Comparing Computational Methods for RNA Secondary Structure Prediction

Harrison LaBollita[1] and Petr Šulc[2]

[1]Department of Physics, Arizona State University, Tempe, AZ 85281 USA
[2]Center for Biological Physics, Arizona State University, Tempe, AZ 85281 USA

Abstract

Text goes here.

Contents

1 Introduction

2 Datasets

We used two independent datasets to benchmark the performance of our CNN and Gillespie algorithm, respectivley. The dataset we used to train, validate and test our CNN contained over 50,000 RNA sequences all of the length 30 nucleotides (ntds). This dataset was generated using NUPACK software [2]. While it is desirable for CNN to function with variable length sequences, to benchmark performance of our model it was sufficient to use homogenous length sequences.

For our CNN model, we must encode the sequence and dotbracket information in such a way for the computer to understand it. The sequences are transformed into a matrix, which is described in more detail in a later section. The dotbracket representation is also transformed into a $N \times 3$ matrix, where N is the number of ntds in the sequence and the three columns represent '(', ')', or '.'. A 1 is placed in the appropriate represenation and 0's are placed in the remaining columns. For example, if the dot bracket representation of the secondary structure was '((..))', then we would encode this information in the following matrix:

$$\begin{array}{|c|c|c|}
\multicolumn{3}{c}{((..))} \\
1 & 0 & 0 \\
1 & 0 & 0 \\
0 & 0 & 1 \\
0 & 0 & 1 \\
0 & 1 & 0 \\
0 & 1 & 0
\end{array}$$

We created a second dataset from the Rfam database, which is an online colletion of thousands of RNA families that contain the sequence, secondary structure, and the free energy of the secondary structure [3, 6]. This dataset contains 27 sequences with variable lengths. The complete dataset statistics are contained in Figure 1. In a later section, we benchmark the performance of our stem level gillespie algorithm compared to a leading secondary structure predictor, Vienna RNA [1].

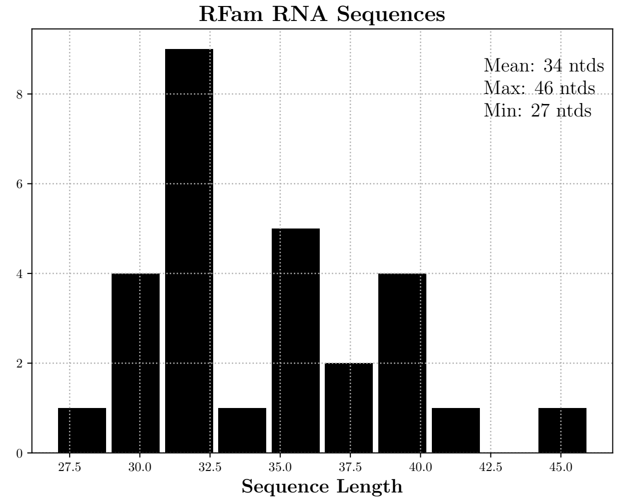


Figure 1: The Rfam dataset contains 27 total sequences with variable length. The average length is 34 ntds. The longest sequences is 46 ntds and the shortest sequence is 27 ntds.

3 CNN Predicts Secondary Structure

Convolutional neural networks (CNN)s are machine learning algorithms that are primarily used for image classification problems. The input layer is the images RGB values, which are then fed through multiple convolution layers, that are finally connected to a standard fully-connected feed forward neural network. Ultimately, to use a CNN one must have data that can be interrupted as an image. Following [8], we have replicated their CNN in PyTorch [5] in efforts to achieve similar results.

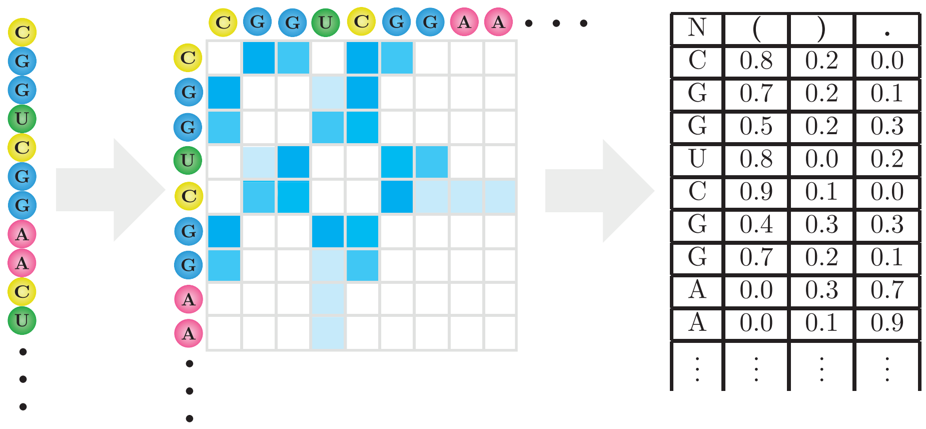
3.1 Matrix Representation of RNA

RNA sequences are comprised of any combination of four nucleotide bases: adenine (A), guanine (G), cytosine (C), and uracil (U). When using ML techniques for RNA secondary structure prediction, a formidable challenge is how to encode the sequence as input for the ML architecture. We have chosen to encode the sequence as a matrix in order to use a CNN architecture. The rules of folding a RNA secondary structure are very simple: A pairs with U, G pairs with C, and sometimes G pairs with C. The canonical pairs A-U and G-C are called Watson-Crick pairs (CITE), while the G-U pair is known as the wobble pair. Following these simple rules, we can build a weighted matrix based on which base is more likely to pair with whom. The complete matrix building algorithm is explain in [8], and our source code is available on this GitHub repository.

3.2 Model

The architecture of our CNN is simple. The model contains two convolutional layers, two pooling layers, and three fully-connected layers. We use a binary cross-entropy loss function, since we are predicting either a value of 0 or 1. We optimized the hyperparameters in our model via configuration space searching, where we randomly choose parameters from our configuration space, train our network and log the performance. We found that a learning rate of 0.001 and momentum of 0.9 performed the best without overfitting.

We have illustrated the workflow of our CNN Model in Figure 2. We begin with an RNA sequence, convert the sequence into a matrix, then feed it into our CNN. The output of the CNN is a $N \times 3$ matrix, where N is the number of nucleotides in the sequence and our model's predicted values of each nucleotide being paired or unpaired. From this information, we reconstuct the dotbracket representation by choosing the highest value in each row as the model's choice for paired ("(" or ")") or unpaired (".") and compare our predicted structure with the actual secondary structure for this sequence.



N	(	)	.
C	0.8	0.2	0.0
G	0.7	0.2	0.1
G	0.5	0.2	0.3
U	0.8	0.0	0.2
C	0.9	0.1	0.0
G	0.4	0.3	0.3
G	0.7	0.2	0.1
A	0.0	0.3	0.7
A	0.0	0.1	0.9
⋮	⋮	⋮	⋮

Figure 2: We convert our dataset of RNA sequences into a new dataset of RNA matrices. These matrices are used as inputs into our CNN. Notice that the darker blue squares correspond to heavier weighted base meaning more likely to be paired. The output of our network is the matrix on the far right which we use to reconstruct the dot bracket represenation of the secondary structure.

We trained and validated our CNN model on dataset containing 50,000 uniform length RNA sequences following the standard condition, where 80% of the data is used for training and 20% of the dataset was used for validation. We trained our network for 100 epochs with a batchsize of 100. We quantified the accuracy of our model by counting the number of mistakes our model made when predicting the secondary structure. A perfect prediction corresponds to 0 mistakes.

3.3 Results and Discussion

In this section, we present our CNN model's performance on RNA secondary structure prediction. In Figure 3, we have plotted the training and validation accuracy of our model throughout the training session. We can see that the model converges to about 84% accuracy for both training and validation accuracies.

Our CNN model achieves similar results to the [8], however, our model will sometimes predict non-physical structures. Therefore, it is necessary that we include a dynamic layer in our model that corrects for these non-physical outputs. This is the central weakness of all non-physical secondary structure predictors. Without including knowledge of physical considerations these models will always produce unphysical secondary structures.

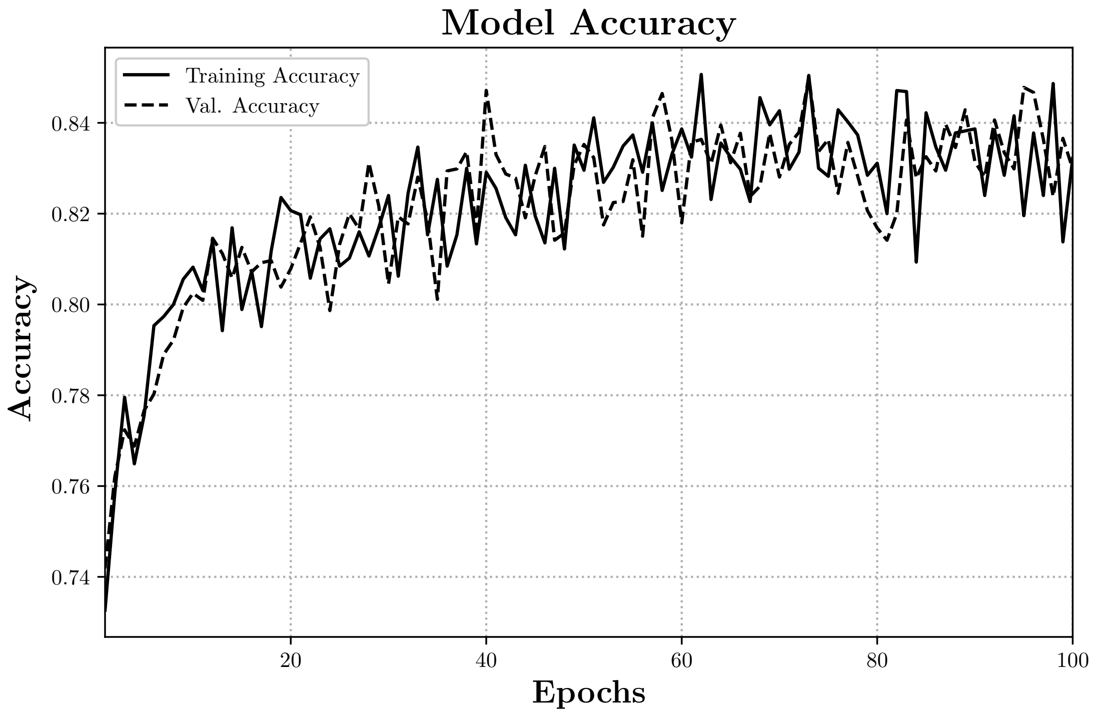


Figure 3: Training and validation accuracy over the 100 epochs. The solid black line is the training accuracy and the dotted black line is the validation accuracy.

4 Stem Level Gillespie Algorithm

4.1 Introduction to Gillespie Algorithm

In [3], the following alogrithm is presented:

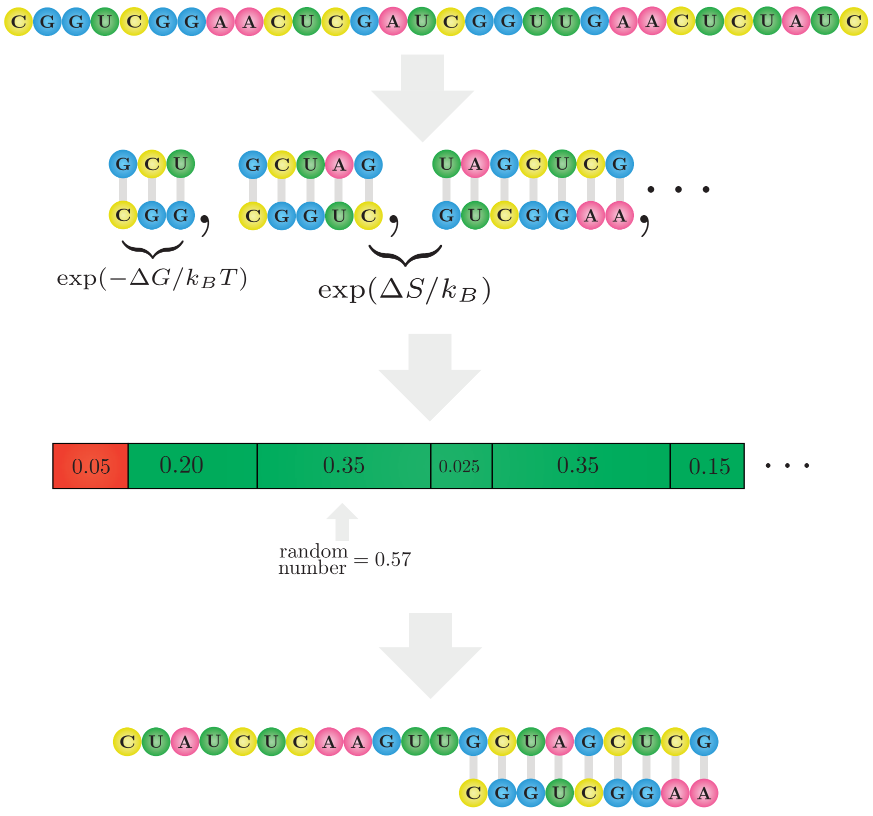
1. Extract the total flux Φ from the partial sum table.
2. Choose two random numbers r_1 and r_2 on the interval $[0,1]$.
3. Increment the time by $\tau = -\ln(r_2)/\Phi$.
4. Identify the first *stem* which satisfies the inequality

$$\sum_{\ell=1} \phi_\ell \geq r_1\Phi$$

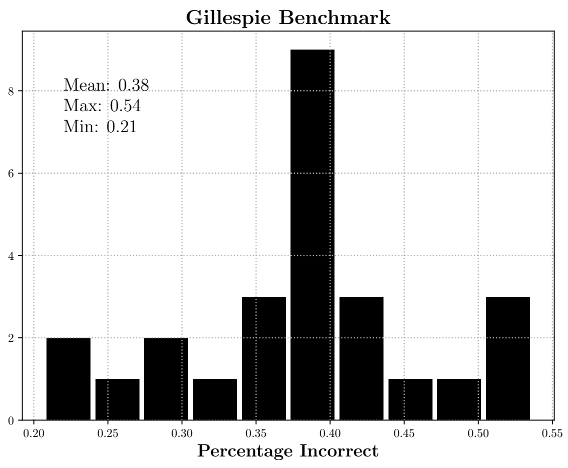
   by using our table of reaction rates for each stem. We then recalculate the remainder $\Phi = r_1\Phi - \sum_{\ell=1}\phi_\ell$ on the fly.

5. Once this condition is met we choose this stem to form. We make sure it is compatible with all of the other stems in the current structure. If it is not compatible with the current structure, we remove the incompatible pieces and add our new stem. However, if adding this move requires breaking almost all of the current stems then we do not allow this move to occur.
6. Go to Step 1 until the the arbitraty cutoff time has been reached.

4.2 Stem Level Implementation



4.3 Results



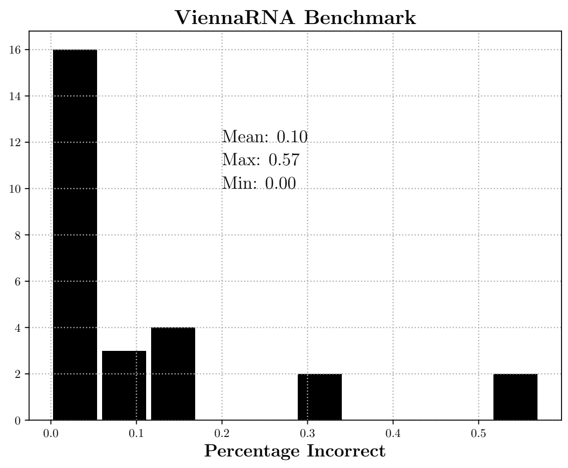


Figure 4

Figure 5

5 Conclusion

References

[1] Ronny Lorenz et al. "ViennaRNA Package 2.0". In: *Algorithms for Molecular Biology* 6.1 (2011), p. 26. ISSN: 1748-7188. DOI: 10.1186/1748-7188-6-26. URL: https://doi.org/10.1186/1748-7188-6-26.

[2] Joseph N. Zadeh et al. "NUPACK: Analysis and design of nucleic acid systems". In: *Journal of Computational Chemistry* 32.1 (2011), pp. 170–173. DOI: 10.1002/jcc.21596. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/jcc.21596. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/jcc.21596.

[3] Eric C. Dykeman. "An implementation of the Gillespie algorithm for RNA kinetics with logarithmic time update". In: *Nucleic Acids Research* 43.12 (May 2015), pp. 5708–5715. ISSN: 0305-1048. DOI: 10.1093/nar/gkv480. eprint: http://oup.prod.sis.lan/nar/article-pdf/43/12/5708/16659220/gkv480.pdf. URL: https://doi.org/10.1093/nar/gkv480.

[4] Ioanna Kalvari et al. "Rfam 13.0: shifting to a genome-centric resource for non-coding RNA families". In: *Nucleic Acids Research* 46.D1 (Nov. 2017), pp. D335–D342. ISSN: 0305-1048. DOI: 10.1093/nar/gkx1038. eprint: http://oup.prod.sis.lan/nar/article-pdf/46/D1/D335/23162120/gkx1038.pdf. URL: https://doi.org/10.1093/nar/gkx1038.

[5] Adam Paszke et al. "Automatic differentiation in PyTorch". In: (2017).

[6] Ioanna Kalvari et al. "Non-Coding RNA Analysis Using the Rfam Database". In: *Current Protocols in Bioinformatics* 62.1 (2018), e51. DOI: 10.1002/cpbi.51. eprint: https://currentprotocols.onlinelibrary.wiley.com/doi/pdf/10.1002/cpbi.51. URL: https://currentprotocols.onlinelibrary.wiley.com/doi/abs/10.1002/cpbi.51.

[7] Ofer Kimchi et al. "RNA structure prediction including pseudoknots through direct enumeration of states". In: *bioRxiv* (2018). DOI: 10.1101/338921. eprint: https://www.biorxiv.org/content/early/2018/06/04/338921.full.pdf. URL: https://www.biorxiv.org/content/early/2018/06/04/338921.

[8] Hao Zhang et al. "A New Method of RNA Secondary Structure Prediction Based on Convolutional Neural Network and Dynamic Programming". In: *Frontiers in Genetics* 10 (2019), p. 467. ISSN: 1664-8021. DOI: 10.3389/fgene.2019.00467. URL: https://www.frontiersin.org/article/10.3389/fgene.2019.00467.