CS7140 Final Project
December 12, 2020

Rajesh Sakhamuru
Zhiyu Chen

COVID-19 mRNA Vaccine Degradation Prediction

Abstract

In the endeavor to produce a vaccine for COVID-19 during this global pandemic, it is necessary to be able to make predictions about mRNA degradation. Given data from the Stanford University OpenVaccine Kaggle competition, we developed CNN and LSTM model architectures and compared them in order to determine which model style should be pursued for predicting mRNA molecule degradation. We made additional models which were modified versions of our base models and compared them with each other as well. Due to the nature of the data which we were given, we found that the LSTM model performed significantly better than the CNN model due to the sequential nature of the relationships which can be seen in the data. Our model performed reasonably well for its level of complexity when compared to the official final competition scores which can be seen on the Kaggle competition website.

Background

One of the biggest challenges facing the scientific community in developing a vaccine for COVID-19 is mRNA degradation. Without being able to accurately predict how an mRNA sequence will degrade without an experimental trial, it can greatly delay a COVID-19 vaccine development which could potentially harm hundreds of thousands of lives. Currently any vaccines in development must be prepared and shipped with strict refrigeration guidelines and will be inaccessible to millions of people unless the vaccines can be produced with mRNA sequences as stable as possible.

By developing deep learning models which are capable of predicting degradation rates at each point in an mRNA sequence, we can prevent the waste of time and resources on the creation of unstable vaccine designs and instead create computer models before running experiments. Our model will predict likely degradation rates at each base of an RNA molecule, trained on the Eterna dataset provided by Stanford University which is comprised of over 3000 RNA molecules and their degradation rates at each position for training and testing deep learning models.

Data

The data source we are using is: OpenVaccine: COVID-19 mRNA Vaccine Degradation Prediction**.** We have been provided structured data by Stanford University via the Kaggle competition with each data sample consisting of data features of an RNA sequence along with degradation predictions in various conditions at each nucleotide base position.

Training (and Validation) Data:
Input data:

sequence	structure	predicted_ loop_type	signal_to _noise	SN_filter	seq length	seq_scored
string	string	string	float	int	int	int
107	107	107	>0.0	0 or 1	107	68

For example:

```
"sequence":"GGAAAAGCUCUAAUAACAGGAGACUAGGACUACGUAUUUCUAGGUAACUGGAAUAACCCAUACCAGCAGUUAGA
GUUCGCUCUAACAAAAGAAACAACAACAACAAC"
"structure":".....((((((.......)))).)).((.....((..((((((....))))))..)).....))....(((((
((....)))))))....................."
"predicted_loop_type":"EEEEESSSSSSHHHHHHHSSSSBSSXSSIIIIISSIISSSSSSHHHHSSSSSSIISSIIIIIS
SXXXXSSSSSSSHHHHSSSSSSSEEEEEEEEEEEEEEEEEEEEE"
```

Output data:

reactivity	deg_Mg_pH10	deg_pH10	deg_Mg_50C	deg_50C
float list	float list	float list	float list	float list
length: 68	length: 68	length: 68	length: 68	length: 68

- reactivity: reactivity values for the first 68 bases in the sequence, used to determine the likely secondary structure of the RNA sample.
- deg_PH10: the likelihood of degradation at the base after incubating without magnesium at high pH (pH 10) for the first 68 bases.
- deg_Mg_pH10: the likelihood of degradation at the base after incubating with magnesium in high pH (pH 10) for the first 68 bases.
- deg_50C: the likelihood of degradation at the base after incubating without magnesium at high temperature for the first 68 bases.

- deg_Mg_50C: the likelihood of degradation at the base after incubating with magnesium at high temperature for the first 68 bases.

We can see from the above description, all the training data only gives the prediction of the first 68 bases. The prediction after 68th mRNA bases is not available in the training dataset.

Testing Data:

sequence	structure	predicted_ loop_type	signal_to _noise	SN_filter	seq length	seq_scored
string	string	string	float	int	int	int
107 or 130	107 or 130	107 or 130	>0.0	0 or 1	107 or 130	**107 or 130**

As we can see, the RNA sequence, structure and loop_type are all sequential data, and we are predicting sequential data in multiple categories for each position in the sequence. We need to predict reactivity, deg_Mg_pH10, deg_pH10, deg_Mg_50C, deg_50C at each base of the RNA molecule. The structure of the test dataset shows in the above table.

The ‘signal_to_noise’ value and the ‘SN_filter’ value are informative of which data is best for training. High signal ratio means that it is less noisy. SN_filter, rather than providing a float value, is boolean where all ‘1’ values are the ideal set for training with the best signal to noise ratio.

The major difference between the training dataset and the test dataset is that, the test dataset has different sequence length, 107 or 130, but the training dataset only has sequence length of 107. Also, the training dataset only has the performance of the first 68 bases, but the test dataset requests to predict the overall 107 or 130 bases performance on the sequence.

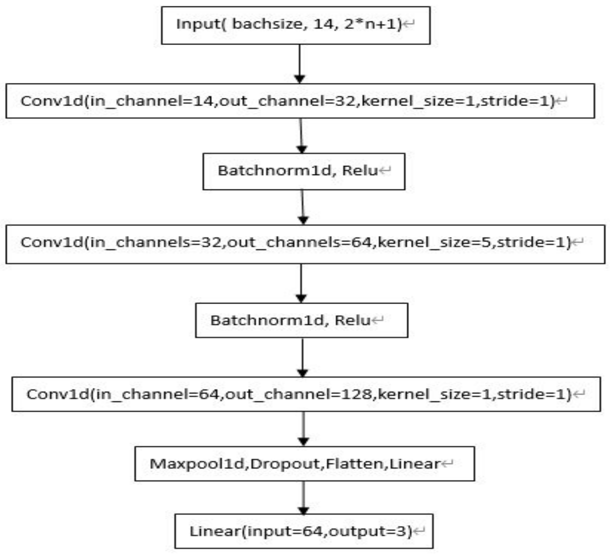
Additionally for actual scoring by the Kaggle competition, only the following: reactivity, deg_Mg_pH10, and deg_Mg_50C predicted values are actually scored, so for the CNN model which we developed we only made predictions for these three columns.

Models

The models which we tried are the following:

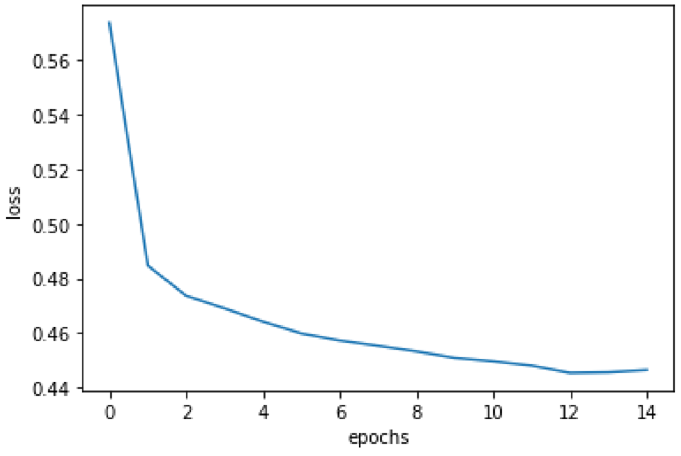
<u>CNN</u>

At the first step, we have to preprocess the sequence data. We suppose that the performance of the current mRNA base is dependent on the previous n mRNA base, n mRNA base afterwards and the base itself. Then, we did one-hot encoding for each base. After preprocessing, each example has the shape (2n+1,one-hot length=14).

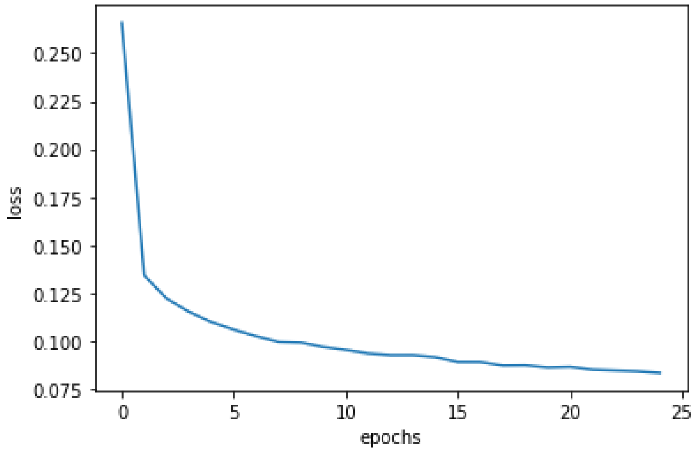

Input(bachsize, 14, 2*n+1)
Conv1d(in_channel=14,out_channel=32,kernel_size=1,stride=1)
Batchnorm1d, Relu
Conv1d(in_channels=32,out_channels=64,kernel_size=5,stride=1)
Batchnorm1d, Relu
Conv1d(in_channel=64,out_channel=128,kernel_size=1,stride=1)
Maxpool1d,Dropout,Flatten,Linear
Linear(input=64,output=3)

CNN Model shows as the figure above. We used three conv1d layers, maxpool layers, batchnorm1d layers, flatten layer, and finally linear layer to output the result.For such a complicated problem, our CNN model produces valid and decent results.

Base CNN Model:



CNN with denoising and embedding:

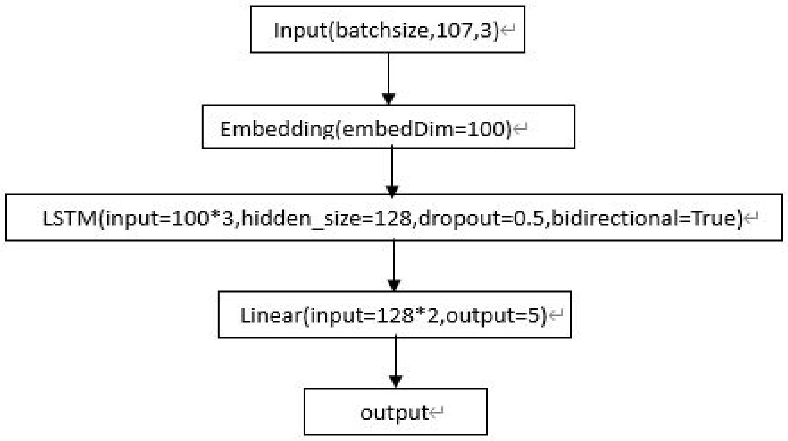


To improve the result, we also clean up the data and use embeddings to let the model learn each symbol of mRNA base a feature vector, rather than use one-hot encoding directly. From the result, we can see there is slight improvement, but it is not very significant. To sum up, CNN model is not the best solution for this type of sequence data prediction.

LSTM

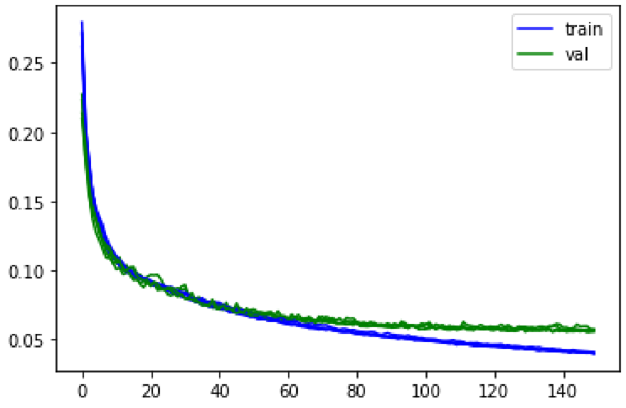
Similar preprocessing in CNN model, but we add an embedding layer before LSTM model. In the original data, each mRNA base has three symbols, one from sequence, one from structure, and one from predicted_loop_type. The shape of the data is (107,3). After embedding, the feature shape becomes (107,3*embedding_dim).

Next, we feed the output of the embedding layer to a bidirectional multi-layer LSTM model. Finally, we use a Linear layer to output the result. The overall model shows the following.

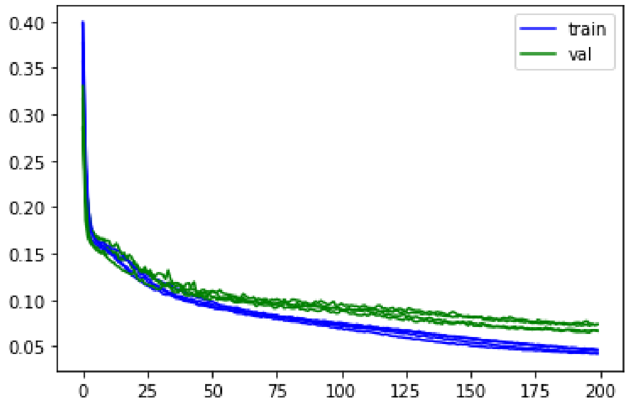

Input(batchsize,107,3)
Embedding(embedDim=100)
LSTM(input=100*3,hidden_size=128,dropout=0.5,bidirectional=True)
Linear(input=128*2,output=5)
output

One problem of LSTM is that the model cannot capture long-range dependencies of every mRNA base to predict current RNA base performance prediction. There is no way to give more importance to some of the mRNA base compared to others. To solve the problem, we add an attention layer before prediction to capture weights from different mRNA bases. However, it shows that the score is even worse. After analysis, the reason behind is, we believe, the training examples only predict the first 68 mRNA bases performance, but the test set requests to predict the first 91 mRNA bases performance. There is a chance that the weights to predict the mRNA performance after 68 are never updated. We have to do further study on this part in the future.

Base LSTM model:



LSTM with Attention:



Because we cannot just compare the loss function between LSTM and CNN models for an accurate prediction of how well each model did compared to the other, we verified our model results via late-submissions to the Kaggle competition for model scoring. It handily confirmed our expectations that the base LSTM model performed the best among the techniques we tried.

Kaggle

One of the benefits of using a dataset from a Kaggle competition is that we were able to make late-submissions to the competition and see how well our models stacked up against the best performing submissions to the competition. The models in the Kaggle competition are scored based on mean column-wise root mean squared error (MCRMSE) of their predictions.

$$\text{MCRMSE} = \frac{1}{N_t}\sum_{j=1}^{N_t}\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_{ij}-\hat{y}_{ij})^2}$$

Additionally, only the following: reactivity, deg_Mg_pH10, and deg_Mg_50C predicted values are actually scored for the competition rankings.

The following table summarizes the ranking of our experiment compared to models made to the actual competition:

model	**Public score**	**Private score**	**Ranking**
CNN	0.32104	0.42497	1487
CNN_EMB_CLEARN	0.32056	0.41448	1487

LSTM	0.27012	0.38448	1236
LSTM_ATT	0.29657	0.38316	1422

As we can see here, the base LSTM model performed the best out of the 4 models which we created. The best public score was 0.228 and the best private score was 0.342. Although this seems like it is not that far from our scores achieved by our simple models, in order for the better scores to be achieved and drive down the MCRMSE loss, many competition models actually ended up using 5 to 10 models in tandem (1st place used 50+) either via blending or ensembling techniques and also incorporated extensive feature engineering. Many of the feature engineering techniques used required a level of domain knowledge beyond what we had.

Summary

Our deep learning models enabled us to make reasonable accurate predictions about mRNA degradation at each nucleotide base in a provided sequence. By training a CNN and an LSTM model we were able to confirm our intuition that a sequential model rather than a convolutional one would be able to make significantly better predictions given this type of data. We were able to see where we could find room for improvement and even where potential improvements hindered rather than helped with prediction.

From here, these models could be further improved and in the footsteps of other highly scoring models built by teams of professionals, we could develop more models and via ensemble techniques and extensive feature engineering they could prove incredibly useful for future vaccine development, not just in the case of a COVID vaccine.

Biologists could use these models to pick the RNA sequence whose performance fit their requirement well. Hopefully, our effort could give the right guidance to the medical community to develop COVID-19 vaccine as soon as possible.

Improving mRNA stability is crucial to creating a COVID vaccine. Being able to predict this kind of thing using a deep learning model could save money and time and once a model is generated it could be reverse-engineered to determine what sorts of features mRNA molecule sequences have which could lead to more or less stability.

Although our performance is not ideal, we experimented with the basic models in advanced Machine Learning class. The project is very interesting. Meanwhile, we found the difficulties of using deep learning frameworks to solve realistic problems. We have to study more in the field to become proficient in developing models for real-world data.