RESEARCH

State inference of RNA secondary structures with deep recurrent neural networks

Devin Willmott*, David Murrugarra and Qiang Ye

Abstract

Motivation: Secondary structure inference is etc. The related problem of state inference can be used to gain 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. Long short-term memory (LSTM) neural networks have emerged as powerful machine learning tools that can model global nonlinear sequence dependency and have achieved state-of-the-art performances on various sequential learning tasks. This work presents a method for RNA state inference centered around deep bidirectional LSTM networks. This method achieves highly accurate state inference predictions and significantly outperforms the hidden Markov models on a set of 16S rRNA sequences with a broad range of MFE accuracies. We also show that LSTM state predictions exhibit similar global pairing patterns to native sequence states.

Keywords: secondary structure; RNA folding; neural networks; recurrent neural networks; LSTM; machine learning

1 Introduction

The secondary structure of an RNA sequence plays an important role in determining its function [9, 21], but directly observing RNA secondary structure is costly and difficult [3, 8]. 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) [34]. Alternatively, comparative sequence analysis methods [12] use a set of homologous sequences to infer a secondary structure [2]. This method remains the gold standard for secondary structure prediction [29].

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 [24], GTfold [30], UNAfold [20], ViennaRNA package [19]. However, NNTM has been shown to be ill-conditioned [15,16,25]. 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 [30].

*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

More recently, high-throughput data that correlates with the state of a nucleotide being paired or unpaired has been developed. This data, called SHAPE [36] 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 prediction is referred to as SHAPE-directed RNA secondary structure modeling [5, 35]. 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 [29]. The latter result has been obtained by statistical modeling of SHAPE. The model in [29] 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 [29] 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 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 se-

quence. 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 the following: first, a good classifier can be used as a restraint on the secondary structure prediction via NNTM; second, this classifier can also help in predicting biomolecular structures such as for identifying binding sites in RNA-RNA interactions [6, 31]; and finally, a good classifier can be used to simulate auxiliary data using the statistical model of SHAPE presented in [29].

There are several classical tools arising from stochastic grammars that can be applied to the problem of state inference on a sequence. Two such tools are hidden Markov models (HMMs) and stochastic context-free grammars (SCFGs). We can train these models with a set of RNA sequences with known secondary structures using maximum likelihood estimation. Once trained, HMMs and SCFGs can be used along with various prediction algorithms, such as the Viterbi and CYK algorithms, respectively [7], to classifying new RNA sequences' nucleotides.

Due to the cubic complexity of SCFGs with respect to the length of the RNA sequence [7], this work will focus on HMMs as a benchmark model. HMMs achieve linear complexity through the Markov property, the assumption that state distribution is determined only by the state of the position immediately before it. This linear complexity makes it an attractive option for state inference on long RNA sequences, such as 16S rRNA sequences. 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 [10, 13, 17]. 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 [10], 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 [14], 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, and compare with a number of HMM variants on the same dataset. We find that our LSTM based method consistently achieves a state classification accuracy that reliably beats HMMs on a test set of 16S rRNA sequences on average by 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 Neural Networks

A neural network is a function composed of a parametrized affine transformation and an elementwise nonlinearity. Network parameters are trained using a dataset of known input-output pairs: we define a loss function based on the difference between machine predictions and target outputs, retrieve gradient directions for parameters with respect to this loss using the ubiquitous backpropagation algorithm [26], and optimize parameters using first order methods, such as gradient descent.

Recent advances in machine learning come primarily from deep neural networks, which are stacks of multiple neural networks: the output of one neural network in the stack acts as the input for the next. Each of these constituent neural networks is referred to as a layer of the deep neural network; layers between the input and output and called hidden layers. The hidden layers of a deep neural network allow it to represent complex nonlinear relationships among inputs.

Typically, deep neural networks require a fixed input and output size. This is undesirable for our purposes, as we are interested in performing state inference on RNA sequences of various lengths. Two major variants of neural networks that can handle variable input and output sizes are recurrent neural networks and convolutional neural networks, both of which we make use of in our proposed method.

2.2 Recurrent Neural Networks & LSTMs

Recurrent neural networks (RNNs) work specifically with sequential data by combining the learning methods of neural networks with the architecture of a discrete-time dynamical system. Sequence elements are fed to the machine one at a time, and machine parameters act on both the current step's input and the previous step's output to produce the current step's output. These parameters act similarly to the input and state matrices in a dynamical system, and are trained with a variant of the backpropagation algorithm for RNNs [22].

In RNA sequences, there exist causal dependencies both forward and backward in time along the sequence.

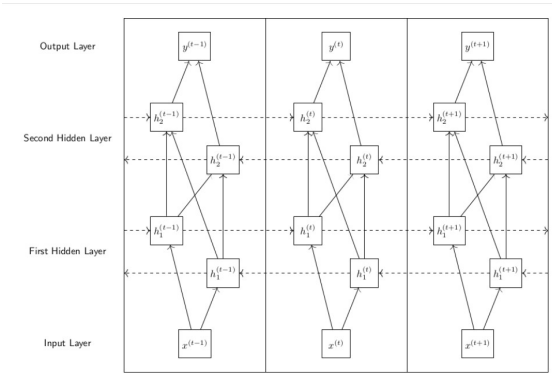

Figure 1 Diagram of a bidirectional RNN with two hidden layers for state inference at three different timesteps. Here x denotes the input, h_1 the first layer hidden variable, h_2 the second layer hidden variable, and y the output, with superscripts representing timesteps and arrows showing the propagation of information through the network. Each hidden layer combines information from the previous layer, earlier and later timesteps, and an internal memory state to compute its output.

A common modification of RNNs in this instance is to reverse the direction of half of the neurons in a layer, so that they receive input from the next timesteps in the sequence. The next hidden layer of the network thus receives information from both directions in time. Such a network is referred to as bidirectional RNN, and they are known to significantly outperform RNNs on tasks such as ours where outputs have dependencies in both the forward and backward directions [11,27]. The network shown in Figure 1 is a deep bidirectional RNN with two hidden layers, where each of these layers uses hidden states from previous and future timesteps to compute the current timestep's output.

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 [23]. The Long Short-Term Memory (LSTM) architecture is a variant of RNN that is widely considered to be the most effective known solution to this problem; it is responsible for most of the currently held records for benchmark sequential classification problems [18]. 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. Once trained, these parameters allow an LSTM to remember events from many steps away in the sequence, change the outputs accordingly, and delete the memory that no longer aids in prediction. Bidirectional LSTMs will form the foundation of our network.

2.3 Convolutional Neural Networks

Just as RNNs learn the parameters of a dynamical system, convolutional neural networks (CNNs) learn small kernels that are convolved with input data to produce an output of the same dimensionality. Most often, CNNs find use in two-dimensional problems such as image classification and segmentation, where they act similarly to image processing filters [10], but they can be used in any problems with a defined spatial structure. Since we are dealing with sequential data, our machine will use one-dimensional convolutions.

In many applications, convolutional layers in a deep neural network are followed by pooling layers that reduce the size of the layer's output. We omit this step in our machine to preserve sequence length while the data passes through the network, as we will interpret our machine's output as a sequence of nucleotide state predictions.

Convolutional layers are inherently unable to learn long-term dependencies, as they act only on small regions of input and, unlike RNNS, do not use outputs at other timesteps. However, they are computationally cheap and invariant to translation. We add convolutional layers as the first and last layers of our machine; these layers are able to detect local features and allow the LSTMs to focus on learning long-term dependencies. We can compare the role of convolutional layers in our machine to the parameters in energy minimization methods for secondary structure prediction. In that framework, each potential secondary structure is assigned a probability based on the presence of various local features of the RNA sequence. In our machine, optimizing convolutional layer parameters will identify those local features that are relevant to state inference in our method.

2.4 State Inference with Convolutional LSTMs

We tested a variety of deep networks composed of recurrent and convolutional layers, and found a four-layer network to be the optimal balance of representational capacity and training speed. Our network first applies a one-dimensional convolution to the input RNA sequence. The output of this convolution is fed in to two layers of bidirectional LSTMs; information flows through these layers as detailed in Section 2.2 and Figure 1. It then finishes with another convolutional layer, which combines information in small regions in the bidirectional LSTM output. The output of this final convolutional layer is a sequence of the same length as the input, where each sequence element is the machine's state prediction for its corresponding

nucleotide, represented as a probability that the nucleotide is paired or unpaired.

The output dimension of each of the machine's four layers at each timestep are 100, 400, 100, and 2, respectively. Both convolutions have a size of 30 nucleotides, and a stride of 1. These specifications give the machine a total of 595,552 trained parameters. We use RMSprop [33] with a learning rate of $\eta = 0.0001$ 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 throughout training.

We implement this model using Keras [4], a deep learning API written in Python, with Theano [32] as a backend. Keras requires that LSTMs be processed in batches of sequences of equal length. Since our datasets contain 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 are not labeled with a target output class, and thus predictions on these elements do not affect the loss function, and are ignored when interpreting the results.

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 L1-regularization term in the loss function and use the dropout training method [28], both of which are used to prevent the machine from overfitting its parameters to the training set.

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 in the literature for RNA state inference. We used our own implementation of HMM in Python, where we treat the chain of nucleotides as the visible or input sequence, and the binary class of paired/unpaired as the state sequence. We use maximum likelihood estimation on a dataset of known input and state sequences to train our parameters, and then perform state inference with the Viterbi algorithm [7]. This is a backtracking algorithm that finds the maximum likelihood state sequence given an input sequence. We trained the HMM and our LSTM-based method on the same datasets, and we compare their performance using their state predictions.

As evidence that our method surpasses straightforward variants of HMM, we also train and test several higher-order hidden Markov models [7]. 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 and validate 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 sequences from this database 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 [29], 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 detect 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 test set before training. In this filtering process, we compared each CRW 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 75% 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, L1-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 binary classification problem, regarding paired elements as positive and unpaired elements as negative. We can thus categorize each element of the 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.

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

4 Results

Table 1 Accuracy of LSTM vs. HMM on validation and test sets

	Validation Set			Test Set		
Machine	Acc	PPV	Sen	Acc	PPV	Sen
Order 1 HMM	0.623	0.632	0.852	0.612	0.646	0.767
Order 2 HMM	0.662	0.671	0.826	0.651	0.686	0.759
Order 3 HMM	0.674	0.693	0.794	0.672	0.713	0.750
Order 4 HMM	0.685	0.714	0.771	0.684	0.729	0.742
Order 5 HMM	0.684	0.711	0.776	0.683	0.730	0.742
LSTM	0.954	0.950	0.972	0.839	0.858	0.873

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 model when investigating the differences between HMM and LSTM output in detail.

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 more than 15 percentage points. 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 metrics on the order 4 HMM and LSTM state predictions of each test set sequence, where the LSTM outperforms the HMM on all but one test set sequence (C. elegans). Sequences are arranged in Table 2 in ascending order of MFE accuracy as reported in [29]. MFE accuracy and LSTM/HMM accuracy are not directly comparable, as they refer to different problems (structure inference and state inference, respectively), but 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 state inference 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. The LSTM generally had very good performance on sequences with middling MFE accuracies, and relatively poor performance on those with the highest and lowest MFE accuracies. LSTM results also had higher variance along every metric than HMM results, and test set sequences can be grouped into two clusters depending on LSTM state predictions: accuracy was above 0.9 for nine sequences, and near or below 0.8 for the remaining seven sequences.

Among those in the latter cluster, we note that there are four sequences (V. necatrix, C. elegans, M. nidulans, M. musculus) for which LSTM sensitivity is much higher than HMM sensitivity, while this gap is much smaller for 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

5.1 Paired Regions & Global Structure

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

Table 2 Detailed results of LSTM vs. HMM on the test set

Name	LSTM Acc	HMM Acc	LSTM PPV	HMM PPV	LSTM Sen	HMM Sen
E. cuniculi	0.680	0.661	0.713	0.693	0.774	0.773
Vairimorpha necatrix	0.661	0.600	0.721	0.689	0.683	0.576
C. elegans	0.558	0.584	0.570	0.613	0.624	0.552
Emericella nidulans	0.657	0.584	0.692	0.681	0.741	0.539
Nicotiana tabacum	0.913	0.705	0.917	0.734	0.938	0.787
Cryptomonas.sp	0.926	0.676	0.935	0.730	0.941	0.728
M. musculus	0.608	0.603	0.626	0.655	0.637	0.520
Mycoplasma gallisepticum	0.919	0.639	0.933	0.713	0.932	0.668
Synechococcus.sp	0.938	0.700	0.943	0.740	0.953	0.769
E. coli	0.924	0.699	0.937	0.742	0.938	0.774
Bacillus subtilis	0.973	0.698	0.979	0.731	0.976	0.788
Desulfovibrio desulfuricans	0.926	0.712	0.940	0.741	0.938	0.803
Chlamydomonas reinhardtii	0.906	0.687	0.915	0.725	0.928	0.761
Thermotoga maritima	0.931	0.752	0.944	0.760	0.943	0.864
Thermoproteus tenax	0.818	0.782	0.845	0.785	0.866	0.894
H. volcanii	0.782	0.739	0.809	0.769	0.841	0.820
Average	0.820	0.676	0.839	0.719	0.853	0.726
Total	0.839	0.684	0.858	0.729	0.873	0.742

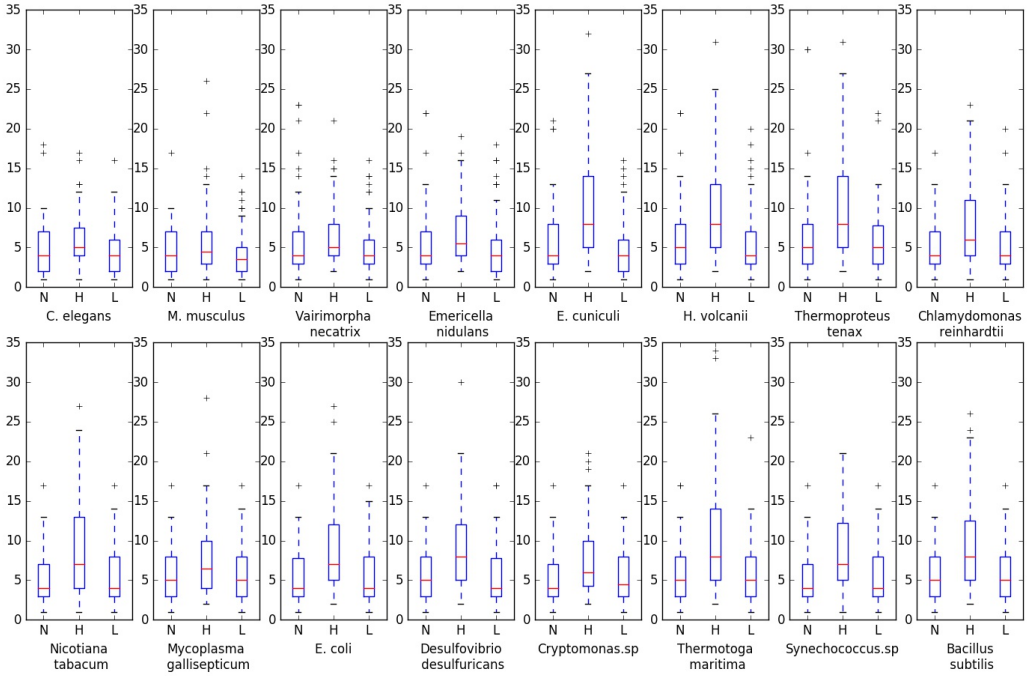

Figure 2 Boxplots of the distribution of sizes of paired regions in the native state sequence, HMM predicted state sequence, and LSTM predicted state sequence (denoted N, H, and L, respectively) for each test set sequence. The red line indicates the median region size, the box contains 25th-75th percentiles, 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.

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 when HMM accuracy was higher than LSTM accuracy. We consider the distribution of paired regions in state sequences; Figure 2 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 each state, the LSTM prediction is more accurate than the HMM prediction in all but one test set sequence (M. musculus). 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 M. musculus, E. nidulans, and E. cuniculi.

We can also consider the total number of paired regions in the state as another global feature of RNA state. The LSTM performs better in this case as well, producing predictions that, on average, had 6 more paired regions than the native state. However, HMM predictions vastly underestimated this number; these predictions had an average of 57 fewer regions than the native state.

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 on certain test set sequences. High LSTM accuracy is 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 2 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 paired region lengths 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 3 plots the LSTM accuracy and HMM accuracy for each test set 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.

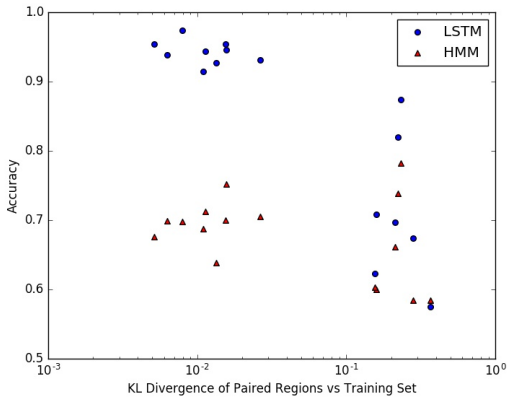

Figure 3 Plot comparing each test set sequence's LSTM and HMM accuracy vs its Kullback-Leibler divergence from training set paired region distribution. KL divergence was calculated as $\mathrm{KL}(P\|Q)$, where P is the test sequence distribution and Q is the training set distribution.

5.2 5S and 23S rRNA

Our training and test sets focus only on 16S rRNA. This is due to the unusually large amount of correctly identified structure available in this case; as noted in [5], 16S rRNA is the most comprehensive data set of correctly predicted RNA secondary structure available. There is limited data available for other sequence types, such as 5S and 23S rRNA. This is insufficient for training neural networks, which often require many thousands of training examples to generalize to new samples. We tested both the HMM and

LSTM machines on 5S and 23S rRNA datasets after being trained on 16S sequences, and found that the LSTM was able to make only slight improvements on HMM predictions on these sets. This indicates that 5S and 23S sequences are sufficiently different from 16S sequences to require their own examples for training.

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 15 percentage points in classification accuracy over an 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, and found that training only on 16S data was insufficient for achieving high accuracy on 5S and 23S rRNA sequences. Effectively incorporating RNA sequences with diverse sequence types and structures is a possible future direction for this work.

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.

Acknowledgements

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

Funding

This work was supported by the National Science Foundation [DMS-1317424, DMS-1318633, DMS-1620082].

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 statistical analysis. All authors helped in the writing of the manuscript. All authors approved the final version of the manuscript.

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. Laura DiChiacchio, Michael F Sloma, and David H Mathews. Accessfold: predicting rna–rna interactions with consideration for competing self-structure. *Bioinformatics*, 32(7):1033–1039, 2015.
7. Richard Durbin, Sean R. Eddy, Anders Krogh, and Graeme Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.
8. Boris Fürtig, Christian Richter, Jens Wöhnert, and Harald Schwalbe. Nmr spectroscopy of rna. *ChemBioChem*, 4(10):936–962, 2003.
9. Paul P Gardner and Robert Giegerich. A comprehensive comparison of comparative rna structure prediction approaches. *BMC Bioinformatics*, 5:140, Sep 2004.
10. Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.
11. Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610, 2005.
12. 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.
13. Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.
14. Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.
15. 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.
16. 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.
17. Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
18. Zachary C Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.
19. Ronny Lorenz, Stephan H Bernhart, Christian Höner Zu Siederdissen, Hakim Tafer, Christoph Flamm, Peter F Stadler, and Ivo L Hofacker. Viennarna package 2.0. *Algorithms Mol Biol*, 6:26, Nov 2011.
20. Nicholas R Markham and Michael Zuker. Unafold: software for nucleic acid folding and hybridization. *Methods Mol Biol*, 453:3–31, 2008.
21. 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.

22. Michael C Mozer. A focused back-propagation algorithm for temporal pattern recognition. *Complex systems*, 3(4):349–381, 1989.
23. Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. *ICML (3)*, 28:1310–1318, 2013.
24. Jessica S Reuter and David H Mathews. Rnastructure: software for rna secondary structure prediction and analysis. *BMC Bioinformatics*, 11:129, 2010.
25. Emily Rogers, David Murrugarra, and Christine Heitsch. Conditioning and robustness of rna boltzmann sampling under thermodynamic parameter perturbations. *Biophysical Journal*, 113(2):321–329, 2017.
26. David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.
27. Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.
28. 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.
29. 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.
30. 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.
31. Hakim Tafer, Fabian Amman, Florian Eggenhofer, Peter F Stadler, and Ivo L Hofacker. Fast accessibility-based prediction of rna-rna interactions. *Bioinformatics*, 27(14):1934–40, Jul 2011.
32. Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.
33. 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.
34. 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.
35. 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.
36. 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.