Exploring Neural Network Models for LncRNA Sequence Identification

Jason Rafe Miller
Shepherd University
Shepherdstown WV USA
jmil02@shepherd.edu

Donald A. Adjeroh
West Virginia University
Morgantown WV USA
donald.adjeroh@mail.wvu.edu

ABSTRACT
Distinguishing long non-coding RNA from protein-coding RNA is important to molecular and cellular biology. The problem can be addressed with machine learning in general and with artificial neural networks in particular. We explore the effects of various network design choices on the accuracy of human LncRNA identification. Perceptron-based neural network models were found to be almost as accurate as more complex recurrent neural networks, and K-mer representations of the data seemed to assist both. Size selection of training data affected results. These explorations could assist in neural network design for RNA analysis.

1 INTRODUCTION

The discovery that the human genome is extensively transcribed [1–3], with a majority of its bases being involved in some primary transcripts, including transcripts from both protein-coding and non-protein coding regions, changed traditional views of gene regulation and gene function. A significant portion of transcription is now known to involve long non-coding ribonucleic acids (LncRNAs). LncRNAs are known to be involved in gene regulation, gene imprinting, chromatin remodeling, and embryonic development (see [4–6] for reviews), and in diseases such as cancer, neurodegenerative diseases, and heart disease (see [7, 8]). Some individual LncRNAs are well-studied (*e.g.* HOTAIR [8], MALAT-1 [9], Xist [10]) but understanding of the nature and functions of LncRNAs remains incomplete.

Genome databases such as Ensembl [11] and GenCode [2, 12, 13] catalog tens of thousands of protein-coding genes from human and other species. They also house DNA gene sequences and RNA transcript sequences for many LncRNA, which are defined as transcribed sequences with minimum length 200 unassociated with any protein product. In many cases, the LncRNA designation is based on bioinformatic analysis of the sequence itself. Several computer programs exist to identify LncRNA (reviewed in [14]) but the task remains challenging, even within the extensively studied human genome [15].

Artificial neural networks (ANNs) provide an attractive software model for approaching the RNA classification problem. In contrast to expert systems that incorporate domain expertise, ANNs are generic learners capable of being trained with labeled data. Given the labeled RNA sequences from public databases, ANNs infer distinguishing features that they use in combination to classify other, unlabeled RNA sequences. For example, the LncADeep [16] software package includes ANNs trained on either full-length or partial transcript sequences plus their annotation *i.e.* features besides the primary sequence, whether experimentally derived or computationally predicted, that are also available in public databases. Many of the annotated features used by LncADeep were measures of protein-coding potential *i.e.* indicators that an RNA is mRNA not LncRNA.

In contrast, the classifier by Hill et al. [17] was trained on primary sequence alone without reliance on the annotation, which could be erroneous or biased. Trained and tested on labeled, full-length, human sequence data from GenCode, the classifier achieved 95% accuracy. Perturbation analysis indicated the trained model relied heavily on the coding regions of protein-coding mRNA, as LncADeep had been programmed to do. Some non-ANN methods

978-1-7281-6215-7/20/$31.00 ©2020 IEEE

applied to LncRNA identification include alignment based approaches [18], logistic regression [19], support vector machine [20], and random forest [21].

Some bioinformatics software goes beyond the problem of LncRNA identification. The DeepLncRNA [22] software uses an ANN to predict the subcellular localization of LncRNA. In contrast, SEEKR [23] is non-ANN software that predicts LncRNA cellular function. SEEKR suggests function for uncharacterized LncRNAs based on correlations to well-characterized LncRNAs. The correlations are computed from K-mer profiles, which are histograms containing the frequencies of each contiguous substring of length K. (For example, at K=2, the sequence 'AAACGT' has one K-mer with frequency ⅖, namely 'AA'. Like the RNA in most databases, this example uses the 4-nucleotide alphabet of DNA.)

We consider the problem of ANN-based LncRNA identification with the hope of gaining insights for ANN application to other questions of LncRNA biology. We build on the approach of Hill *et al.* [17] in which a variety of ANN designs were tested on the way to building one successful classifier. Although ANNs are trained automatically, they are typically designed by investigators who select the overall architecture as well as a large number of hyperparameters. It is therefore important to understand whether certain architectures are most applicable to the problem domain.

We focus on two ANN architectures. The most basic architecture is the multilayer perceptron (MLP) [24, 25] which consists of components called neurons organized into an ordered list of layers. The neurons are fully-connected between successive layers but not within layers. The connections receive initial weights which get adjusted during training. While this architecture is capable of machine learning, it is not tailored for any specific task. In contrast, the recurrent neural network (RNN) [26, 27] incorporates the notion of a time series. Whereas an MLP would process an RNA as one bag of features, an RNN could process it as a time-ordered sequence of nucleotides. Two RNN variants enhance the long-term memory of the basic RNN design: the long short-term memory (LSTM) model [28] and the gated recurrent unit (GRU) model [29]. Training these models involves automatically adjusting weights that control the portion of past information that is carried forward.

Using both the MLP and RNN architectures, we explore hyperparameters such as number of layers and number of neurons per layer. Our study is similar to parts of Hill *et al.* [17] and it differs in the following ways. We explore the MLP architecture in addition to the RNN. We experiment with preprocessing that presents K-mers rather than nucleotides to the model; this extends the analysis of the initial embedding layer included in the RNN. We test RNNs with more than one recurrent layer and we test LSTM and bidirectional GRU in addition to GRU layers. We implemented our models with the open-source Keras and TensorFlow software libraries [30]. Unlike Hill *et al.*, we have not explored data augmentation, ensemble methods, or permutation analysis, and we have not built or tested a final model for LncRNA identification.

Sequence length is a defining characteristic of long non-coding RNAs, and length may be characteristic of certain functional roles [31, 32], a fact that has been exploited previously to predict LncRNA protein interaction [33]. Whereas Hill *et al.* trained models on only sequences of lengths 200 to 1000, and tested only on sequences of all lengths, we explored the effect of training and testing on different length-based subsets of the data. We hope that some of our results would assist in neural network design for future RNA studies.

2 METHODS:

2.1 Data Preparation

GenCode release 34 data was preprocessed using our custom script (preprocess2.py) that roughly follows the GenCode release 25 procedure in Hill *et al.* [17]. Coding and non-coding sets were filtered to remove short (<200) sequences and sequences with letters other than {A,C,G,T}. From the remaining data, the median-length transcript per gene was identified as a candidate, and 16K coding and 16K non-coding candidates were selected for training and validation; the remainder was held for testing in future work.

Except for the length-effect study, only sequences of length 200 to 1000 were used as

Hill *et al.* trained on this subset exclusively. This subset was imbalanced, being 64% non-coding (**Table I**). Prior to every experiment, a random 10% of the data was set aside for validation with the rest used for training. Every sequence was 'N' padded to the maximum length of the subset. For embeddings, any K-mer containing an 'N' was encoded as zero and masked. For K-mer profiles, any K-mer with at least one 'N' counted as the Kmer with 100% 'N'.

Table I. Data set preprocessing

GenCode34	# Coding	# Non-coding	Total
Original	100566	48479	149045
Filtered	95319	45597	140916
Candidates	19170	16742	35912
Selected	16000	16000	32000
200-1Kbp	5781	10323	16104

2.2 Neural Network Implementation

Machine learning programs were developed in Python 3.8 using numpy 1.18.4, pandas 1.0.5, matplotlib 3.2.2, sklearn 0.23.1, tensorflow 2.2.0, and keras 2.4.3. All models used the Keras Sequential API [30] plus combinations of the Bidirectional, GRU, LSTM, Dense, and Dropout [34] layers. All layers used Keras default values unless otherwise noted. Training used mini-batch gradient descent with the default batch size of 16, binary cross entropy loss, and the Adam [35] optimizer, unless otherwise noted. Models were run in Jupyter 6.1.4 notebooks on Google CoLab virtual machines using its standard GPU configuration: K80, 12 GB RAM. Models were trained and evaluated using 5-fold cross validation with 200 epochs per fold. The highest validation accuracy per fold was recorded, and the mean and standard deviation of these five numbers was reported. Unless otherwise noted, each fold randomly partitioned the sequences into 90% training and 10% validation. Statistics were computed with Scipy 1.5.3: Pearson correlation or Student two-tailed t-test at $p<0.05$ assuming independent samples but not assuming equal variance.

The MLP model design was selected after two preliminary explorations. With K values 1 through 5, a 2-layer, 16-neuron model (2x16) outperformed simpler models and about as well as more complex models. Then, with the 2x16 model and K=3, a search over various hyperparameters led to these choices: activation = *elu*, regularizer = adam with the following learning rate decay schedule: initial learn rate = 0.01 (10 times the default), decay rate = 0.99, decay steps = 10K, and staircase = true. For the RNN study, the basic design was inspired by Hill *et al.*, which settled on a 128-dimension embedding layer, a 1x32 GRU layer and a 32-neuron FC layer, with 40% dropout. Source code is available online at https://github.com/ShepherdCode/IEEE.BIBM.2020.

3 RESULTS

3.1 High Accuracy with a Basic ANN

We developed a basic ANN using the MLP architecture and hyperparameters for 2 layers and 16 neurons per layer (2x16 MLP). The inputs were K-mer profiles containing either single-nucleotide frequencies or 3-nucleotide frequencies (*i.e.* K-mers with K=1 or 3, stride=1). The K-mer profiles, including one K-mer of 'N's, consist of $d=4^K+1$ frequencies, *i.e.* d=5 at K=1 and d=65 at K=3. The model incorporates a weight on each connection from every profile dimension to all 16 neurons of the initial layer, so the model incorporates more trainable parameters at K=3. Thus the MLP receives more information and has more capacity to analyze it at K=3 and, as expected, performed better; the difference in validation accuracy between K=1 and K=3 was significant. When the study was extended to additional values of K, accuracy increased as K increased from 1 to 3 and it remained high as K increased from 3 to 5 (**Table II**). We inferred that MLP training should use $K \geq 3$.

Table II. MLP accuracy vs K

K	Params	Val Acc	Std Dev
1	385	75.06%	0.26
2	577	80.50%	0.55
3	1345	85.86%	0.46
4	4417	86.80%	0.82
5	16705	86.48%	0.67

The training data was 64% LncRNA. To assess the impact of data imbalance, we down-sampled the LncRNAs. A 2x16 MLP was trained and validated on 5785 coding and 5781 non-coding sequences with K=4. Validation accuracy was measured as 85.57% ± 0.43, a slight reduction that is possibly due to the overall reduction in training set size. We

concluded that the data imbalance would not confound further use of the full training data set.

3.2 Effect of Model Complexity

In the previous results, the number of MLP trainable parameters increased with K. To estimate whether the increased accuracy was due to parameter expansion only, we designed six MLPs that would apply high parameter counts to low values of K. Three models trained on K=1 data using 1281 to 1713 parameters. The maximum accuracy was 75.20%, short of the 80.50% achieved previously by setting K=2. Another three models trained on K=2 data with 1121 to 1666 parameters. The maximum accuracy was 80.83%, short of the 85.86% achieved previously by setting K=3. Similar results were observed with models designed to approximate the K=5. These results indicate that model complexity was less critical than K, *i.e.* feature dimensionality, for boosting MLP accuracy.

We also tested an MLP with K-mers formed by skipping the middle nucleotide of each consecutive five nucleotides *i.e.* a spaced seed. We saw no improvement over K=4 (data not shown).

A trained model is overfitting if it performs better on the training data than on the validation data. We observed overfitting at K=5 as training accuracy hit 100% in every round, and to a lesser extent, at K=4. This cost CPU cycles but did not change results since we always selected the model with the maximum validation accuracy over 200 epochs, analogous to early stopping. However, models that overfit at some early epoch never improved their validation accuracy later. Dropout is a design feature that can reduce overfitting by inactivating a random subset of neurons before each training epoch [34]. Overfitting was reduced after we applied 50% dropout, though accuracy did not improve. At K=5, validation accuracy declined by 0.3 percentage points.

In summary, our basic MLP models achieved up to 87% accuracy at identifying the LncRNAs within our validation set. These models were trained on K-mer profiles, *i.e.* word frequencies. Accuracy was highest using K values of 3 to 5 and this was not merely due to model complexity. (Larger values of K were not tested since the 4^K feature dimensions would approach the training data size.) Since primary sequences contain more information than K-mer profiles, these results provide a lower bound for what should be achievable with networks designed for primary sequences.

3.3 Exploiting Nucleotide Order

To test whether sequence-aware ANNs could out-perform the basic ANNs used so far, we designed a recurrent neural network (RNN). This model included an embedding layer to learn a helpful transformation of nucleotides or K-mers into vectors. It included one GRU layer using 32 neurons, and one fully connected (FC, *i.e.* dense) layer with 32 neurons, both incorporating 50% dropout. The model used *tanh* activation in intermediate layers and *sigmoid* activation in the output layer; other activations tested sometimes led to numerical instability *i.e.* loss equal to zero or infinity.

The input to an RNN is usually an ordered list. For the RNA data, we input each RNA as a sequence of nucleotides. This is equivalent to using K-mers with K=1. We also tested K=3 by presenting the model with overlapping 3-letter "words" extracted in order from consecutive RNA positions *i.e.* stride=1. In contrast to the MLP experiment, the RNN inputs consisted of K-mer sequences not frequencies. Holding the other layer dimensions constant, we tested several values for the embedding dimension. The RNN accuracy ranged from 87% to 91% (**Table III**). The maximum accuracy from any one fold of training was 92.05% using K=3 and the 16-dimensional embedding.

Table III. Various embeddings with a 1x32 GRU RNN.

Embed Dim	K	Params	Val Acc	Std Dev
4	1	4757	86.68%	1.24
	3	4997	89.26%	2.32
16	1	5969	87.66%	0.95
	3	6929	90.59%	2.01
64	1	10817	87.92%	0.62
	3	14657	88.80%	2.09
128	1	17281	88.19%	0.95
	3	24961	87.56%	0.55

As expected, the RNNs achieved higher accuracy than the MLPs. The increase was not due to model complexity, since the RNNs were more accurate even with fewer parameters than

the MLP with the equivalent value of K. The differences due to the dimensionality of the embedding layer were not significant. Although the MLP with 16-dimensional embedding using K=3 scored highest, the overall effect due to K was not significant.

These results indicate that the RNNs were able to extract more information from sequences than the MLPs could extract from profiles.

3.4 Effect of RNN Model Complexity

To test whether RNN adjustments would increase RNN accuracy, we varied some of our RNN's hyperparameters. We held constant the 16-dimensional embedding and the K=3 data treatment, while varying the numbers of internal layers and neurons per layer. We also tested a bidirectional GRU which uses forward and backward GRU layers of the same dimension. Accuracy peaked with the 2x64 unidirectional GRU (**Table IV**) and this model also produced the maximum accuracy in any one fold of training, 94.54%. (Because that maximum and variance were high on this configuration, this test was repeated. The table reflects statistics for 10 train+validation folds for this model, and 5 for the others.)

The additional layers and neurons increased the number of trainable parameters, and there was a slight positive correlation of validation accuracy to parameter count (r=0.58). Inclusion of the bidirectional GRU provided no significant improvement.

Table IV. Variations on GRU depth.

GRU & FC	BiDir	Params	Val Acc	Std Dev
2x16	no	2961	87.98%	1.32
	yes	4849	88.14%	2.05
1x32	no	6929	90.59%	2.01
	yes	12753	89.72%	2.23
2x32	no	14321	89.67%	1.38
	yes	32625	91.66%	1.84
2x64	no	50129	91.27%	2.65
	yes	119505	91.25%	3.26

These results indicate that increasing model complexity should improve RNN accuracy. We did test a more complex 2x128 GRU network but numerical instability was observed (loss="nan"). More fine tuning will be required to explore such models.

3.5 Effect of LSTM vs GRU Models

The LSTM model is an RNN layer with more parameters than the GRU. In order to compare networks based on these two models, we tested both using five values of K, a 16-dimensional embedding layer, 2 recurrent layers with 64 neurons each, and 2 fully connected layers with 64 neurons each (**Table V**).

Table V. Variations on K and RNN type.

	K	Params	Val Acc	Std Dev
LSTM 2x64	1	62225	88.85	0.39
	2	62417	88.58	n/a
	3	63185	88.34	1.12
	4	66257	88.21	n/a
	5	78545	87.21	0.77
GRU 2x64	1	49169	87.83	1.96
	2	49361	87.44	1.43
	3	50129	91.27	2.65
	4	53201	87.61	0.77
	5	65489	86.52	0.91

Of these configurations, the highest accuracy was achieved by the GRU with K=3, which was already reported in **Table IV**. The LSTMs did not provide a significant improvement over the GRUs. Overfitting was observed with both model types trained on K=5 data, though not as pronounced as with the MLP.

In summary, the performance of our LSTM models on our data was not significantly different from that of the GRU models, despite their extra complexity.

3.6 Effect of LncRNA Sequence Length

The prior experiments used the subset of sequences of length 200 to 1000, which may not be representative of the full data set (see **Fig. 1**). To explore the effect on sequence length on training and validation, the 32K sequences selected from GenCode (**Table I**) were partitioned six ways using arbitrary thresholds chosen for balance (**Table VI**).

After 20% of each subset was set aside for validation, a network was trained on each of five partitions and validated on six. The network used K=3, a 16-dimensional embedding layer, and 1x16 GRU and FC layers with 50% dropout. Each validation used the parameters that maximized training accuracy in one fold of 100 epochs of training.

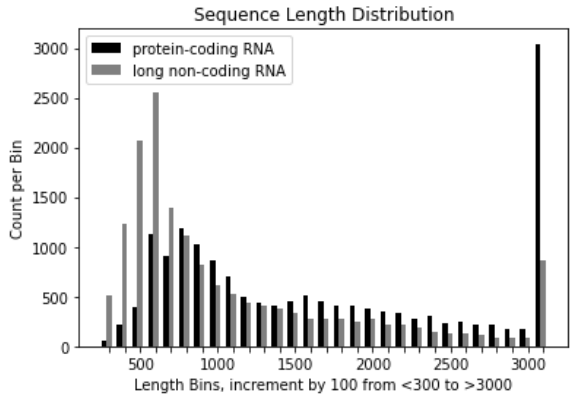

Fig. 1. Distribution of sequence lengths in GenCode 34 data.

Table VI. Data subset characteristics.

	Min Len	Max Len	Coding	NonCoding	Total
A	200	600	1793	6372	8165
B	600	900	3114	3328	6442
C	900	1300	2492	1986	4478
D	1300	1900	2642	1779	4421
E	1900	3000	2918	1672	4590
F	3000	30000	3041	859	3900

The results (**Table VII**) show sensitivity of GRU accuracy to length of the training and validation data. Models trained on subsets B (600-900 nucleotides) and D (1.3K-1.9K) were the most versatile. The model trained on D provided highest accuracy on the longest subset, F, while the model trained on E (1.9K-3K) performed poorly on every subset. LncRNA identification was more accurate on the longer validation sequences. LSTM and MLP models also classified longer sequences better if they were trained on longer sequences (not shown). Thus, sequence length should be considered when training models and when applying trained models to unlabeled sequences.

Table VII. Variation of accuracy with sequence length.

Validation Subset	Train Subset					
	A	B	C	D	E	Avg
A	**88**	83	63	81	32	69
B	78	**81**	75	75	47	71
C	70	**81**	78	76	56	72
D	66	81	81	**82**	60	74
E	64	85	86	**87**	60	76
F	76	85	74	**90**	69	79
Avg	74	83	76	82	54	74

3.7 Computation

Most experiments used 5-fold cross-validation and 200 epochs per fold. MLP training used 1-2 GB RAM and 261±92 elapsed sec per 200 epochs. RNN training, excluding single-layer models, used 1-4 GB RAM and 9828±2807 sec per 200 epochs.

4 DISCUSSION

We explored neural network designs for the LncRNA identification problem for which prior solutions achieved 95% accuracy [16, 17]. We explored two architectural types: MLP and RNN (**Fig. 2**).

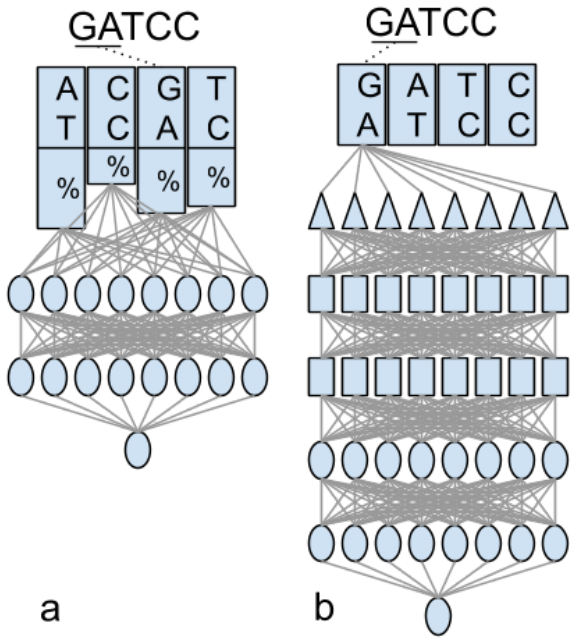

Fig. 2. Schematic of the two architectures explored. For brevity, the figure shows a hypothetical 5-nucleotide RNA input, only 4 of 16 possible K-mers at K=2, and only 8 neurons per layer. (a) Each RNA sequence was presented as a K-mer frequency profile to an MLP having 2 fully connected layers. (b) Each RNA sequence was presented as a list of K-mers in sequence order to an RNN having an embedding layer (triangles), 2 recurrent layers (rectangles), and 2 fully connected layers (ovals).

The MLP is a basic neural network with no special adaptation for handling sequences. Our MLP models ran on K-mer profiles of RNA sequences, a "bag of words" approach that discards word order. By setting K=1, we also tested a "bag of nucleotides" approach and achieved less accuracy, as expected. In contrast, the RNN is a neural network specially adapted for time-ordered sequences. RNNs have been applied to language translation [29] but it is unclear how they perform on very long RNA sequences of up to 30K nucleotides. RNNs

retain prior information as their focus moves forward through each sequence, and bidirectional RNNs also do the reverse. While this "sequence of nucleotides" approach receives sufficient information to construct K-mer profiles in theory, we also tested it with a "sequence of words" approach by providing RNNs with consecutive K-mers, ordered according to the primary sequence.

Our MLP networks achieved high accuracy *e.g.* 87% with a 2x16 MLP and K=4 inputs. While our results may not generalize to all datasets and test conditions, they demonstrate that the "bag of words" approach is sufficient for a great deal of inference about RNAs, confirming previous K-mer based studies of LncRNA function [23]. Our RNN networks achieved higher accuracy, although the gain was less than we expected from sequence-based approaches.

Several of our RNN networks exceeded 90% accuracy. The maximum accuracy measured on any one RNN was 94.5% for a model that scored 91.3% on average. This RNN model was able to incorporate more information than all the MLP networks tested. This model used a 2x64 GRU and K=3 inputs.

We observed only slight differences based on the RNN's embedding dimension, despite a large range of values tested. However we saw a slight trend toward higher accuracy with RNNs of increasing neuron counts, so larger models should be tested. As with RNNs, one dimensional convolutional networks (CNNs) can combine information from different parts of a sequence, and their filter size parameter might play a role similar to the K we used to break sequences into K-mers. We observed no significant difference between RNN types (LSTM, GRU, BiGRU) but we would like to explore others such as convolutional LSTMs.

We have so far not explored the relative contributions of the recurrent *vs.* the fully connected layers in our RNN models. We observed that overfitting early in a 200-epoch training fold was a good predictor of low accuracy overall, so it may be helpful to implement early stopping followed by a training restart on shuffled data.. All the models tested here relied on RNA primary sequence data alone, but other features could also be incorporated [16].

The data set used here was not particularly large for a study of modern machine learning. Paucity of data constrained our choices for the values of K and led us to use imbalanced data sets. Downsampling would have risked overfitting, especially by the most complex models. Data augmentation has been applied for pre-training models [16, 17] but one alternative exists in the GenCode data itself in the form of multiple transcripts per gene. Although we used only one transcript per gene to eliminate redundancy, a remaining challenge is to harvest non-redundant information from this larger set.

ACKNOWLEDGEMENTS

This work was supported in part by funding from the National Science Foundation (award #1747788, #1920920).

REFERENCES

1. Djebali S, Davis CA, et al. Landscape of transcription in human cells. Nature. 2012; 489:101–8.

2. Derrien T, Johnson R, et al. The GENCODE v7 catalog of human long noncoding RNAs: analysis of their gene structure, evolution, and expression. Genome Res. 2012;22:1775–89.

3. Harrow J, Frankish A, et al. GENCODE: the reference human genome annotation for The ENCODE Project. Genome Res. 2012; 22:1760–74.

4. Wang KC, Chang HY. Molecular mechanisms of long noncoding RNAs. Mol Cell. 2011; 43:904–14.

5. Moran VA, Perera RJ, Khalil AM. Emerging functional and mechanistic paradigms of mammalian long non-coding RNAs. Nucleic Acids Res. 2012; 40:6391–400.

6. Mercer TR, Mattick JS. Structure and function of long noncoding RNAs in epigenetic regulation. Nat Struct Mol Biol. 2013; 20:300–7.

7. Nie L, Wu H-J, et al. Long non-coding RNAs: versatile master regulators of gene expression and crucial players in cancer. Am J Transl Res. 2012; 4:127–50.

8. Wapinski O, Chang HY. Long noncoding RNAs and human disease. Trends Cell Biol. 2011; 21:354–61.

9. Tripathi V, Ellis JD, et al. The nuclear-retained

noncoding RNA MALAT1 regulates alternative splicing by modulating SR splicing factor phosphorylation. Mol Cell. 2010; 39:925–38.

10. Kohlmaier A, Savarese F, et al. A chromosomal memory triggered by Xist regulates histone methylation in X inactivation. PLoS Biol. 2004; 2:E171.

11. Zerbino DR, Achuthan P, et al. Ensembl 2018. Nucleic Acids Res. 2018; 46:D754–61.

12. Jia H, Osak M, et al. Genome-wide computational identification and manual annotation of human long noncoding RNA genes. RNA. 2010; 16:1478–87.

13. Frankish A, Diekhans M, et al. GENCODE reference annotation for the human and mouse genomes. Nucleic Acids Res. 2019; 47:D766–73.

14. Yotsukura S, duVerle D, et al. Computational recognition for long non-coding RNA (lncRNA): Software and databases. Brief Bioinformatics. 2017; 18:9–27.

15. Abascal F, Juan D, et al. Corrigendum: Loose ends: almost one in five human genes still have unresolved coding status. Nucleic Acids Res. 2018; 46:12194.

16. Yang C, Yang L, et al. LncADeep: an ab initio lncRNA identification and functional annotation tool based on deep learning. Bioinformatics. 2018; 34:3825–34.

17. Hill ST, Kuintzle R, et al. A deep recurrent neural network discovers complex biological rules to decipher RNA protein-coding potential. Nucleic Acids Res. 2018; 46:8105–13.

18. Kong L, Zhang Y, et al. CPC: assess the protein-coding potential of transcripts using sequence features and support vector machine. Nucleic Acids Res. 2007; 35 Web Server issue:W345-9.

19. Wang L, Park HJ, et al. CPAT: Coding-Potential Assessment Tool using an alignment-free logistic regression model. Nucleic Acids Res. 2013; 41:e74.

20. Li A, Zhang J, Zhou Z. PLEK: a tool for predicting long non-coding RNAs and messenger RNAs based on an improved k-mer scheme. BMC Bioinformatics. 2014; 15:311.

21. Achawanantakun R, Chen J, et al. LncRNA-ID: Long non-coding RNA IDentification using balanced random forests. Bioinformatics. 2015; 31:3897–905.

22. Gudenas BL, Wang L. Prediction of LncRNA Subcellular Localization with Deep Learning from Sequence Features. Sci Rep. 2018; 8:16385.

23. Kirk JM, Kim SO, et al. Functional classification of long non-coding RNAs by k-mer content. Nat Genet. 2018; 50:1474–82.

24. Rosenblatt F. The perceptron: a probabilistic model for information storage and organization in the brain. Psychol Rev. 1958; 65:386–408.

25. Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back-propagating errors. Nature. 1986; 323:533–6.

26. Elman JL. Finding Structure in Time. Cogn Sci. 1990; 14:179–211.

27. Werbos PJ. Backpropagation through time: what it does and how to do it. Proc IEEE. 1990; 78:1550–60.

28. Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput. 1997; 9:1735–80.

29. Cho K, van Merrienboer B, et al. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. EMNLP; 2014. p. 1724–34.

30. Chollet F, Others. Keras. 2015. https://github.com/fchollet/keras. Accessed 14 Oct 2020.

31. Mattei E, Ausiello G, et al. A novel approach to represent and compare RNA secondary structures. Nucleic Acids Res. 2014; 42:6146–57.

32. Rabani M, Kertesz M, Segal E. Computational prediction of RNA structural motifs involved in posttranscriptional regulatory processes. Proc Natl Acad Sci USA. 2008; 105:14885–90.

33. Adjeroh D, Allaga M, et al. Feature-Based and String-Based Models for Predicting RNA-Protein Interaction. Molecules. 2018; 23.

34. Srivastava N, Hinton G, et al. Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research. 2014; 15:1929–58.

35. Kingma DP, Ba J. Adam: A method for stochastic optimization. arXiv preprint arXiv:14126980. 2014.