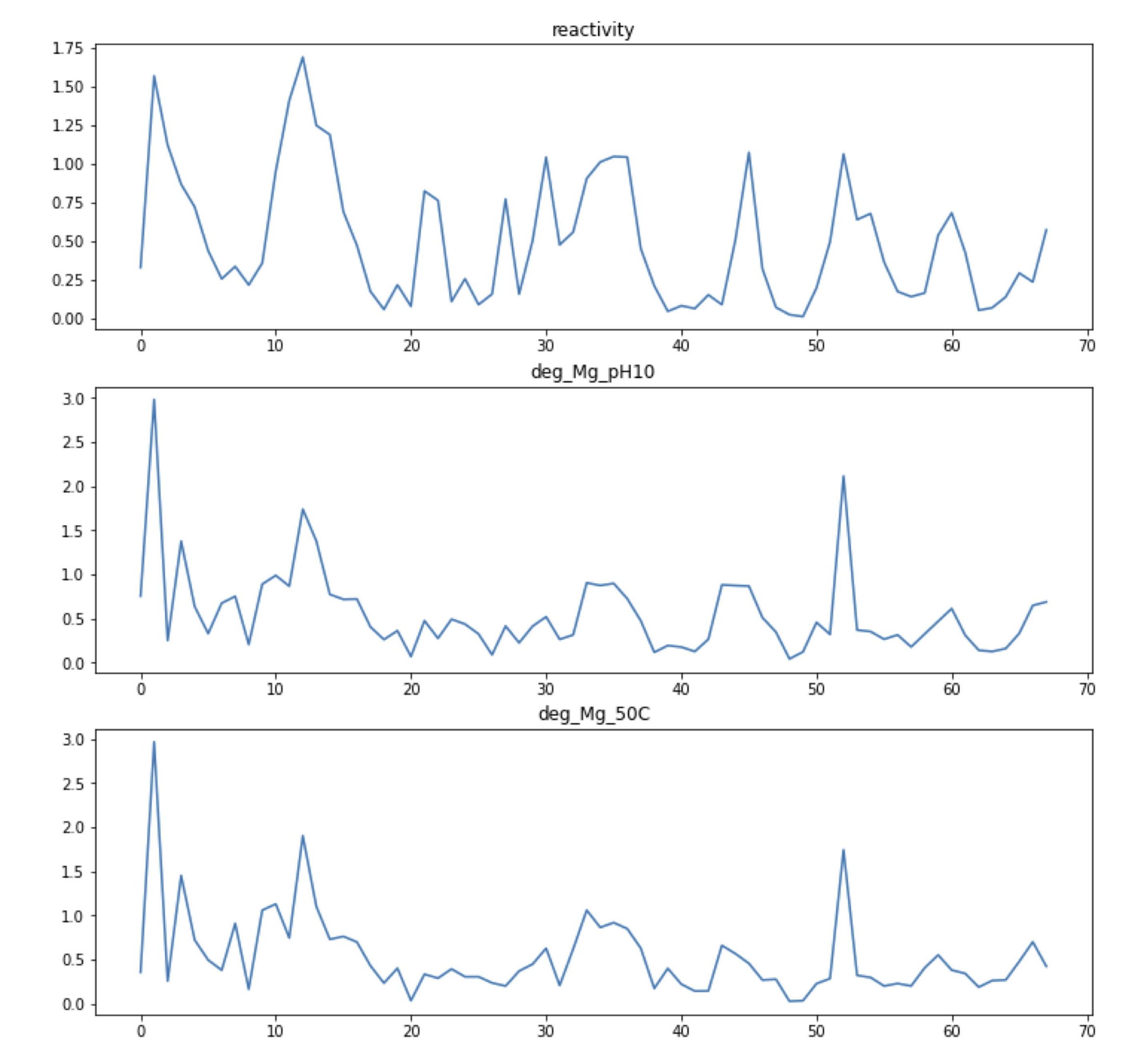
Problem statement

The goal of this competition was to predict, for each of the bases (A, C, G or U) in the provided RNA sequences, the reactivity and degradation under certain circumstances such as incubation with or without magnesium in high temperatures or high pH. In summary, we have to predict three continuous values (i.e. regression) for each of the bases in the sequence (i.e. sequence-to-sequence).

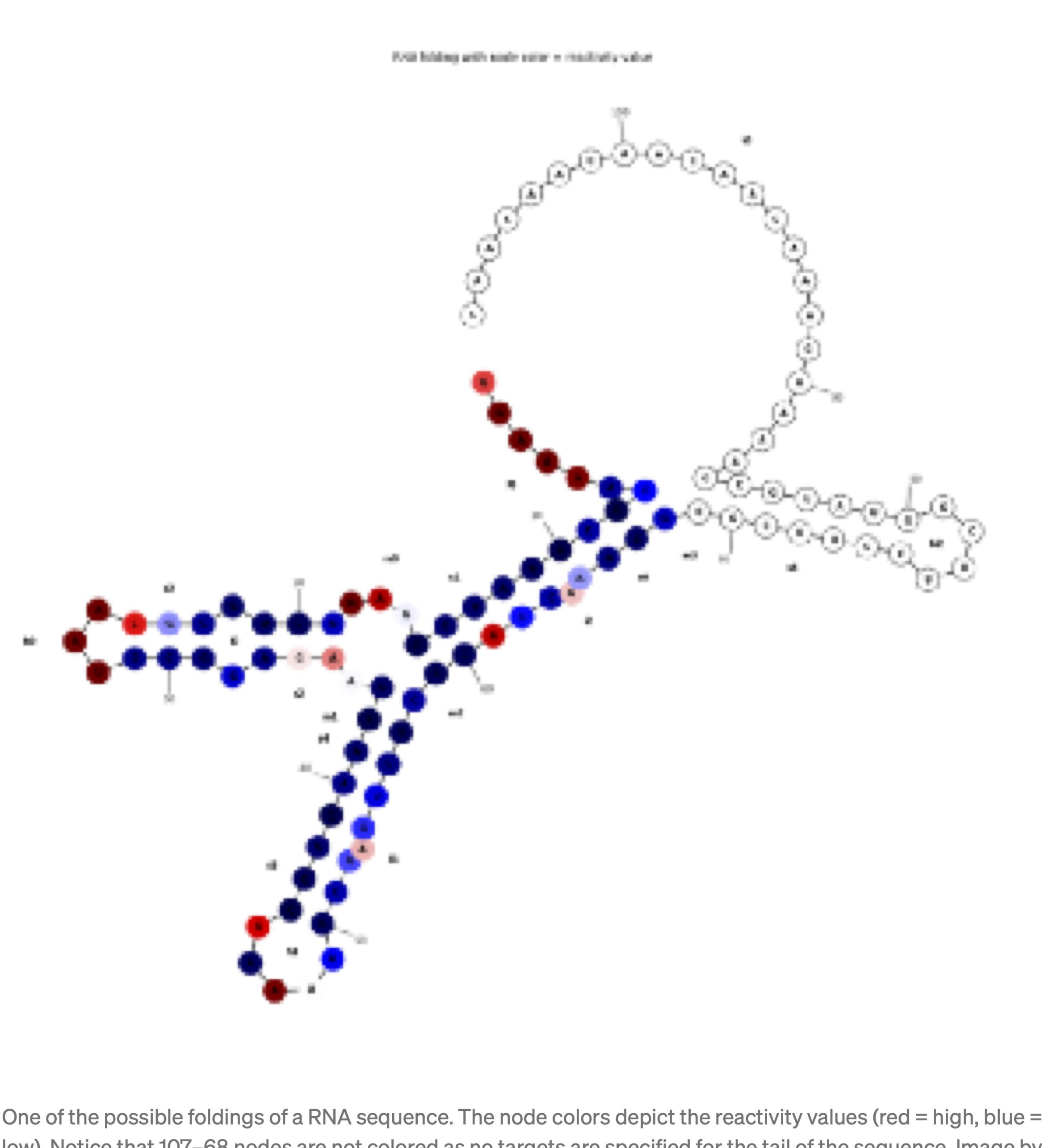
The training set consists of 2400 sequences of length 107, the public test set contains 629 sequences of length 107 and the private test set consists of 3005 sequences of length 130. There is thus a difference in the sequence length in the private test data. It is therefore of great importance that the model has built-in invariance to the sequence length. Moreover, the sequences are only partially scored. For the train and public test set, only the first 68 of 107 bases are scored while for the private test set, the first 91 out of 130 bases are scored. An example of the targets for one of the training sequences is provided below:

GGAAAAGCUCUAAUAACAGGAGACUAGGACUACGUAUUUCUAGGUAACUGGAAUAACCCAUACCAGCAGUUAGAGUUCGCUCUAACAAAAGAAACAACAACAACAAC



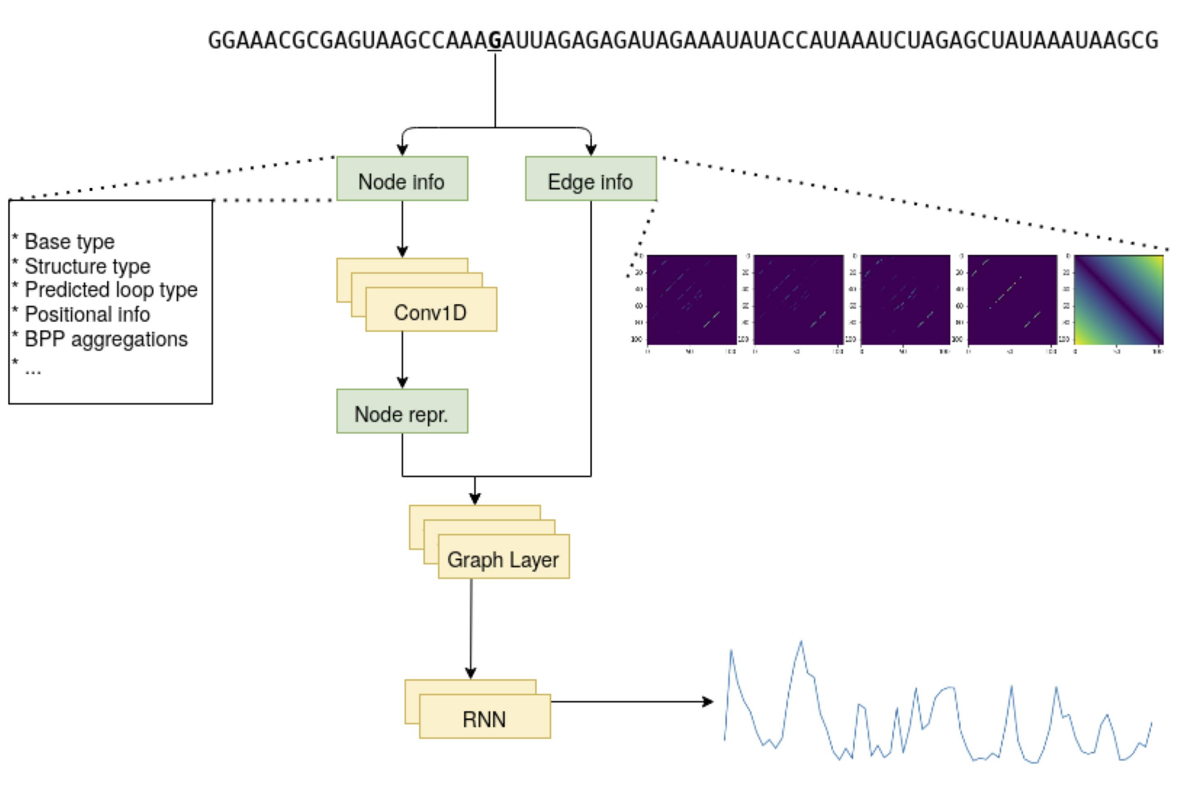
The time series of target values for one of the training sequences. Image by Author.

We can also display these target values on the RNA sequence graph directly as done for the reactivity in the example below:


One of the possible foldings of a RNA sequence. The node colors depict the reactivity values (red = high, blue = low). Notice that 107–68 nodes are not colored as no targets are specified for the tail of the sequence. Image by Author.

High-level solution overview

By looking at the targets, it is clear that a certain value is correlated with its neighbouring values in the time series. Recurrent Neural Networks are a perfect fit for these kinds of situations, as their built-in memory takes into account the nearby predicted values. We learn a representation for each of the bases in our sequence and pass these through 2 (bi-directional) RNN layers. The representation for each node is learned by using both information of the node itself and relationship or edge information. As can be seen in the introduction, the RNA sequence can be represented by a graph, and relations between certain bases in the sequence are present. We use Graph Layers to aggregate this relationship information. Our final solution consists of a blend of 4 different models, in which we vary the RNN layers (LSTM+LSTM, LSTM+GRU, GRU+LSTM and GRU+GRU). Some details are left out from this high-level overview. One important detail is that we pre-trained the network until the RNN layers as an Auto-Encoder. This allowed us to train on the unlabeled test data as well.



A high-level overview of our solution. Image by Author.

Generating extra information with ARNIE

ARNIE is a software package that wraps around 6 different libraries which can be used to generate extra information for each of the sequences:

- Vienna
- NUPACK
- RNAstructure
- RNAsoft
- CONTRAfold
- EternaFold

For a provided RNA sequence, ARNIE is able to calculate a Base Pairing Probability (BPP) matrix. Based on that BPP, a potential structure can be predicted and from these structures, the loop type can be inferred for each of the bases in the sequence.

We used different libraries in ARNIE to generate extra information. This often required a bit of bash magic in order to get everything installed. We provide a notebook to demonstrate how to generate this information. We experimented with all 6 libraries, but only found CONTRAfold to be the most useful, followed by RNAsoft, RNAstructure and Vienna.

Node features

From our high-level overview, we can see 2 types of inputs are required for our network: node information and edge information. The node information is a `NxLxV` matrix with `N` the number of RNA sequences, `L` the length of the sequences and `V` the number of features per node. We use `V=25` features in total:

- One-hot-encoded **base** ('A', 'G', 'C' or 'U') (4 features)
- One-hot-encoded **loop type** (provided: 'S': paired Stem, 'M': Multiloop, 'I': Internal loop, 'B': Bulge, 'H': Hairpin loop, 'E': dangling End or 'X': eXternal loop) (7 features)
- one-hot-encoded **positional** feature (index % 3) (3 features)
- **certainty**: fraction of packages (3 different ones: Vienna, CONTRAfold and RNAstructure) that predict `(`, `)` or `.` (3 features).
- **CapR loop type** probabilities (6 features)
- **BPP** sum and number of zeroes (2 features)

Edge features

The edge information is captured in a `NxLxLxE` matrix, with `N` the number of RNA sequences, `L` the length of the sequences and `E` the number of edge features. In total, we have `E=5` features for each edge:

- 3 different **BPPs:** the provided ones, generated with RNAsoft (ARNIE) and generated with contrafold (ARNIE)
- whether there is a **base pairing** indicated in the structure (`(` and `)`)
- the **distances** (manhattan) between bases, normalized by sequence length

The model

Our pipeline was an adaptation from this great notebook by mrmakr. Make sure to give that notebook an upvote! We did make a few modifications to the original notebook:

- Corrected the loss function of the classification part of the network.
- Changed the loss function of the auto-encoder, which was tailored for binary representations (it was a variant of binary cross-entropy). We simply used Reconstruction Loss/MAE (|y — y_hat|).
- Added 2 RNN layers after the Encoder. As was illustrated during the competition (by @tuckerarrants and others), blending a GRU+GRU, LSTM+GRU, GRU+LSTM and LSTM+LSTM model results in a slight boost. We generated models in 10-fold CV for each of these 4 combinations (so 40 models in total) and blended those with uniform weights.
- Tuned some hyper-parameters. The epochs of AE, Classification part, the number of units in the GCN layers, ...
- Adding augmented data, from the notebook made by @its7171.
- Training on all the samples but using different sample weights based on their signal_to_noise. Again a shout out to @its7171 for this! Evaluation was only performed on data with SN_filter=1. Because we expected the private LB to do that as well ;)

Things that did not work

We tried many things that did not work:

- We got 3D foldings of all the RNA and calculated 3D (euclidean) distances between the RNA bases. We also tried to use angle information (binned and then categorically encoded): this boosted our simple LSTM architecture but deteriorated the performance of the AE+LSTM. There's definitely some potential in these features, but they require some more thorough experimentation. A dataset to allow for such experimentation can be found here.
- Many software packages that give some RNA features were not informative. Some of these include RNAup, Shaker, MXFold, ...
- All of the spectral representations discussed in this paper.
- Richer graph representations (e.g. including amino-acid information)
- Entropy features
- Atom information