tuning and Improving Model:

For this research project, we are only looking at validation and training losses and not accuracy because it is a regression problem. Thus, the predictions are not being compared with true class labels, so it does not provide meaningful information since there is no concept of "correct" or "incorrect" prediction.

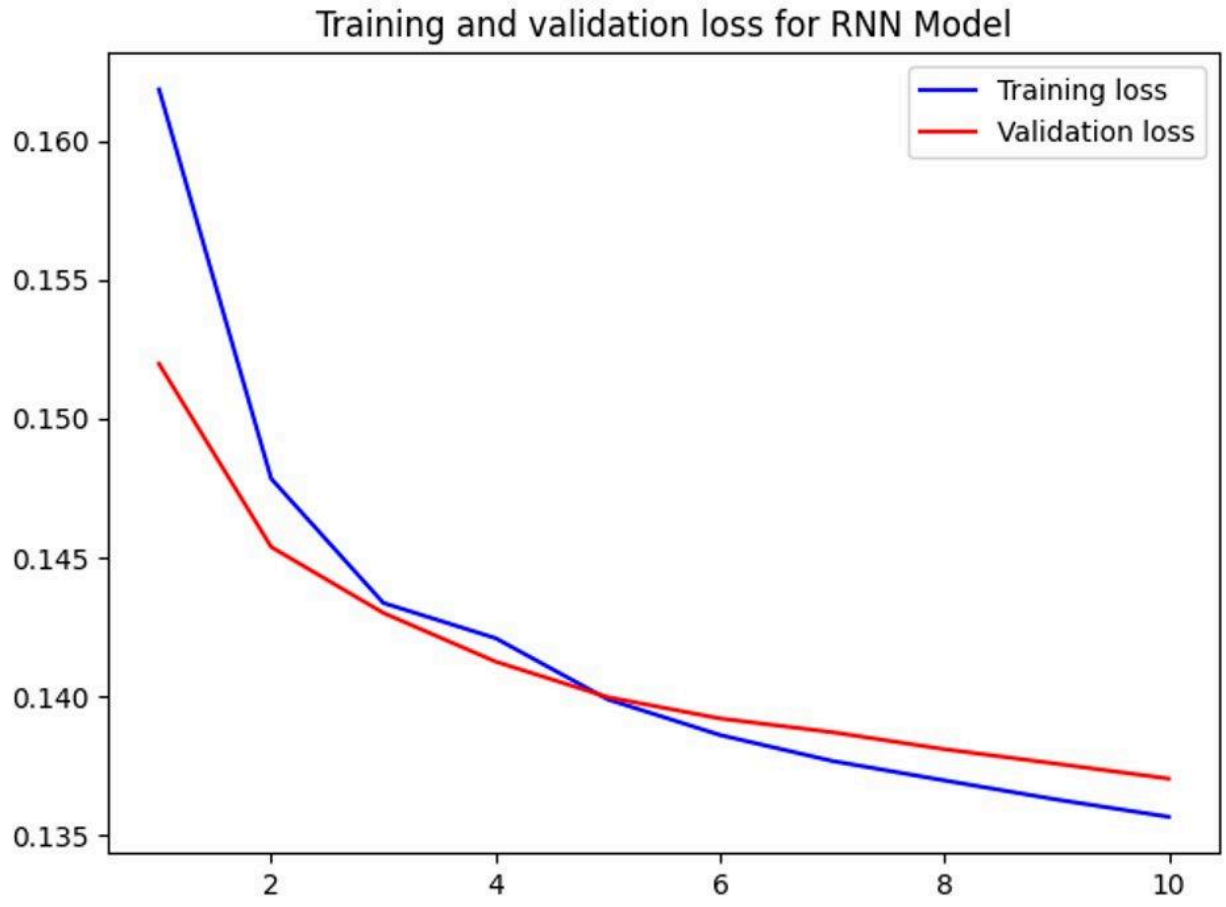


Fig 3.1

From figure 3.1, we can see that our model is overfitting and reducing the validation and training loss after every epoch. Also, the mean absolute error is 11.84%.

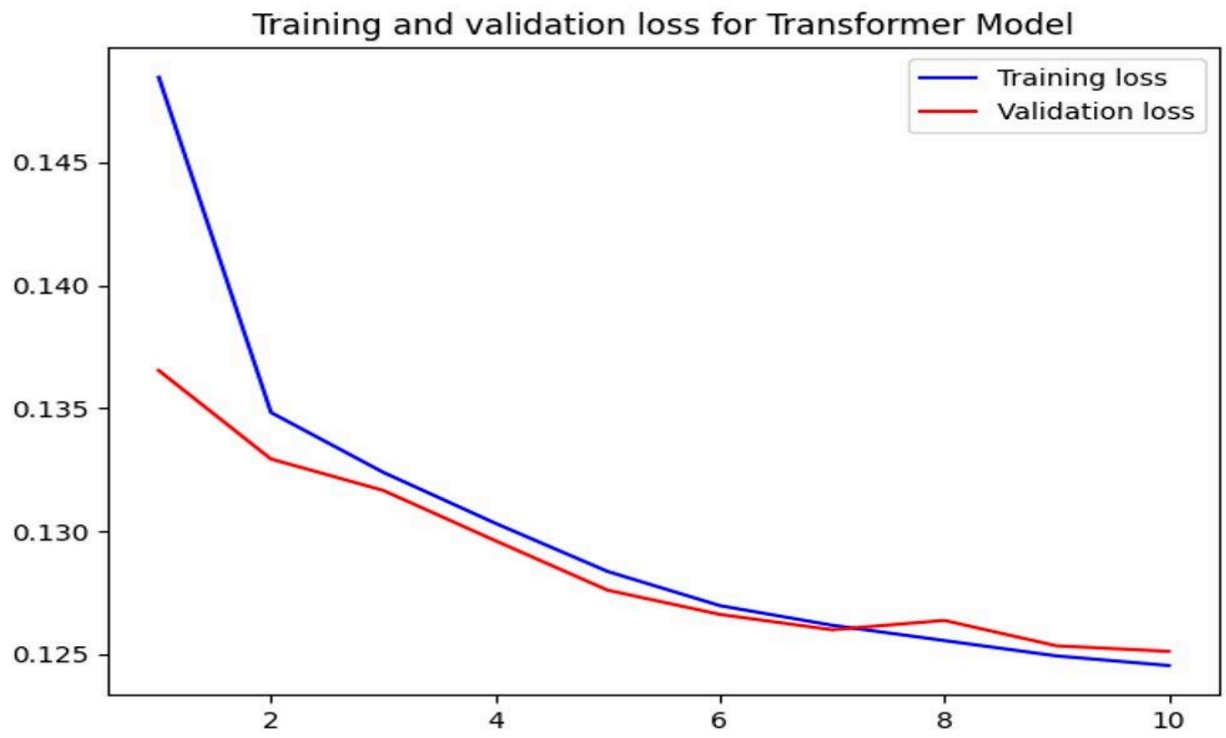


Fig 3.2

To improve the model, a transformer block was applied, with a dropout rate of 0.1, with an input layer, and an embedding layer, after that we applied the transformer block. Then, a dropout layer is followed by a dense and dropout layer again. Finally, the output layer. After the evaluation of the model, we can observe from figure 3.2 that the model is learning since the loss is reduced after each epoch. We also compared both models, and we can observe that the RNN model is showing better performance, since the mean average estimate is higher for the transformer model, showing 12.50% against 11.84%. Thus, we take into consideration the predictions from the RNN model.

Conclusion:

After creating the models, training and evaluating the dataset, we can conclude that we learned how to create a model that predicts values for a very complex dataset, in which we are dealing with inconsistent data. We learned how to clean and preprocess this data. Also, to make this project simpler, we filtered the dataset to have only one experiment type instead of two, which makes the dataset half the original size. For future work, we will preprocess and clean out data differently, clipping out values between 0 and 1 before calculating MAE and then we could apply sigmoid for our output later to get the reactivities between 0 and 1. Also, we will use the whole dataset, and get the desired results we want for the transformer model, which was supposed to have better predictions.

Reference:

[1]hhttps://www.kaggle.com/competitions/stanford-ribonanza-rna-folding/data?select=train_data.csv

[2]https://www.kaggle.com/code/rafiko1/standford-ribonanza-rna-theory-short-eda

[3]

https://www.kaggle.com/code/ayushs9020/understanding-the-competition-standford-ribonaza