A Deep Learning Framework for RNA Secondary Structure Prediction

Aditya Mittal, Karthik Mittal, Sid Srivastava, Krish Suraparaju

Abstract

Purpose: RNA secondary structure prediction is a vital stepping stone towards tertiary prediction, providing researchers with an understanding of protein binding mechanisms. Through identification of hydrogen bond patterns, we discover important structures that govern tertiary binding, like pseudoknots, hairpins, and internal/external loops. Finding a more efficient mechanism for calculating these secondary structures can lead to exciting discoveries in the RNA field.

Methods: We construct a deep learning framework for RNA secondary structure prediction that reproduces the work of Zhang et al. 2019 [1] that converts sequences into a corresponding neighbor matrix, runs a CNN that outputs a dot-bracket structure, and uses dynamic programming to compute a maximum probability sum model for the secondary structure. This framework is compared to the landmark Nussinov algorithm.

Results: Running against 1000 sequences in the bprna dataset, we observed an accuracy of 38.32% for Nussinov [2], 49.4% for Zuker [3], and 43.2% for our algorithm. This shows the deep learning framework wasn't effective as the baseline Zuker algorithm in RNA secondary structure prediction; we conclude this is likely because of interpolation of paired bases in the loop region of the stem-loop structure. In the future, we hope to train against a larger subset of bprna and consider pseudoknots in our analysis.

1 Introduction

1.1 Motivations

RNA is a crucial component of the regulation and expression of genes within organisms. RNA transcripts typically fold via base pairings into complex patterns, which forms their secondary structure. These secondary structures are incredibly useful to analyze, as they impact ligand binding and scaffolding, as well as other functions [4].

One useful research area of RNA secondary structure analysis involves structural peturbations to coding RNA. For example, mutating certain base pairs of tRNA can reduce the structural integrity of its core, leading to weaker binding of aminoacyl-tRNA to the A/T site, among other effects [5]. This occurs because tRNA conformation depends on interaction between the D- and T- loops, which will be affected because of the aforementioned base pair mutations.

Analysis of secondary structure has also been pertinent in understanding the functions of various types of RNA, including ribosomal RNAs, microRNAs, nucleolar RNAs, and small nuclear RNAs [4] [6]. Each of these RNAs play a crucial role in our bodily function; for instance, microRNAs are used to repress translation and transcription within eukaryotes.

1.2 Definitions

1. **Primary structure:** a single-stranded sequence of nucleotides (A, U, C, G)
2. **Secondary structure:** a structure that assigns each nucleotide as paired or unpaired, where the pairing consists of all base-base interactions [8]

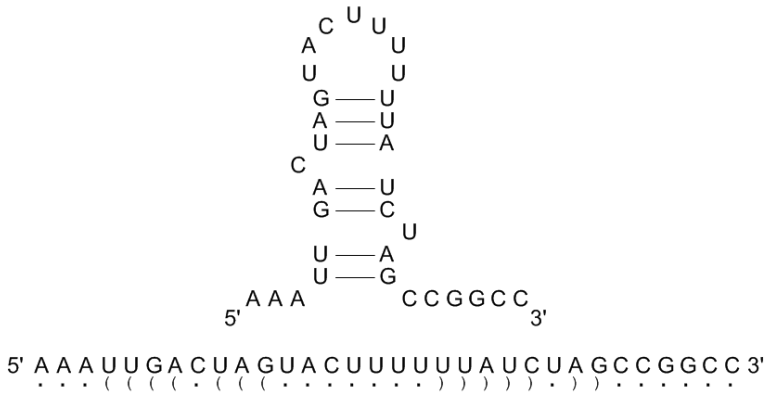

Fig. 1 Example of dot bracket notation compare to RNA secondary structure diagram. [7]

3. **Hairpin loop:** two regions in the same strand base-pair to form a double-strand helix which ends with an unpaired loop
4. **Psuedoknots:** a secondary structure containing two hairpin structures where half of a stem is intercalated between two halfs of another stem [9]
5. **(Gibbs) free energy:** energy available in a system to do work
6. **Dot-bracket notation:** a method to represent RNA secondary structure by denoting unmatched nucleotides with . and matching pairs with (). Reference Figure 1 for a diagram.
7. **Wobble base pair**: A pairing between two nucleotides that doesn't follow typical Watson Crick base pairing rules. We will only define guanine-uracil as a wobble.

1.3 Computational Problem

We define the general RNA secondary secondary prediction problem as defined below:

1. **Input:** RNA primary structure s and other hyperparameters (depending on the algorithm chosen)
2. **Output:** secondary structure of s in dot-bracket notation

We attempt to solve this problem with various dynamic programming and deep learning algorithms, as explained below.

Nussinov Algorithm:

1. **Input:** RNA primary structure s and minimum hairpin loop length m
2. **Output:** secondary structure of s in dot-bracket notation maximizing the total number of bonds between bases

Zuker Algorithm:

1. **Input:** RNA primary structure s, hairpin energy $eH(i,j)$, stacking energy $eS(i,j)$, internal loop energy $eL(i,j,i',j')$, multiloop energy $eM(i,j,k,k')$ where k is the number of base pairs and k' is the number of unpaired bases within loop, stem energy $eSt(i,j)$, and a distance constant d
2. **Output:** secondary structure of s in dot-bracket notation

Deep Learning Algorithm:

1. **Input:** RNA primary structure s, sliding window length l, and CNN network parameters p
2. **Output:** secondary structure of s in dot-bracket notation

1.4 Current Algorithms

There are three major classes of RNA secondary structure prediction algorithms. The first class uses dynamic programming and encompasses algorithms such as Nussinov [2], Zuker [3], and RNAfold [10]. The Nussinov algorithm maximizes the total number of bonds between bases; because its optimization function is simple, it does not account for the 3D shape of RNA or predict pseudoknots. Additionally, it does not account for a minimum stem loop size but the algorithm has been tweaked to account for this. Zuker's algorithm builds on the Nussinov algorithm by removing the unrealistic assumption of base pair maximization. Its optimization function minimizes the free energy of RNA based on the Gibbs free

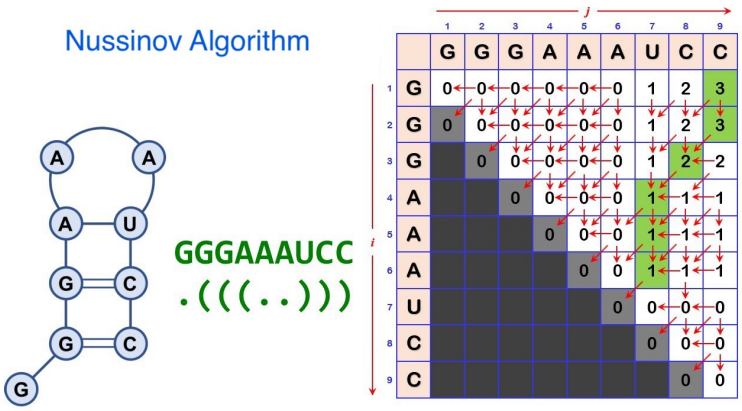

Fig. 2 Nussinov algorithm dot bracket and matrix example. Source: YouTube (Ashok Kumar)

energy of adjacent base pairs and the types of base pairing. The Zuker algorithm have been implemented in web applications, such as RNAfold. Note that this optimization function is still unrealistic, and thus, the Zuker algorithm decreases in accuracy for longer RNAs. For a more detailed explanation of Nussinov and Zuker's algorithm, reference 2.1 and 2.2.

The second class uses comparative sequence analysis to process sets of homologous sequences of RNA, where there is conservation of structure. Currently, there are three forms of this analysis. The first uses stochastic context-free grammars to give a prior probability distribution about the structures based on the evolutionary history [11]. The second optimizes for parsimony and free energy while minimizing mutations for multiple sequences at a time [12]. The third represents structures as trees and analyzes mutations as fusing/deleting nodes or edges, allowing for bioinformaticians to more easily analyze the structural cores of the RNA sequences [13].

The last class, and perhaps most relevant class to our work, involves the use of artificial intelligence. Some of the oldest uses of AI use the genetic algorithm to find common secondary structure elements between homologous sequences of RNA [14]. Other approaches use support vector machines and neural networks to predict RNA secondary structure. More recently, deep learning has been used to infer hidden features in large datasets to construct effective prediction algorithms. One recent example is UFold, which uses fully convolutional networks (only performs convolution operations without intermediate fully connected layers) to process RNA sequences [15].

1.5 Our Approach

We compare traditional dynamic programming algorithms to deep learning approaches of tackling RNA secondary prediction. Specifically, we compare the Nussinov algorithm to a CNN-based prediction model [4]. We hope this analysis will highlight the effectiveness of using machine learning in secondary structure prediction.

2 Methods

2.1 Nussinov Algorithm

The Nussinov algorithm [2] is a novel RNA secondary structure prediction model that forms a structure with maximum base pairing. Through a dynamic programming approach, it follows a 4x4 scoring table S (lower values indicate greater stability) for each of the four RNA nucleotides. The time complexity is $O(n^3)$ where n is the length of the RNA string.

To solve the computational problem described in Section 1.3, we construct a matrix M, which stores scores based on complementarity of base pairs (higher scores are rewarded more). This is an upper triangular matrix, computed recursively through a recurrence relation emphasizing dot bracket notation. As shown above constructing this matrix, we discover the minimum free energy path using backtracking pointers formed from the feedforward recurrence.

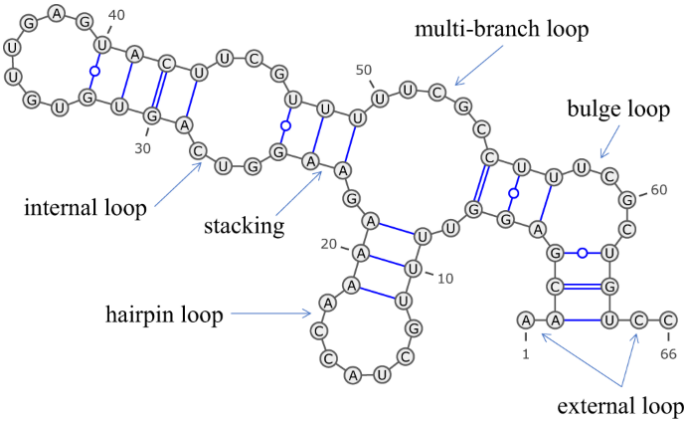

Fig. 3 Examples of various types of structural loops that contribute to free energy.[16]

Formalized, the steps of the algorithm are as follows for an RNA sequence R:

- Construct initial matrix M with $M(i,i) = 0, M(i,i-1) = 0$ for $1 \leq i \leq n$.
- Fill M with the recurrence: $M(i,j) = \max_{i \leq k < j} \begin{cases} M(i,k-1) + M(k+1,j-1) + S(R_k, R_j) \\ M(i,j-1) \end{cases}$ We note that $S(R_k, R_j)$ is 1 if S_k and S_j are complementary and 0 otherwise. For example, as shown in Fig 2, we see that $M(1,7) = 1$, because for $k = 6$, $M(i,k-1) + M(k+1,j-1) + S(R_k, R_j) = M(1,5) + M(7,6) + 1 = 1$ since A and U are complementary. We note that the entire first row was equal to zero before this because there $\nexists k, j$ such that R_k, R_j were complementary.
- We do an iterative backtrack with a starting initialization of $i = 1, j = n$. If
  - $j \leq i$, halt.
  - $M(i,j) = M(i,j-1)$, then set $i = i, j = j-1$. Run our backtrack again.

  Otherwise, for all $i \leq k < j$, if S_k and S_j complementary and $M(i,k-1) + M(k+1,j-1) + 1$, then we add (k,j) to our final output and run our backtrack twice with $i = i, j = k-1$ and $i = k+1, j = j-1$. We return our final output.
  Figure 2 below shows an example backtrack, which starts from the upper right corner and recursively moves down and to the left.
- Convert pairs to dot bracket notation with parantheses and nonpairs with points.

The Nussinov code can be found here. We set the minimum loop length to zero.

2.2 Zuker Algorithm

The Zuker algorithm is a landmark RNA secondary structure prediction model that minimizes the free energy of RNA. We first discuss various structural elements that contribute to free energy in the RNA system:

1. Hairpin loop: As discussed before, a hairpin loop are two distinct regions that form a double-stranded helix with an unpaired loop through complementary base pairing.
2. Stacking loop: The packing of base pairs into a double-stranded helix.
3. Internal loop: A place where double-stranded RNA separates due to a lack of complementary base pairing. Asymmetrical internal loops are called left/right bulges based on the side of asymmetry
4. Multi-loop: A structure of multiple mini-loops within a larger loop connected to a double-stranded helix.

Reference Figure 3 for a visual diagram of loops. For simplicity, we only consider the hairpin function, namely $eH(i,j) = 2(i-j+5)$ and $eS, eL, eM = 0$, when implementing the Zuker algorithm. We also set the distance constant d to 5.

Lastly, we set $eSt(s,i,j) = \begin{cases} -4 & \text{s[i] and s[j] pair} \\ 0 & \text{s[i] and s[j] are wobble base pairs} \\ 4 & \text{else} \end{cases}$.

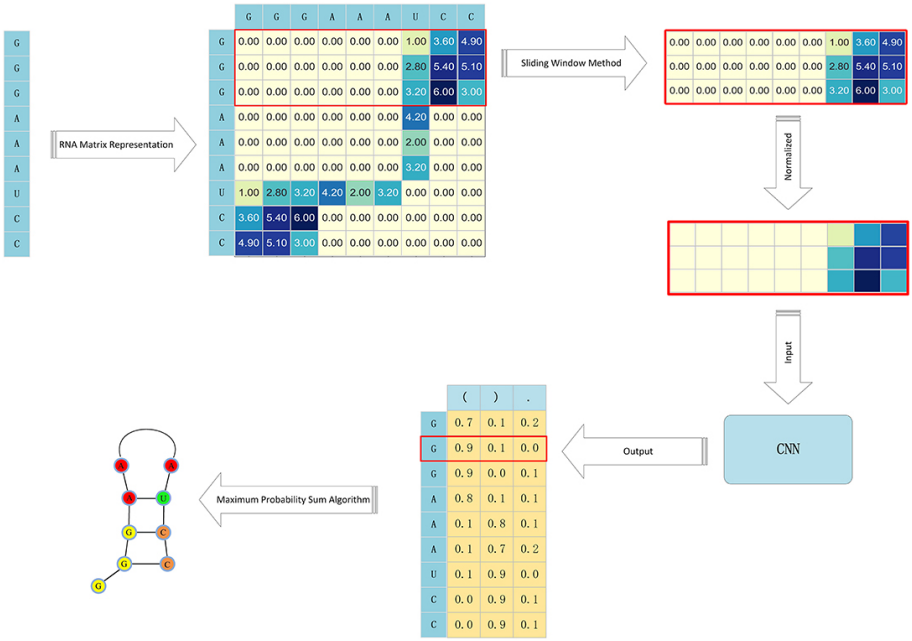

Fig. 4 Architecture of CNN prediction and maximum probability sum algorithm [4]

We define the W-matrix as the minimum ij-substructure of the RNA sequence for some element W_{ij} in the matrix. We define the V-matrix as the minimum closed ij-substructure of the RNA sequence for some element W_{ij} in the matrix (namely (i, j) are paired).

We run the following steps:

1. Initialize V and W as such: $V_{ij}/W_{ij} = \begin{cases} \infty & j + d > i \\ 0 & \text{else} \end{cases}$. Create an $n \times n$ matrix path as well.
2. For all $1 \leq i, j \leq n$ where $j + d \leq i$ and $W[i-1, j]$ or $W[i, j+1] = 0$:
   (a) Calculate the hairpin energy ($eH(i, j)$+$eS(s, i, j)$) and the match energy ($W[i-1, j+1]$+$eS(s, i, j)$). Set $V[i, j]$ equal to the minimum of those and set struct equal to H or M respectively.
   (b) Let $m = \min_{j+2 \leq k \leq x}(W[i, k] + W[k-1, j])$. Let $W[x, y] = \min(W[i-1, j], W[i, j+1], V[x, y], min)$ and set path equal to L, D, struct, or m respectively based on the minimum value of $W[x, y]$.
3. Create an $n \times n$ matrix backtrack and an empty list match, and backtrack starting with $x = 0, y = 0$. Set backtrack[x, y] = #. While path[x, y] is not equal to H do the following:
   (a) If path[x, y] = M, append (x, y) to match, increment x by 1, and decrement y by 1.
   (b) If path[x, y] = L, decrement y by 1.
   (c) If path[x, y] = D, increment x by 1.
   (d) Set backtrack[x, y] = #.
4. Convert to dot-bracket notation based on the matches.

Note this algorithm creates $O(n^2)$ matrices where n is the length of RNA sequence s, and it has been shown that this algorithm runs in $O(n^3)$ time [17].

Tbe Zuker code can be found here with the hyperparameters specified above.

2.3 Deep Learning Approach

2.3.1 Architecture

The overall architecture is shown in Figure 5. We convert a string of length S into a $|S| \times |S|$ matrix using the preprocessing method described in Section 2.3.2. Afterwards, we use a sliding window method and CNN classifier to output a $|S| \times 3$ matrix of probabilities for each dot bracket letter. We then use a maximum probability sum algorithm to describe the RNA secondary structure.

Convolutional neural networks [18] are machine learning frameworks that aim to find distinctive features in images to reach an objective goal. Convolutional layers (or hidden layers) have filters that pass through an input matrix and perform element-wise multiplications to produce new output matrices. Usually, CNNs are used for classification; for example, an animal classifier may have filters that

delineate prominent features in animals, like whiskers, trunks, and hind legs. After traversing the neural network, a probabilistic output will show from the previous convolutional layers.

In the RNA secondary sequence problem [1], the objective goal is to discover a folding with minimum free energy using CNNs and dynamic programming.

2.3.2 Preprocessing

First, we encode our RNA string S as an $|S| \times |S|$ matrix W. We set the pairing weight, based on if it's a wobble base pair and the number of hydrogen bonds:

$$P(S_i, S_j) = \begin{cases} 2 & (R_i = A \wedge R_j = U) \vee (R_i = U \wedge R_j = A) \\ 3 & (R_i = G \wedge R_j = C) \vee (R_i = C \wedge R_j = G) \\ x & (R_i = G \wedge R_j = U) \vee (R_i = U \wedge R_j = G) \\ 0 & \text{otherwise} \end{cases}$$

x is bounded between 0 and 2. We define our pairing weight as so since dot bracket notation heavily depends on the # of hydrogen bonds and complementarity. We also want to account for wobble base pairs to add complexity to our algorithm (since they are commonly seen inside of biological processes).

To increase the complexity of this encoding, we account for paired bases on the stem. So, for arbitrary $i, j \in S$, we checked pairings for all bases to the left of i and the right side of j. Since the closer i and j are, the higher the weight/effect for base pairs, we integrated a Gaussian function that updates the weights to the left/right of i, j:

$$W_{left} = W_{left} + e^{-\frac{\alpha^2 \cdot P(S_{i-\alpha, j+\alpha})}{2\tau^2}}, W_{right} = W_{right} + e^{-\frac{\beta^2 \cdot P(S_{i-\beta, j+\beta})}{2\tau^2}}$$

where $\alpha, \beta \in \mathbb{N}$ delineate how far the base pairs are away from i and j.

For example, string ACUGGGAUAAAA has matrix W shown below. $W_{31} = 2$ since $R_3 = U$ and $R_1 = A$. The actual Gaussian function is only applied to $W_{43}, W_{53}, W_{63}, W_{48}, W_{58}, W_{68}$ after checking paired bases on the stem.

	A	C	U	G	G	G	A	U	A	A	A	A
A	0	0	2	0	0	0	0	2	0	0	0	0
C	0	0	0	3	3	3	0	0	0	0	0	0
U	2	0	0	0	0	0	2	0	2	2	2	2
G	0	3	0.8	0	0	0	0	0.8	0	0	0	0
G	0	3	0.8	0	0	0	0	0.8	0	0	0	0
G	0	3	0.8	0	0	0	0	0.8	0	0	0	0
A	0	0	2	0	0	0	0	2	0	0	0	0
U	2	0	0	0	0	0	2	0	2	2	2	2
A	0	0	2	0	0	0	0	2	0	0	0	0
A	0	0	2	0	0	0	0	2	0	0	0	0
A	0	0	2	0	0	0	0	2	0	0	0	0
A	0	0	2	0	0	0	0	2	0	0	0	0

2.3.3 CNN Prediction

We use a sliding window of size $19 \times |S|$ after inspection of the length of the stem regions of our input RNAs. Before running our CNN, we increase the uniformity of our input data since the size of each RNA input differs from each other; we do this by normalizing the data according to the mean RNA length. So, we have that $W' = \frac{W}{|S|}$ where S is the RNA string and W is the original preprocessed matrix.

The output of the CNN is a $|S| \times 3$ matrix which contains three probabilities for each letter in the string: $p_{left}, p_{right}, p_{point}$, which are the probabilities of having a (,), or . in the dot bracket notation respectively.

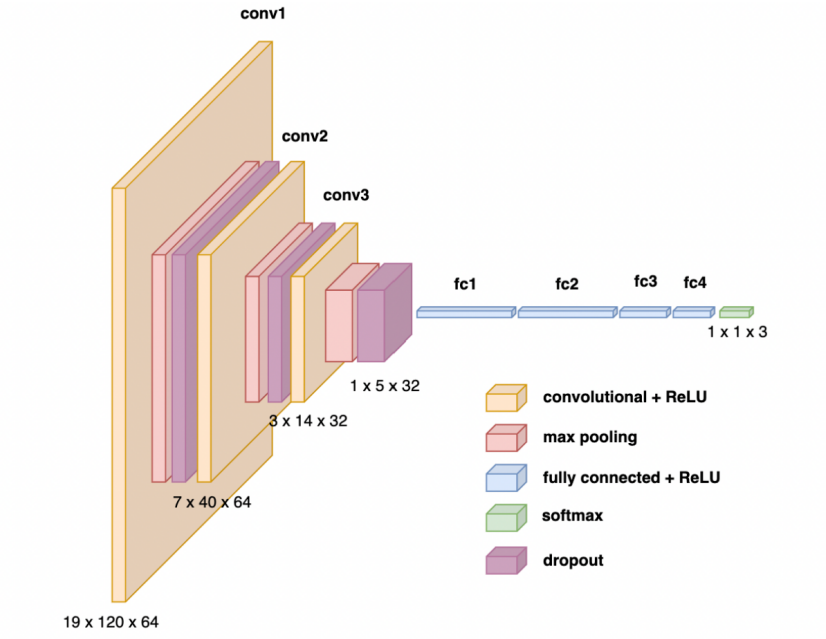

Fig. 5 CNN Layer Architecture Diagram

In our model, we use 3 2D convolutional layers with sixteen filters, 3 max pooling layers, and two dense layers (with ReLU activation). This is done using the Keras package in Python. We use an Adam optimizer and have categorical cross entropy loss when training our model. Lastly, we have a batch size of 32 against our bprna dataset. The underlying specifications of our code can be found in the Github. The overall model architecture is as follows:

2.3.4 Maximum Probability Sum Algorithm

For each index of the sequence, the CNN outputs three probabilities: $p_{left}, p_{right}, p_{point}$, which corresponds to $(,),.$ in the dot bracket notation respectively.

Note that we can't simply take the highest probability for each index since they may not produce a suitable secondary structure (specifically in ensuring there's a matching number of left/right brackets). If there are more left than right brackets (or vice versa), then a base pair must be matched to multiple other base pairs, which is impossible in RNA secondary structure, since one base pair can only be matched to another. Therefore, one of our constraints is ascribing to the dot bracket structure.

To arrive at the final pairing, we define our iteration recurrences as follows (ensuring dot bracket notation and maximum probability sum constraints are satisfied):

$$M(i,j) = \begin{cases} M(i+1,j) + p_{point}(S_i) \\ M(i,j-1) + p_{point}(S_j) \\ M(i+1,j-1) + \delta(S_i, S-j) \\ \max\limits_{i<k<j} M(i,k) + M(k+1,j) \end{cases} \qquad \delta(S_i,S_j) = \begin{cases} p_{left}(S_i) + p_{right}(S_j) & \text{if paired} \\ p_{point}(S_i) + p_{point}(S_j) & \text{if not paired} \end{cases}$$

where $p_{left}, p_{right}, p_{point}$ are probabilities outputted by the CNN.

The maximum probability sum algorithm code can be found here.

2.4 Dataset

We use the bprna [19] dataset, which contains RNA and their annotated secondary structures, to train our CNN. We started with 1000 sequences of less than 120 nucleotides to evaluate the model's efficacy. For our neural network, we used a 60-40 train-test split on these sequences to train and test our adapted algorithm.

For each of these algorithms, after computing the pairwise predicted dot-bracket notation structure, we compared these with the actual structure in the bprna data.

3 Results

3.1 Performance of the Nussinov and Zuker Algorithm

From the bprna [19] dataset, we picked 1000 sequences to calcualate the performance of the Nussinov and Zuker algortithm. To ensure a fair comparison, we only considered sequences that were less than 120 base pairs, since that's the subs set of the bprna dataset that we drained our CNN on. In short, we ran the algorithm on the chosen sequences, and compared how similar the predicted structure is to the actual secondary structure. Note that RNA structure is represented using dot-bracket notation, so a simple way to measure similarity was to traverse the sequences pairwise and check for equality. Code for running these tests on the Nussinov and Zuker algorithm can be found here.

Formally, we define accuracy of a single sequence to be the percent of predicted characters in the dot-bracket sequence that matched with the actual sequence. We define overall accuracy to be the average across all accuracies from the 1000 sequences. After running tests, we see that the Nussinov algorithm had a 34.2% accuracy while the Zuker algorithm had a 49.4% accuracy.

3.2 Performance of the Convolutional Neural Network

After 50 epochs of training, our model has a validation accuracy of around 82%. The training and validation accuracies and loss over time are shown in the following two figures:

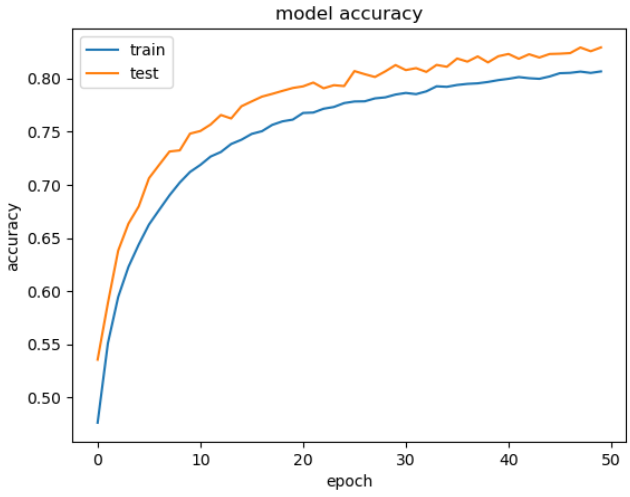

Fig. 6 Model accuracy plotted against epochs

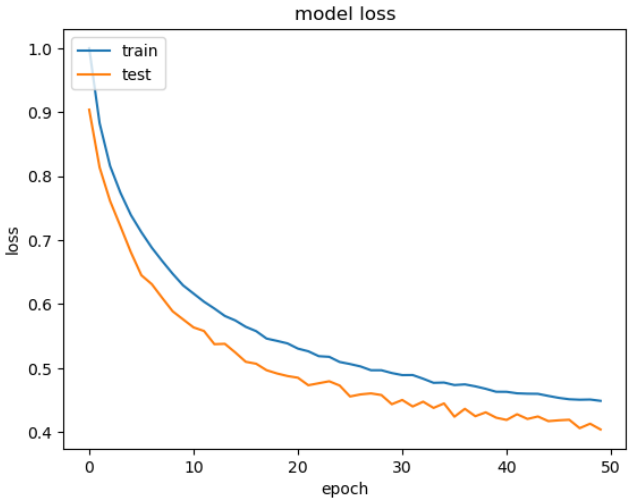

Fig. 7 Model loss plotted against epochs

After running the maximum probability sum algorithm, which outputs a dot bracket structure from the probability matrix, we compared these predicted structures to the actual structures from our dataset. We ran this across the 400 test sequences (60-40 train-test split) and achieved a 43.2% accuracy.

We note our algorithm performs worse than the Zuker algorithm but performs better than the Nussinov algorithm, when using 1000 sequences from the bprna dataset.

4 Conclusion

We reproduced the work of Zhang et al. 2019 [1] and tested it against different baselines, like the Nussinov and Zuker algorithm. The Nussinov and Zuker algorithms had unrealistic optimization functions, optimizing on the number of paired nucleotide bases and the minimum free energy respectively. Therefore, we proposed a new machine learning focused algorithm that integrates a modified Nussinov recurrence based on the CNN output, a matrix of dot bracket probabilities for each index.

After redefining our CNN architecture to add extra complexity with more convolution kernels and dense neurons, reduce overfitting with extra dropout layers, and increase accuracy by tuning hyperparameters, we found our trained CNN accuracy to be approximately 83%, with a CNN loss of 0.404. After adding these probabilities to the maximum probability sum algorithm, we had an accuracy of 43.24%. This performed better than Nussinov (38.32%) and worse than Zuker (49.4%).

Some limitations of our approach were that we only trained on a limited dataset size of 1000 distinct RNA sequences, since this translated to approximately 120K input matrices to input to the CNN. For future directions, we could train on a larger subset of the bprna dataset, to increase training accuracy. Furthermore, our algorithm does not take pseudoknots, which is a structure containing at least two stem-loop structures where half of one stem is interconnected with the half of another, in account.

There is also a significant discrepancy between our convolutional neural network accuracy and our adjusted accuracy after running the modified Nussinov dynamic programming algorithm. While we were not able to increase our accuracy, we have an idea of why this might be occurring. When we looked at the actual base pair outputs of the CNN and DP algorithm, we found that the DP algorithm was interpolating many paired bases in the "loop" region of the stem-loop structure in many of these RNAs. Examples of this can be found by looking at the following sequences:

```
UGCCUGGCGGCCUUAGCGCGGUGGUCCCACCUGACCCCAUGCCGAACUCAGAAGUGAAACGCCGUAGCGCCGAUGGUAGUGUGGGGUCUCCCCAUGCGAGAGUAGGGAACUGCCAGGCAU
.(((((((((.......)))..)......(..(((((((.((((...)))))...).....())....)))))..)).......))))))....(((..(((()....)))))..)))))))))).
.(((((((((.(()).).)).(().).()(..((((((..((((.().).).((((....())(...()).).().)).)))))).)(((((.(()(..(((((((.)))))))).)))))))..
```


```
UGCCUGGCGGCCUUAGCGCGGUGGUCCCACCUGACCCCAUGCCGAACUCAGAAGUGAAACGCCGUAGCGCCGAUGGUAGUGUGGGGUCUCCCCAUGCGAGAGUAGGGAACUGCCAGGCAU
.(((((((((.......)))..)......(..(((((((.((((...)))))...).....())....)))))..)).......))))))....(((..(((()....)))))..)))))))))).
.(((((((((.(()).).)).(().).()(..((((((..((((.().).).((((....())(...()).).().)).)))))).)(((((.(()(..(((((((.)))))))).)))))))..
```

For both sequences, the top dot bracket string is the CNN output, while the bottom dot bracket string is the output of the dynamic programming. As is evident, the dynamic programming is adding interpolated base pairing within the longer stretches of dots, which correspond to the unpaired loop regions. This may be because our neural network was trained on a limited amount of data and is not passing accurate enough base pairing information to the dynamic programming algorithm. Our high loss also means that the model is not as sure about its predictions, so the probabilities of the non-correct symbols is higher and thus affects the outcome of the maximum probability sum algorithm. To rectify this, we can train on more data, which we could not do due to a lack of computational power.

References

[1] Zhang, H., Zhang, C., Li, Z., Li, C., Wei, X., Zhang, B., Liu, Y.: A new method of RNA secondary structure prediction based on convolutional neural network and dynamic programming. Frontiers in Genetics **10** (2019) https://doi.org/10.3389/fgene.2019.00467

[2] Nussinov, R., Pieczenik, G., Griggs, J.R., Kleitman, D.J.: Algorithms for loop matchings. SIAM Journal on Applied Mathematics **35**(1), 68–82 (1978) https://doi.org/10.1137/0135006

[3] Zuker, M., Stiegler, P.: Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. Nucleic Acids Research **9**(1), 133–148 (1981) https://doi.org/10.1093/nar/9.1.133

[4] Vandivier, L.E., Anderson, S.J., Foley, S.W., Gregory, B.D.: The conservation and function of RNA secondary structure in plants. Annual Review of Plant Biology **67**(1), 463–488 (2016) https://doi.org/10.1146/annurev-arplant-043015-111754

[5] Pan, D., Zhang, C.-M., Kirillov, S., Hou, Y.-M., Cooperman, B.S.: Perturbation of the tRNA tertiary core differentially affects specific steps of the elongation cycle. Journal of Biological Chemistry **283**(26), 18431–18440 (2008) https://doi.org/10.1074/jbc.m801560200

[6] Nissen, P., Hansen, J., Ban, N., Moore, P.B., Steitz, T.A.: The structural basis of ribosome activity in peptide bond synthesis. Science **289**(5481), 920–930 (2000) https://doi.org/10.1126/science.289.5481.920

[7] Zhong, L., Wang, J.: Effective classification of microrna precursors using combinatorial feature mining and adaboost algorithms (2016)

[8] Macke, T.J.: RNAMotif, an RNA secondary structure definition and search algorithm. Nucleic Acids Research **29**(22), 4724–4735 (2001) https://doi.org/10.1093/nar/29.22.4724

[9] Staple, D.W., Butcher, S.E.: Pseudoknots: RNA structures with diverse functions. PLoS Biology **3**(6), 213 (2005) https://doi.org/10.1371/journal.pbio.0030213

[10] Hofacker, I.L., Fontana, W., Stadler, P.F., Bonhoeffer, L.S., Tacker, M., Schuster, P.: Fast folding and comparison of RNA secondary structures. Monatshefte fr Chemie Chemical Monthly **125**(2), 167–188 (1994) https://doi.org/10.1007/bf00818163

[11] Knudsen, B., Hein, J.: RNA secondary structure prediction using stochastic context-free grammars and evolutionary history. Bioinformatics **15**(6), 446–454 (1999) https://doi.org/10.1093/bioinformatics/15.6.446

[12] Sankoff, D.: Simultaneous solution of the RNA folding, alignment and protosequence problems. SIAM Journal on Applied Mathematics **45**(5), 810–825 (1985) https://doi.org/10.1137/0145048

[13] Allali, J., Sagot, M.-F.: A new distance for high level RNA secondary structure comparison. IEEE/ACM Transactions on Computational Biology and Bioinformatics **2**(1), 3–14 (2005) https://doi.org/10.1109/tcbb.2005.2

[14] Hu, Y.-J.: GPRM: a genetic programming approach to finding common RNA secondary structure elements. Nucleic Acids Research **31**(13), 3446–3449 (2003) https://doi.org/10.1093/nar/gkg521

[15] Fu, L., Cao, Y., Wu, J., Peng, Q., Nie, Q., Xie, X.: UFold: fast and accurate RNA secondary structure prediction with deep learning. Nucleic Acids Research **50**(3), 14–14 (2021) https://doi.org/10.1093/nar/gkab1074

[16] Sato, K., Akiyama, M., Sakakibara, Y.: RNA secondary structure prediction using deep learning with thermodynamic integration. Nature Communications **12**(1) (2021) https://doi.org/10.1038/s41467-021-21194-4

[17] Rivas, E., Eddy, S.R.: A dynamic programming algorithm for RNA structure prediction including

pseudoknots 1 1edited by i. tinoco. Journal of Molecular Biology **285**(5), 2053–2068 (1999) https://doi.org/10.1006/jmbi.1998.2436

[18] Yamashita, R., Nishio, M., Do, R.K.G., Togashi, K.: Convolutional neural networks: an overview and application in radiology. Insights into Imaging **9**(4), 611–629 (2018) https://doi.org/10.1007/s13244-018-0639-9

[19] Danaee, P., Rouches, M., Wiley, M., Deng, D., Huang, L., Hendrix, D.: bpRNA: large-scale automated annotation and analysis of RNA secondary structure. Nucleic Acids Research **46**(11), 5381–5394 (2018) https://doi.org/10.1093/nar/gky285