RESEARCH

State inference of RNA secondary structures with deep recurrent neural networks

Devin Willmott*, David Murrugarra and Qiang Ye

Abstract

Motivation: The problem of determining which nucleotides of an RNA sequence are paired or unpaired in the secondary structure of an RNA can be studied by different machine learning techniques. Successful state inference of RNA sequences can be used to get further insights into the RNA secondary structure. Typical tools for this task, such as hidden Markov models, exhibit poor performance in RNA state inference, owing in part to their inability to recognize nonlocal dependencies. Bidirectional long short-term memory (LSTM) neural networks have emerged as a powerful tool that can model global nonlinear sequence dependency and have achieved state-of-the-art performances on many different classification problems. This paper presents a method for RNA state inference centered around deep bidirectional LSTM networks.

Results: Our method, based on deep recurrent neural networks, achieves highly accurate state inference predictions and significantly outperforms the hidden Markov models on a set of RNA sequences that have a broad range of MFE accuracies as well as nonuniform patterns of paired and unpaired regions.

Availability: Python implementation of method available on Github at: https://github.com/dwillmott/lstm-rna.

Keywords: RNA; HMM; LSTM; RNN; binary classifier; secondary structure

1 Introduction

The secondary structure of an RNA sequence plays an important role in determining its function [8, 19], but directly observing RNA secondary structure is costly and difficult [3, 7]. Therefore, researchers have used computational tools to predict the secondary structure of RNAs. One of the most popular methods is the Nearest Neighbor Thermodynamic Model (NNTM) [30]. Alternatively, comparative sequence analysis methods [11] use a set of homologous sequences to infer a secondary structure [2]. This method remains the gold standard for secondary structure prediction [26].

NNTM is based on thermodynamic optimization to find the secondary structure with the minimum free energy (MFE). There are several implementations of NNTM; some of the popular ones include RNAStructure [22], GTfold [27], UNAfold [18]. However, NNTM has been shown to be ill-conditioned [14, 15]. That is, for a given sequence, significantly different secondary structures might exhibit very similar energies. Additionally, the range of accuracies of the predictions of NNTM shows significant variance [27].

More recently, high-throughput data that correlates with the state of a nucleotide being paired or unpaired has been developed. This data, called SHAPE [32] for 'selective 2'-hydroxyl acylation analyzed by primer extension', has been incorporated as auxiliary information into the objective function of NNTM with the goal of improving the accuracy of the predictions. This type of predictions are referred to as SHAPE-directed RNA secondary structure modeling [5, 31]. The addition of auxiliary information usually improves the accuracy of the predictions of NNTM [5] but it has been shown that the improvements are correlated with the MFE accuracy [26]. The latter claim has been done by statistical modeling of SHAPE. The model in [26] gives distributions for the values of SHAPE if the state of the nucleotide (as paired or unpaired or helix-end) is known. Thus the model in [26] can be used to generate SHAPE data in silico. A limitation of this model is that it requires knowing the states of the nucleotides.

In this paper, we study the problem of determining which nucleotides of an RNA sequence are paired or

*Correspondence: devin.willmott@uky.edu
Department of Mathematics, University of Kentucky, Lexington, KY 40506-0027, USA
Full list of author information is available at the end of the article

unpaired in the secondary structure as state inference using machine learning techniques. State inference is a binary classification task on each nucleotide, which we note is in contrast to full secondary structure inference, which seeks to identify sets of base pairs. We have developed a deep recurrent neural network that can classify the states of the nucleotides of an RNA sequence. The machine is a binary classifier that predicts if a given state is paired or unpaired in the secondary structure. The motivations for developing such a classifier are twofold: 1. A good classifier can be used to simulate auxiliary data using the statistical model of SHAPE presented in [26]; and 2. This binary classifier is a first step in an effort to use deep learning techniques to predict the actual base pairs in the secondary structure. We will explore this second goal in a future work.

A hidden Markov model (HMM) is a classical tool for the problem of state inference on a sequence. We can train an HMM using a set of RNA sequences with known secondary structure using maximum likelihood estimation. Once trained, we can use an HMM along with one of several prediction algorithms, such as the Viterbi algorithm [6], for classifying new RNA sequences' nucleotides.

The theoretical framework behind HMMs hinges on the Markov property, the assumption that state distribution is determined only by the state of the position immediately before it. It is only through this assumption that HMM inference algorithms become tractable. However, state inference for RNA is a fundamentally nonlocal problem: base pairs can form between nucleotides that are hundreds of positions away in the sequence. It would thus be desirable to use a method that can take into account information from much earlier or later in the sequence in making a prediction.

Since the advent of deep learning a decade ago, neural networks have become some of the most powerful tools available for classification problems in a variety of contexts [9,12,16]. Recurrent neural networks (RNNs) are designed specifically to deal with sequential data. The problem of learning long-term dependencies with RNNs has been studied in considerable depth by the machine learning community [9], and there exist a number of variants that have exhibited such capabilities. In this paper, we consider the most popular of these variants, called the Long Short-Term Memory (LSTM) architecture [13], which affixes a memory cell to each neuron that can remember inputs from previous timesteps and alter the output of the neuron.

We present an LSTM based method for RNA state inference. We compare performance of our method with that of a number of HMM variants. We find that our LSTM based method consistently achieves a state classification accuracy that reliably beats HMMs on average by 10%-15%. Our results also indicate some interesting connections between the performance of LSTMs and the distribution of the lengths of paired regions of RNA sequences. Such insights may be helpful in future design of neural networks for related classification problems for RNA sequences.

2 Methods

2.1 LSTMs

Neural networks are compositions of parameterized linear transformations and elementwise nonlinearity functions. In the standard framework, network parameters are trained using a dataset of known input-output pairs: the ubiquitous backpropagation algorithm [23] provides a gradient direction for the parameters with respect to a defined loss function, and we optimize parameters with respect to this loss using gradient descent.

This formulation require a fixed input and output size; in contrast, recurrent neural networks (RNNs) operate on sequences of variable length. Each sequence element is considered as a distinct input, and the same set of parameters is used as each sequence element is introduced to the machine. At each step, the RNN takes as input both the current sequence element and the previous step's output. The result is a sequence of machine predictions of the same length as the input sequence; we can compare these predictions to target outputs and optimize parameters using an analogous backpropagation algorithm for RNNs [20].

While RNNs theoretically possess the capacity to modeling long-term dependencies in sequential data, its implementation is difficult with learning algorithms often not converging due to vanishing or exploding gradients over a long duration [21]. The Long Short-Term Memory (LSTM) architecture is widely considered to be the most effective known solution to this problem, and has been responsible for most of the currently held records for benchmark sequential classification problems [17]. The key modification in an LSTM is the addition of a memory cell to each neuron that is updated each time data propagates forward through the network. Additional parameters and gates allow the machine to decide when to update or delete values in memory, and how memory values interact with inputs to produce the output. We train these additional parameters using the same backpropagation algorithm for traditional RNNs and, once trained, they allow the machine to remember events from many steps back in the sequence, change the outputs accordingly, and delete the memory that no longer aids in prediction.

For many sequences such as RNA sequences that we are interested here, there may be causal dependencies in both forward and backward time along the

sequences. Thus, a further modification of RNNs is to reverse the direction of half of the neurons in a layer, so that they receive input from the future timesteps in the sequence. Neurons in the next layer thus receive information from both directions in time. Such a network is referred to as bidirectional RNN, and is essential for tasks such as ours where outputs have dependencies in both the forward and backward directions [10, 24].

Our proposed method is thus a 3-layer bidirectional LSTM, with hidden layer sizes of 200 and 50. Our LSTM network takes an RNA sequence as an input and produces a sequence of the same length as an output where each element of the output sequence is the probability that the corresponding nucleotides belongs to one of the three classes: Paired, Unpaired, and End-of-Sequence (see the next paragraph for a description of end-of-sequence labels). At each timestep, the first hidden layer receives each nucleotide base in sequence, as well as hidden state information from earlier and later in the sequence. This information propagates forward through the network, and the resulting output of the last layer is the machine's state prediction for the nucleotide. We use RMSprop [29] as our training algorithm. RMSprop is a variant of gradient descent that keeps track of a decaying average of previous gradients to incorporate momentum and gradient normalization.

We implement this method using Keras [4], a deep learning API written in Python, with Theano [28] as a backend. Keras requires that LSTMs be processed in batches of sequences of equal length. Since our dataset contains a variety of sequence lengths, we append additional elements to smaller input and target output sequences to make all sequences have uniform size. These additional elements were labeled with a distinct class, which we call EOS (end-of-sequence). We found that this additional class did not affect the results of our binary classification: at the conclusion of training, our LSTM was successful at distinguishing EOS elements from actual paired/unpaired classes more than 99.99% of the time on both training, validation, and test sets.

We make a number of additional modifications to our machine's architecture, all of which are standard throughout the machine learning literature. We employ an L2-regularization term in the cost function and use the dropout training method [25], both of which are used to prevent the machine from overfitting its parameters to the training set. We also use a decaying learning rate, beginning at $\eta = 0.003$ and decaying by 3% at the conclusion of each epoch. This also helps to prevent overfitting, and forces the machine to converge to a single set of parameters by the time training has ended.

3 Method Comparison

3.1 HMMs

A hidden Markov model (HMM) is a traditionally popular method for modeling sequential data. However, we are unaware of work using HMMs or other methods in the literature for RNA state inference. We used our own implementation of HMM in Python and used this to compare HMM performance on state inference to that achieved by our LSTM based method. In our implementation of HMM, we treat the chain of nucleotides as the visible or input sequence, and the binary class of paired / unpaired as the state sequence. To train HMM parameters, we use maximum likelihood estimation on the same dataset we use to train our LSTMs. Once our parameters are found, we perform state inference with the Viterbi algorithm [6].

As evidence that our method surpasses straightforward variants of HMM, we also train and test several higher-order hidden Markov models [6]. An order k HMM uses the same algorithms for training and inference as standard HMMs, but makes uses the previous k states to make inferences; the standard HMM is thus an order 1 HMM. We found that increasing the order of the HMM to 2 resulted in a marked increase in performance, though no orders approached our method's performance.

3.2 Datasets

To train our models, we used secondary structure data from the Comparative RNA Web site, run by the Gutell Lab at the University of Texas [1]. This site hosts a collection of known RNA sequences and secondary structures obtained using comparative sequence analysis. Compiling all of the available 16S rRNA results in a set of 17032 sequences and a total of over 21 million nucleotides. We refer to this as the CRW dataset.

We also focus attention on the set of sixteen sequences examined in detail in [26], which we present as a test set for our method. These sequences exhibit a broad range of MFE accuracies; focusing on this set will allow us to find any relationships between MFE accuracies and the results of our methods. We will dedicate a portion of our later discussion to the relationships among state inference accuracy, MFE accuracy, and various sequence characteristics.

To ensure that our models, and in particular our LSTMs, do not simply memorize large portions of the test sequences, we removed CRW sequences with significant similarities to those in our validation set before training. In this filtering process, we compared each training set sequence against each test set sequence. If the two sequences have a common block of nucleotides of more than 10% of the length of the test sequence,

or if the two sequences can be aligned such that they have common blocks accounting for more than 80% of nucleotides of the shorter sequence, we remove it from the training set. (See the github code for more details.) This process leaves us with 13118 sequences and a total of approximately 16.5 million nucleotides. The set's mean and median sequence length is 1264 and 1431, respectively.

3.3 Model Selection

Both higher-order HMMs and LSTMs have hyperparameters that are best chosen using model validation techniques. For HMMs, this is the order of the model; for LSTMs, there are a variety of hyperparameters: network size and depth, learning rate and learning rate decay, L2-regularization coefficient, dropout rate, etc. We used half of the CRW dataset as a validation set to test models with different sets of hyperparameters. The unusually large size of the validation set was chosen due to the large degree of redundancy within the training set: since the dataset was built using comparative methods, there are many near-duplicate samples within the training set, and training and validation errors were nearly identical when we used a smaller proportion. The remaining half was used as the training set.

We note in section 5 that larger machines appear to be biased toward outputting negative predictions. Thus, during the validation process we looked for models that both minimized test set error and output false positives and false negatives in roughly equal proportion.

3.4 Metrics

Each machine outputs a predicted state sequence. In assessing our models, we consider each nucleotide as a separate classification problem, regarding paired elements as positive and unpaired elements as negative. We can thus categorize machine output as either true positive (TP), true negative (TN), false positive (FP), or false negative (FN). Our LSTM outputs a probability distribution for the state of each nucleotide, so we take the maximum probability to be the predicted state. EOS states (see Methods) are not considered in our results, and EOS predictions on paired or unpaired nucleotides are considered incorrect.

From these results, we generate three metrics that we focus on in our analysis of the results. The first, accuracy ($\text{Acc} = \frac{\text{TP}+\text{TN}}{\text{TP}+\text{TN}+\text{FP}+\text{FN}}$), is a simple measure of the proportion of correct predictions. We also look at positive predictive value ($\text{PPV} = \frac{\text{TP}}{\text{TP}+\text{FP}}$) and sensitivity ($\text{Sen} = \frac{\text{TP}}{\text{TP}+\text{FN}}$), which measure the proportion of true positives among positive predictions and true positives among positive states, respectively.

4 Results

Our results are based on using the CRW dataset to train our LSTM, as well as a number of higher order HMMs for comparison. Results for HMM orders 1 through 5 are compared against our method in Table 1. Though the table exhibits an upward trend in accuracy as the order of the HMM increases, we found that accuracy plateaued and eventually decreased beyond order 5. The order 4 HMM exhibits the highest accuracy on both validation and test sets; we will use this machine when investigating the differences between HMM and LSTM output in later sections.

The LSTM clearly outperforms HMMs of all orders on the validation set. More importantly, this is the case for our test set as well, where the LSTM outperforms the best HMM in accuracy by nearly 15%. This is also the case for PPV, but we note that the gap between the sensitivity of LSTM and HMM output is much smaller, suggesting that the LSTM is not a straightforward improvement on HMM predictions.

Considering machine predictions on each sequence gives some insight into these differences. Table 2 presents the testing results for each test set sequence and we arrange the test set in ascending order of MFE accuracy, as reported in [26]. We emphasize that MFE accuracy and LSTM/HMM accuracy are not directly comparable, as they refer to different problems (structure inference and state inference, respectively). However, presenting the sequences in this way expresses in some sense the difficulty current thermodynamic methods have in understanding the secondary structure of each sequence.

Neither machine's accuracy exhibited a strong relationship with MFE accuracy. HMM accuracy remained mostly the same for all sequences, regardless of MFE accuracy, while LSTM accuracy exhibited much more variance. However, the LSTM generally had very good performance on sequences with middling MFE accuracy, and poor performance on those with the highest and lowest MFE accuracies. LSTM results also had higher variance along every metric than HMMs, and

Table 1 A comparison of accuracy, PPV, and sensitivity of output from our LSTM, compared with those of HMMs of various orders. Training error and validation error remained very similar throughout training. The composition of both sets (validation and test) is described in Section 3.

Order 1 HMM	0.623	0.632	0.851	0.612	0.645	0.772
Order 2 HMM	0.662	0.671	0.826	0.651	0.688	0.758
Order 3 HMM	0.675	0.694	0.796	0.672	0.712	0.752
Order 4 HMM	0.686	0.714	0.773	0.684	0.731	0.744
Order 5 HMM	0.684	0.711	0.775	0.683	0.732	0.748
LSTM	0.971	0.971	0.981	0.820	0.863	0.828

Table 2 Table of accuracy, PPV, and sensitivity for our model vs. an order 4 HMM. Sequences are arranged in order of MFE accuracy as reported in [26], which is listed in the last column for reference.

Name	LSTM Acc	HMM Acc	LSTM PPV	HMM PPV	LSTM Sen	HMM Sen	MFE Accuracy
cuniculi	0.627	0.663	0.702	0.694	0.648	0.773	0.171
vnecatrix	0.600	0.600	0.678	0.690	0.598	0.573	0.181
celegans	0.554	0.597	0.566	0.631	0.624	0.549	0.203
nidulansM	0.601	0.584	0.670	0.683	0.621	0.535	0.272
tabacumC	0.923	0.701	0.937	0.731	0.933	0.786	0.310
cryptomonas	0.942	0.676	0.947	0.730	0.957	0.729	0.339
musM	0.568	0.603	0.595	0.655	0.565	0.520	0.375
gallisepticum	0.872	0.641	0.919	0.715	0.862	0.669	0.385
syne	0.950	0.700	0.958	0.739	0.958	0.772	0.388
ecoli	0.936	0.699	0.969	0.742	0.924	0.774	0.411
subtilis	0.974	0.699	0.978	0.733	0.979	0.788	0.512
desulfuricans	0.952	0.712	0.963	0.740	0.957	0.802	0.533
reinhardtiiC	0.911	0.687	0.934	0.725	0.916	0.761	0.537
maritima	0.943	0.753	0.959	0.762	0.946	0.863	0.562
tenax	0.748	0.782	0.831	0.786	0.744	0.891	0.618
volcanii	0.696	0.735	0.790	0.767	0.683	0.814	0.752
average	0.800	0.677	0.837	0.720	0.807	0.725	0.409
total	0.820	0.672	0.863	0.713	0.828	0.745	N/A

could be grouped into two clusters depending on accuracy: accuracy on each sequence was either above 0.85 or below 0.75.

Among those with poor performance, we note that there are four sequences (V. necatrix, C. elegans, M. nidulans, M. musculus) for which LSTM sensitivity is higher than HMM sensitivity, while the HMM's sensitivity is markedly better on the remaining three (E. cuniculi, T tenax, V. volcanii). This distinction can help us to better understand the difference between errors in LSTM and HMM predictions.

5 Discussion

Our metrics in Table 2 give us an idea of the proportion of correct machine predictions on each nucleotide's state, but they do not indicate whether HMM or LSTM predictions produce state sequences that preserve global properties, such as patterns of paired and unpaired states. In particular, we want the number and sizes of paired and unpaired regions of the state sequence prediction to match those in the original. A paired region in the state roughly corresponds to one half of a helix in the secondary structure, so it is vital that machines are able to emulate this property of the sequence, particularly if we wish to use the inferred state to generate experimental data, such as SHAPE.

We found evidence that the LSTM was more capable than the HMM of capturing this global structure, even in cases where HMM accuracy was higher than LSTM accuracy. We consider the distribution of paired regions in state sequences; Figure 1 shows boxplots of these distributions for native states and compares them to HMM and LSTM state predictions for our 16 test set sequences. A visual inspection of this figure shows that HMMs routinely produce paired regions that are significantly larger than those in the actual states. Indeed, when considering median paired region size of states, the HMM prediction is further from the actual state than the LSTM in all but one test set sequence (M. musculus), where all of the medians (native, HMM, and LSTM) are equal. We also note, however, that for sequences where the LSTM performs poorly, LSTM predictions tend to produce regions that are somewhat smaller than those in the actual state; this is readily seen in boxplots for C. elegans, E. nidulans, and E. cuniculi.

The LSTM was more capable of predicting the total number of regions in a state. Both machines produced predictions with fewer paired regions than the actual states, but LSTM predictions underestimated the number of paired regions in test set sequences, on average, by 19.7 (st dev 8.6) as compared to the HMM, which was off by 55.7 on average (st dev 11.8).

We note that this discrepancy is to be expected in the context of nonlocal interactions. Paired region size is exactly the sort of nonlocal feature that HMMs cannot predict: at a given time, the HMM does not know how long it has been outputting positive predictions, and is thus limited in its capacity to detect large paired regions.

Considering the non-locality of paired regions can also help to explain the poor performance of the LSTM

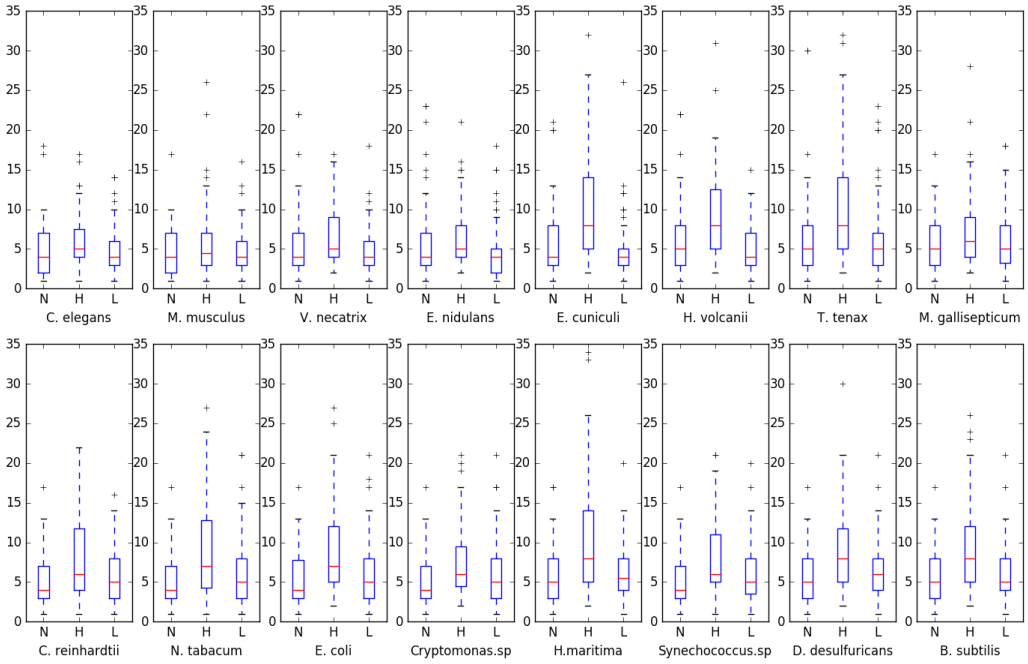

Figure 1 Boxplots of the distribution of sizes of paired regions in the native state sequence, HMM predicted state sequence, and LSTM predicted state sequence (denoted N, H, and L, respectively) for each test set sequence. The red line indicates the median region size, the box contains 25th-75th percentiles, and the whiskers contain 5th-95th percentiles. Sequences are ordered from lowest to highest LSTM prediction accuracy. Several large paired regions in HMM predictions beyond the y-axis limit of 35 are not shown.

on certain test set sequences. Successful LSTM predictions are often accompanied by a particular distribution of paired regions: one of length 17, one of length 13, and several more of length 12 and 11. In contrast, inspection of the native states in Figure 2 shows that the other sequences either have paired regions of length larger than 20 (E. cuniculi, V. necatrix, M. nidulans, tenax, and V. volcanii) or very few paired regions of length larger than 10 (C. elegans and M. musculus).

We can compare the distribution of the lengths of paired regions in each of our test sequences to the distribution in the training set. We find that the training set overwhelmingly contains sequences with paired region distributions similar to the test set sequences on which the LSTM performs well. In particular, we note that the training set has relatively few large paired regions: there are 5 regions of length 18, 2 regions of size 19, 4 regions of size 20, and none larger than 20. Thus, during training the machine is penalized for outputting more than 20 contiguous positive predictions. Consequently, from Figure 1 we can see that LSTM predictions do not create sufficiently large regions for a number of sequences, particularly those for which LSTM accuracy is poor.

We further consider the KL divergence of the distribution of the lengths of paired regions between the entire training set and each test set sequence. This measurement indicates how well the distribution of a test sequence may be learned from the training dataset. Figure 2 plots the LSTM accuracy and HMM accuracy for each testing sequence against its KL divergence. We can see that all of the sequences with poor LSTM performance diverge significantly in their distribution of paired regions. This may account for the relatively poor performance of LSTM predictions, as the plot indicates these sequences are outliers with respect to the training set. In contrast, HMM results do not appear to have much correlation with the KL divergence.

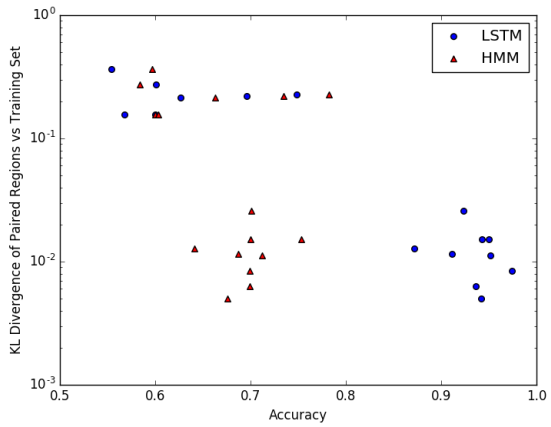

Figure 2 Plot comparing each test set sequence's LSTM and HMM accuracy vs its Kullback-Leibler divergence from training set paired region distribution. KL divergence was calculated as $\mathrm{KL}(P\|Q)$, where P is the test sequence distribution and Q is the training set distribution.

6 Conclusions & Future Work

We have presented a method for RNA state inference that uses long short-term memory networks to detect long-range interactions among nucleotides. Our results show an improvement of 10-15% in classification accuracy over HMM, a standard state inference tool, on a set of sequences that were distinct from our training data. We also present evidence that our method is better able to produce state sequence predictions that more closely resemble actual RNA states by investigating the distribution of paired regions in the output.

We emphasize that these results are presented primarily as a proof of concept. The efficacy of machine learning methods are necessarily limited by the dataset used to train the machine. In our case, we found that discrepancies between sizes of paired regions in the training and test sets accounted for much of the error in our test set. We conjecture that a training set with more diversity in its outputs will generalize better to RNA sequences with other pairing structures.

Our work is the first of which we are aware of that uses neural networks for RNA state inference. The results demonstrate that neural networks are capable of capturing nonlocal interactions of RNA sequences. An extension of this neural network based approach could be a promising method for the more general problem of secondary structure inference.

Competing interests
The authors declare that they have no competing interests.

Author's contributions
DW, DM, and QY conceived the study. DW designed code and performed the numerical experiments and theoretical analysis. All authors helped in the writing of the manuscript. All authors approved the final version of the manuscript.

Acknowledgements
We would like to thank John Hirdt, who had initially participated in this project, for many valuable discussions and suggestions.

Funding
Research of Qiang Ye was supported in part by NSF under Grants DMS-1317424, DMS-1318633 and DMS-1620082.

References

1. Jamie J Cannone, Sankar Subramanian, Murray N Schnare, James R Collett, Lisa M D'Souza, Yushi Du, Brian Feng, Nan Lin, Lakshmi V Madabusi, Kirsten M Müller, et al. The comparative rna web (crw) site: an online database of comparative sequence and structure information for ribosomal, intron, and other rnas. *BMC bioinformatics*, 3(1):2, 2002.

2. Jamie J. Cannone, Sankar Subramanian, Murray N. Schnare, James R. Collett, Lisa M. D'Souza, Yushi Du, Brian Feng, Nan Lin, Lakshmi V. Madabusi, Kirsten M. Müller, Nupur Pande, Zhidi Shang, Nan Yu, and Robin R. Gutell. The comparative rna web (crw) site: an online database of comparative sequence and structure information for ribosomal, intron, and other rnas. *BMC Bioinformatics*, 3(1):2, 2002.
3. Jonathan L Chen, Stanislav Bellaousov, and Douglas H Turner. Rna secondary structure determination by nmr. *Methods Mol Biol*, 1490:177–86, 2016.
4. François Chollet et al. Keras, 2015.
5. Katherine E Deigan, Tian W Li, David H Mathews, and Kevin M Weeks. Accurate shape-directed rna structure determination. *Proc Natl Acad Sci U S A*, 106(1):97–102, Jan 2009.
6. Richard Durbin, Sean R. Eddy, Anders Krogh, and Graeme Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.
7. Boris Fürtig, Christian Richter, Jens Wöhnert, and Harald Schwalbe. Nmr spectroscopy of rna. *ChemBioChem*, 4(10):936–962, 2003.
8. Paul P Gardner and Robert Giegerich. A comprehensive comparison of comparative rna structure prediction approaches. *BMC Bioinformatics*, 5:140, Sep 2004.
9. Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.
10. Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610, 2005.
11. Robin R Gutell, Jung C Lee, and Jamie J Cannone. The accuracy of ribosomal rna comparative structure models. *Curr Opin Struct Biol*, 12(3):301–10, Jun 2002.
12. Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.
13. Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.
14. D M Layton and R Bundschuh. A statistical analysis of rna folding algorithms through thermodynamic parameter perturbation. *Nucleic Acids Res*, 33(2):519–24, 2005.
15. S Y Le, J H Chen, and J V Maizel, Jr. Prediction of alternative rna secondary structures based on fluctuating thermodynamic parameters. *Nucleic Acids Res*, 21(9):2173–8, May 1993.
16. Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
17. Zachary C Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.
18. Nicholas R Markham and Michael Zuker. Unafold: software for nucleic acid folding and hybridization. *Methods Mol Biol*, 453:3–31, 2008.
19. David H Mathews and Douglas H Turner. Prediction of rna secondary structure by free energy minimization. *Curr Opin Struct Biol*, 16(3):270–8, Jun 2006.
20. Michael C Mozer. A focused back-propagation algorithm for temporal pattern recognition. *Complex systems*, 3(4):349–381, 1989.
21. Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. *ICML (3)*, 28:1310–1318, 2013.
22. Jessica S Reuter and David H Mathews. Rnastructure: software for rna secondary structure prediction and analysis. *BMC Bioinformatics*, 11:129, 2010.
23. David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1.
24. Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.
25. Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
26. Zsuzsanna Sükösd, M Shel Swenson, Jørgen Kjems, and Christine E Heitsch. Evaluating the accuracy of shape-directed rna secondary structure predictions. *Nucleic Acids Res*, 41(5):2807–16, Mar 2013.
27. M Shel Swenson, Joshua Anderson, Andrew Ash, Prashant Gaurav, Zsuzsanna Sukosd, David A Bader, Stephen C Harvey, and Christine E Heitsch. Gtfold: Enabling parallel rna secondary structure prediction on multi-core desktops. *BMC Res Notes*, 5(1):341, Jul 2012.
28. Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.
29. Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2), 2012.
30. Douglas H Turner and David H Mathews. Nndb: the nearest neighbor parameter database for predicting stability of nucleic acid secondary structure. *Nucleic Acids Res*, 38(Database issue):D280–2, Jan 2010.
31. Stefan Washietl, Ivo L Hofacker, Peter F Stadler, and Manolis Kellis. Rna folding with soft constraints: reconciliation of probing data and thermodynamic secondary structure prediction. *Nucleic Acids Res*, 40(10):4261–72, May 2012.
32. Kevin A Wilkinson, Robert J Gorelick, Suzy M Vasa, Nicolas Guex, Alan Rein, David H Mathews, Morgan C Giddings, and Kevin M Weeks. High-throughput shape analysis reveals structures in hiv-1 genomic rna strongly conserved across distinct biological states. *PLoS Biol*, 6(4):e96, Apr 2008.