Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
Advance Access Publication Date: Day Month Year
Manuscript Category

OXFORD

Sequence analysis

lncRNAnet: Long Non-coding RNA Identification using Deep Learning

Junghwan Baek [1], Byunghan Lee [2], Sunyoung Kwon [2] and Sungroh Yoon [1, 2,*]

[1] Interdisciplinary Program in Bioinformatics, Seoul National University, Seoul 08826, Korea and
[2] Electrical and Computer Engineering, Seoul National University, Seoul 08826, Korea

*To whom correspondence should be addressed.

Associate Editor: XXXXXXX

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Abstract

Motivation: Long non-coding RNAs (lncRNAs) are important regulatory elements in biological processes. LncRNAs share similar sequence characteristics with messenger RNAs (mRNAs), but they play completely different roles, thus providing novel insights for biological studies. The development of next-generation sequencing (NGS) has helped in the discovery of lncRNA transcripts. However, the experimental verification of numerous transcriptomes is time consuming and costly. To alleviate these issues, a computational approach is needed to distinguish lncRNAs from the transcriptomes.
Results: We present a deep learning-based approach, lncRNAnet, to identify lncRNAs that incorporates recurrent neural networks (RNNs) for RNA sequence modeling and convolutional neural networks (CNNs) for detecting stop codons to obtain an open reading frame (ORF) indicator. LncRNAnet performed clearly better than the other tools for sequences of short lengths, on which most lncRNAs are distributed. In addition, lncRNAnet successfully learned features and showed 7.83%, 5.76%, 5.30%, and 3.78% improvements over the alternatives on a human test set (HT) in terms of specificity, accuracy, F1-score, and area under the curve (AUC), respectively.
Availability: Data and codes are available in http://data.snu.ac.kr/pub/lncRNAnet.
Contact: sryoon@snu.ac.kr

1 Introduction

Only less than 2% of the three billion base pairs in the human genome encode proteins (Mattick, 2001; Mattick and Makunin, 2006; Lee, 2012; Alexander *et al.*, 2010), and the functions of the remaining parts remain unknown. Among these remaining parts, non-coding RNAs (ncRNAs), which refer to transcripts that are not translated into proteins, are often considered key regions responsible for various biological processes.

The long non-coding RNAs (lncRNAs), which are ncRNAs composed of more than 200 nucleotides (nt), are of particular interest. Although the low conservation of lncRNAs often makes them appear as transcriptional noise (Pang *et al.*, 2006; Struhl, 2007), the known lncRNAs play key roles as regulatory elements in biological processes and engage in various disease processes, including cancers, neurological disorders, and immunological disorders (Wapinski and Chang, 2011; Bhan and Mandal, 2014). Identifying lncRNAs is thus essential for understanding gene regulation and the potential causes of important diseases.

Typically, lncRNAs have sequence characteristics similar to those of protein-coding transcripts, making lncRNAs difficult to identify, whereas short ncRNAs (sncRNAs) are clearly distinguished from protein-coding transcripts. Some lncRNAs even undergo transcriptional and post-transcriptional processes just like protein-coding transcripts (Ponting *et al.*, 2009). Although various transcriptomes have been sequenced to date, many lncRNAs remain undiscovered and exist as "dark regions" in the genome (Kapranov and Laurent, 2012). In expression measurements, the low expression levels of lncRNAs are treated as anomalies, hindering their discovery (Kung *et al.*, 2013; Derrien *et al.*, 2012). The application of next-generation sequencing (NGS) has improved the efforts to annotate lncRNAs in terms of cost and accuracy, but many experimental approaches to verifying lncRNA transcriptomes still require significant time and resources.

To complement available experimental techniques, computational lncRNA identification methods have been proposed (Liu *et al.*, 2006; Kong *et al.*, 2007; Wang *et al.*, 2013; Sun *et al.*, 2013b,a; Li *et al.*, 2014; Lertampaiporn *et al.*, 2014; Achawanantakun *et al.*, 2015; Pian *et al.*, 2016; Lin *et al.*, 2011; Tripathi *et al.*, 2016). Their common formulation

is to use a binary classifier that can predict whether a given nucleotide sequence encodes an lncRNA. Various classification models have been used, including support vector machines (SVMs), random forests (RFs), and neural networks.

Existing computational methods heavily rely on the features extracted from identified lncRNAs and/or their comparative genomics-based profiles obtained by database searches and multiple sequence alignments (MSAs). The use of the features and MSA profiles may be necessary to some extent to reveal some common patterns of lncRNAs. However, it clearly limits the accuracy and robustness of lncRNA identification and the opportunity to identify novel ncRNAs that have subtle properties different from those of the known lncRNAs.

Specifically, existing techniques often suffer from the following issues. First, DB searches are useful for highly conserved sequences (e.g., protein-coding transcripts) but not for lncRNAs. They have lower conservation values than the exons of the protein-coding transcripts. Consequently, DB search-based identifications of lncRNAs may erroneously predict lncRNAs as coding transcripts (Guttman *et al.*, 2009). In addition, DB searches are limited to well-annotated species and otherwise produce unsatisfactory results, and the underlying alignment operations are often time consuming and affected by alignment parameters that are usually set heuristically. Second, many of the previous computational approaches depend on manually crafted features and heuristic decision criteria. Certain features may be helpful in designing a robust lncRNA classifier, but identifying and validating effective features require a substantial amount of human effort. Even with such effort, manually designed features may still fail to capture non-canonical signals that exist in elusive lncRNAs. The use of an unprincipled approach for determining decision criteria can hurt the accuracy and generalization (i.e., the performance of a machine learning algorithm for the data not used for training) of the classifier.

To overcome these limitations, we propose lncRNAnet, a deep learning-based approach for identifying lncRNAs. The key characteristics and notable contributions of our approach include the following:

1. The proposed lncRNAnet successfully learned intrinsic features by incorporating recurrent neural networks (RNNs) for RNA sequence modeling.
2. An open reading frame (ORF) indicator was proposed by exploiting stop codon detections based on convolutional neural networks (CNNs). The ORF indicator adopts prior knowledge about translation to reinforce the model.
3. LncRNAnet successfully detected short lncRNA candidates and robustly performed in the global length range, which is important because lncRNAs are relatively shorter than mRNAs due to smaller exon numbers.

To validate our approach, we tested lncRNAnet with two datasets containing 7,000 transcripts each and compared it with four existing tools in terms of widely used metrics. In our experiments, the proposed lncRNAnet successfully learned the inherent features of lncRNA transcripts and delivered the highest performance, outperforming the best alternative on the HT dataset in terms of specificity, accuracy, F1-score, and the area under the curve (AUC) by 7.83%, 5.76%, 5.30%, and 3.78%, respectively.

2 Background

As mentioned in the previous section, the existing approaches use various handcrafted features to identify lncRNAs. The novelty of our method is the use of deep CNNs and RNNs to learn the features of lncRNAs. A sequence of nucleotides can be divided into a set of consecutive non-overlapping triplets (*i.e.*, six reading frames). Among the reading frames, an open reading frame (ORF) is a series of codons that have the potential to be translated. The lncRNAs can be considered a complementary set of coding transcripts that includes the ORFs; hence, it is essential to detect the ORFs to distinguish lncRNAs from the abundant transcripts. Based on this point, we use CNNs to detect the ORFs as the coding transcript candidates and stacked RNNs to distinguish lncRNAs from them.

2.1 Long non-coding RNA (lncRNA)

LncRNAs are an RNA family that does not encode proteins, and they play special roles in various biological processes by regulating gene expression (Mercer *et al.*, 2009). Specifically, lncRNAs act as *cis* and *trans* elements, regulating nearby and distant genes, respectively, to activate or degrade the transcription process by binding to the transcriptional factors. The characteristics of lncRNAs are similar to those of protein-coding transcripts, such as splicing, $5'$-capping, and poly-A tailing (Wang and Chang, 2011; Quinn and Chang, 2016; Wilusz *et al.*, 2009), although their roles are different. Furthermore, the fact that some lncRNAs act as both protein-coding and non-coding transcripts complicates lncRNA classification (Dinger *et al.*, 2008).

To distinguish lncRNAs from protein-coding transcripts, we can consider the following rules (Dinger *et al.*, 2008): ORF-based, sequence- and structure-based, and filtering-based rules. The ORF-based rules include the length of an ORF, the conservation of the ORFs, and the ratio between the length of an ORF and a transcript sequence. The sequence-structure-based rules consider the conservation of the secondary structure. The filtering-based rules preprocess the artifacts caused by the sequencing. The central point of these rules is based on the fact that lncRNAs can be considered as a complementary set of coding transcripts because they do not code proteins.

In this study, we exploit the ORFs detected by CNNs in lieu of the aforementioned handcrafted features.

2.2 Convolutional Neural Network (CNN)

A CNN is a neural network specialized for image data that models the relations of the adjacent pixels. A CNN applies filters in the form of a convolution operation to extract features from the data. The advantage of a CNN is that it reduces the parameters compared to other neural networks by sharing them as multiple filters (LeCun *et al.*, 1998). The convolution filters share the parameters independent of position; thus, we can reduce the number of used parameters. This parameter sharing of the convolutional filters and the local connections of the nodes increase the performance in handling sparsely connected data. CNNs have shown outstanding performance in two-dimensional sparse data such as images or matrices (Simonyan and Zisserman, 2014; Van den Oord *et al.*, 2013). In addition, one-dimensional CNNs have been recently applied to sequential data classification, language modeling, and other related problems in natural language processing (Kim, 2014; Kalchbrenner *et al.*, 2014; Collobert and Weston, 2008).

In this study, we use one-dimensional CNNs to detect the ORFs as the candidates of coding transcripts.

2.3 Recurrent Neural Network (RNN)

An RNN is a neural network that feeds the output of a previous cell as the current input of the network (*i.e.*, acyclic graph). This shape can help the RNNs to learn the sequential behavior; hence, they are widely used for classifying sequences such as in sequence-to-sequence learning and speech recognition. An acyclic graph can be described as a series of connections of cells. This graph can be unrolled infinitely according to the length of the input, and the parameters of each cell are shared through each time step; thus, it behaves flexibly for the sequential data. The training procedure using the unrolled network is called backpropagation through time (BPTT).

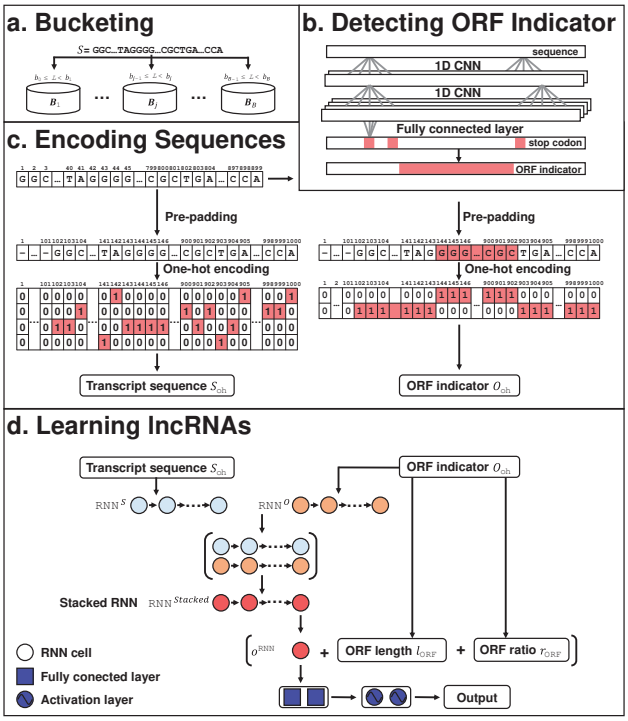

Fig. 1. The overview of lncRNAnet. First, each sequence is sorted into a bucket, and an ORF indicator is obtained by CNNs. Then, a transcript sequence and an ORF indicator are encoded. The RNN model uses a transcript sequence and an ORF indicator as inputs, and then, an ORF length and an ORF ratio are calculated from the ORF indicator as additional features of the model.

Algorithm 1 Pseudocode of lncRNAnet

1: **Input:** Sequences $S : s_1, s_2, \cdots, s_L$
▷ $s_i \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$, L: length of S, N: Number of sequences
2: **Output:** $\mathbf{y}$ (lncRNA/ Protein coding transcript)
3: Maximum sequence length of a bucket : $b_1, b_2, \cdots, b_B$ ▷ $b_0 = 0$
4: Buckets : $\mathbf{B}_1, \mathbf{B}_2, \cdots, \mathbf{B}_B$ ▷ B: Number of buckets
5: **for** $i = 1$ to N **do**
6: **for** $j = 1$ to B **do**
7: **if** $L \leq b_j$ **then**
8: $\mathbf{B}_j$.append(S)
9: O=ORFindicator(S) ▷ $O : o_1, \cdots, o_L$
10: S_{pad}=pre-padding(S)
11: O_{pad}=pre-padding(O)
12: S_{oh}=one-hot(S_{pad}) ▷ S_{oh}: preprocessed transcript sequence
13: O_{oh}=one-hot(O_{pad}) ▷ O_{oh}: preprocessed ORF indicator
14: **break**
15: **end if**
16: **end for**
17: **end for**
18: **repeat**
19: select random number $k \leq B$ to select $\mathbf{B}_k$
20: take n_{batch} sequences from $\mathbf{B}_k$ ▷ n_{batch} : mini-batch size
21: Calculate ORF length l_{ORF}
22: Calculate ORF ratio r_{ORF}
23: Compute RNN^S, and RNN^O
24: Merge two output representations for each time point out$_t^S$, and out$_t^O$ into x_t^{RNN} ▷ out^S, out^O: output representations of the RNN^S and RNN^S
25: Send x_t^{RNN} to stacked RNNs
26: Merge out$^{\text{Stacked}}$, l_{ORF} and r_{ORF} as an input to the fully connected layer
27: Softmax calculation from the fully connected layer output out$^{\text{FC}}$
28: Minimize the cross entropy loss
29: $\mathcal{L}(\mathbf{y}) = -\frac{1}{n_{\text{batch}}} \sum_{n=1}^{n_{\text{batch}}} (y_n \log(p_n) + (1 - y_n) \log(1 - p_n))$
30: **until** $\#epoch = n_{\text{epoch}}$

The BPTT calculates the gradient of each input-output pairs in an unrolled network. The network shares parameters, so the updated cost is equal to the average of calculated gradients in each time step. The BPTT method has constraints regarding training time with inputs of highly variable lengths, so we used a bucketing technique to address this issue.

Although RNNs show outstanding performances on various tasks, they are vulnerable to long sequences in saving long-term dependencies. For a standard RNN, learning the whole input causes the vanishing gradient problem during the training. To control the data flow of the internal memories to learn the long-term dependencies by adding gates, Hochreiter and Schmidhuber (1997) developed the long short-term memory (LSTM), and Cho *et al.* (2014) developed the gated recurrent unit (GRU).

In this study, RNNs with LSTM units are used to learn the intrinsic behavior of lncRNAs.

3 Proposed Methodology

Fig. 1 shows the overview of our proposed method, lncRNAnet. To determine whether a given sequence is an lncRNA, our method uses the following four phases: (a) bucketing, (b) detecting ORF indicators, (c) encoding sequences, and (d) learning lncRNAs. Algorithm 1 shows the details of each phase. Our model accepts candidate transcript sequences as inputs. In the bucketing phase, each sequence is sorted into buckets with regard to their sequence length (lines 6–8). Then, an ORF indicator is identified for each sequence (line 9). In the encoding sequences phase, the transcript sequence and the ORF indicator are pre-padded to match the maximum sequence length of each bucket. Using a one-hot encoding scheme, the transcript sequence and the ORF indicator are preprocessed (lines 10–13). To learn lncRNAs, the sequence data are trained through the whole architecture (lines 18–29).

3.1 Bucketing

As mentioned in the previous section, RNNs can expand their shape with regard to the input sequence length. Because the length distribution of transcripts varies from hundreds to hundreds of thousands, we decided to use an RNN as the building block of our architecture. However, if the lengths of the sequences in a dataset vary greatly, randomly selected sequences in a batch will have different lengths. In this case, the length of the sequences can be made to be equal through sequence padding, but sparse values (*e.g.*, [0, 0, 0, 0]) of padded inputs may occupy a large proportion and may hamper training time. Bucketing can alleviate these sparse inputs (*i.e.*, padded values) by selecting batches from a bucket, which contains sequences with similar lengths (Khomenko *et al.*, 2016). A sequence S of length L such that $b_{j-1} < L \leq b_j$ where $j \in \{1, \cdots, B\}$ is sorted into a bucket $\mathbf{B}_j$ with the maximum sequence length b_j. Batch-wise training with sequences of similar lengths can be performed later by padding each sequence in the bucket to the maximum sequence length of the bucket.

3.2 Detecting an ORF Indicator

All protein-coding transcripts possess ORFs, sequences that can be translated. Identifying candidate ORFs in the transcript $S = (s_1, \cdots, s_L)$ is an important guideline for distinguishing lncRNAs from protein-coding transcripts. Normally, ORFs can be identified by finding sequences between a start codon and a stop codon. However, the occurrence of non-canonical start codon signals disturbs the ORF detection. In this study, ORFs are searched using only the stop codons to identify non-canonical signals. ORFs between stop codons are called stop-to-stop ORFs.

Stop-to-stop ORFs are identified using the one-dimensional CNN. The model is shown in Fig. 1(b). One-dimensional CNNs are stacked, and fully

connected layers are attached to detect stop codons for each time step. After identifying all stop codons, an ORF indicator $O = (o_1, \cdots, o_L)$ is processed by finding the longest stop-to-stop ORF in three forward frames. Each character o_t is encoded to 0 if the character is in the ORF and to 1 if the nucleotide is not in the ORF.

$$o_t = \begin{cases} 0 & s_t \in \text{ORF} \\ 1 & s_t \notin \text{ORF} \end{cases} \quad where \quad t \in \{1, \cdots, L\}$$

From the ORF indicator, we consider two additional ORF-related features, l_{ORF} and r_{ORF}. Let the maximum ORF length be l_{ORF}, and the ratio, l_{ORF} over transcript length L be denoted by r_{ORF}.

3.3 Encoding Sequences

After bucketing, each bucket has sequences with similar lengths. Our method exploits the premise that the RNN unrolls to the maximum sequence length of the batch from $\mathbf{B}_j, b_j$. Hence, sequence padding is mandatory for training. The transcript sequence S is pre-padded with character '$-$' to match the maximum length. The character s_t of sequence S consists of four nucleotides $\{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{T}\}$. A sequence $S = (s_1, \cdots, s_L)$ whose length $b_{j-1} \leq L < b_j$ is pre-padded to S_{pad} as follows:

$$S = (s_1, \cdots, s_L) \xrightarrow{\text{pre-padding}} S_{\text{pad}} = (\overbrace{-, \cdots, -}^{b_j - L}, s_1, \cdots, s_L).$$

The pre-padded sequence S_{pad} is preprocessed to be trained in RNNs. The padded sequence is transformed to S_{oh} by using one-hot encoding (Baldi and Brunak, 2001). One-hot encoding is a method in which each character in a sequence is represented as a binary vector whose dimension is equal to the number of characters. In this case, nucleotides consist of four characters $\{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{T}\}$. Each nucleotide s_t can be expressed as $\texttt{A} = [1, 0, 0, 0]$, $\texttt{C} = [0, 1, 0, 0]$, $\texttt{G} = [0, 0, 1, 0]$ and $\texttt{T} = [0, 0, 0, 1]$, and the padded value '$-$' can be represented as $[0, 0, 0, 0]$. For instance, the start codon '$\texttt{ATG}$' can be expressed as a series of four-dimensional binary vectors $\langle [1, 0, 0, 0], [0, 0, 0, 1], [0, 0, 1, 0] \rangle$. As a result, the transcript sequence is projected to a series of four-dimensional tensors of shape $(4, b_j)$.

The ORF indicator sequence O, processed from the transcript sequence S, is also pre-padded with character '$-$' to match the maximum length. The ORF indicator character o_t of O consists of two characters $\{\texttt{0}, \texttt{1}\}$. The sequence $O = (o_1, \cdots, o_L)$ whose length $b_{j-1} \leq L < b_j$ is pre-padded as follows:

$$O = (o_1, \cdots, o_L) \xrightarrow{\text{pre-padding}} O_{\text{pad}} = (\overbrace{-, \cdots, -}^{b_j - L}, o_1, \cdots, o_L)$$

In addition, O_{pad} is preprocessed to be trained in the RNN by one-hot encoding. As a result, the ORF indicator is projected to a series of two-dimensional tensors of shape $(2, b_j)$.

3.4 Learning lncRNAs

Preprocessed inputs are trained through the neural network. The transcript sequence S_{oh} (dimension $(4, b_j)$) and the ORF indicator O_{oh} (dimension $(2, b_j)$) are passed through each masking layer. The masking layer does not apply a gradient to the input-output pairs if the input originates from padded values; thus, the model is not trained from input $[0, 0, 0, 0]$. Thus, the network does not calculate the loss from paddings. Each masked input from the transcript sequence and the ORF indicator is then fed into many-to-many RNNs with n_h cells. As shown in Fig. 1(d), the output representations of RNN^S and RNN^O for each time step are out_t^S and out_t^O (both having dimension (n_h)). For each time step, outputs out_t^S and out_t^O from two RNNs are merged to be a state x_t^{RNN} of $\text{RNN}^{\text{Stacked}}$ (dimension $(2n_h)$),

$$\begin{aligned} x^{\text{RNN}} &= \left[x_1^{\text{RNN}}, \cdots, x_t^{\text{RNN}}, \cdots, x_{b_j}^{\text{RNN}}\right] \\ &= \left[\left[\ \text{out}_1^S, \text{out}_1^O\ \right], \cdots, \left[\ \text{out}_t^S, \text{out}_t^O\ \right], \cdots, \left[\ \text{out}_{b_j}^S, \text{out}_{b_j}^O\ \right]\right] \end{aligned}$$

where $t \in \{1, \cdots, b_j\}$.

The merged state is sent to a many-to-one stacked RNN, $\text{RNN}^{\text{Stacked}}$, with dropout layers (Srivastava *et al.*, 2014). A dropout layer regularizes the network to prevent overfitting of the network. Generally, hidden units in a network are intricately connected. The number of hidden units are relatively larger than the obtained data, resulting in overfitting. The dropout layer helps alleviate this issue by removing part of the connections in the network. In the training phase, dropping the connections reduces the number of learnt perceptrons, resulting a decrease in overfitting and improvement in training speed. Dropping connections can be seen as a sampling of the proposed network, which generalizes the model. The final output from RNN $\text{out}^{\text{Stacked}}$ is merged with the ORF length l_{ORF} and the ORF ratio r_{ORF} to x^{FC} (dimension $(n_h + 2)$),

$$x^{\text{FC}} = [\text{out}^{\text{Stacked}}, l_{\text{ORF}}, r_{\text{ORF}}].$$

The final output is connected to the fully connected layer of dimension two. On the top of the fully connected layer output out^{FC}, each output is out_m^{FC} where $m \in \{0, 1\}$. A softmax activation layer (Bishop, 2006) was added to predict binary classification values. Softmax layers are commonly used in classification problems. The softmax layer modifies the output to normalize probabilities for each class. A softmax function is expressed as follows, where $\mathbf{y}$ is a label if the candidate is an lncRNA or not.

$$P(\mathbf{y} = m | \text{out}^{\text{FC}}) = \frac{1/(1 + \exp(-\text{out}_m^{\text{FC}}))}{\sum_{m=0}^{1} 1/(1 + \exp(-\text{out}_m^{\text{FC}}))} \tag{1}$$

In the model training period, from buckets $(\mathbf{B}_1, \cdots, \mathbf{B}_B)$, $\mathbf{B}_k$ is chosen randomly, and the number of sequences equal to the batch size is selected to create batches. Depending on the bucket $\mathbf{B}_k$, the RNNs unroll themselves to their maximum length b_k. For an epoch, it repeats choosing buckets and selecting sequences until all sequences in the dataset are trained to reduce the model loss $\mathcal{L}$. The cross-entropy loss that is common in binary classification is as follows:

$$\mathcal{L}(\mathbf{y}) = -\frac{1}{n_{\text{batch}}} \sum_{n=1}^{n_{\text{batch}}} (y_n \log(p_n) + (1 - y_n) \log(1 - p_n)) \tag{2}$$

where y_n is the label if it is an lncRNA or not, p_n is the probability of a candidate lncRNA transcript, and n_{batch} is the mini-batch size.

4 Experimental Results

4.1 Datasets

We used lncRNA and protein-coding transcript data from humans and mice downloaded from GENCODE release 25 (Harrow *et al.*, 2012), which were merged with manual annotations from the Human and Vertebrate Analysis and Annotation (HAVANA) group and automated gene annotations from ENSEMBL 87 (Yates *et al.*, 2016). The GENCODE data incorporates features from automatic annotation, manual annotation, and experimental verification.

The lncRNAs were used as positive data, and the protein-coding transcripts were used as negative data. Redundant sequences and sequences containing characters other than $\texttt{A}$, $\texttt{C}$, $\texttt{G}$, and $\texttt{T}$ were excluded. As shown in Fig. 2, the lncRNAs less than 3,000 in length cover about 95.22% of the human data and 90.29% of the mouse data. Only sequences shorter than 3,000 were used in our experiments.

Table 1. The number of sequences used in our experiments

	Model-Training	HT	MT	others	RNA-seq
lncRNAs	21,000	3,500	3,500	1,000	1,210
PCTs	21,000	3,500	3,500	1,000	5,233
total	42,000	7,000	7,000	2,000	6,443

PCTs: protein-coding transcripts
HT: human dataset (Harrow *et al.*, 2012), MT: mouse dataset (Harrow *et al.*, 2012), others: other cross-species datasets (O'Leary *et al.*, 2015; Yates *et al.*, 2016; Bu *et al.*, 2011), RNA-seq: RNA-seq dataset (Spurlock III *et al.*, 2015)

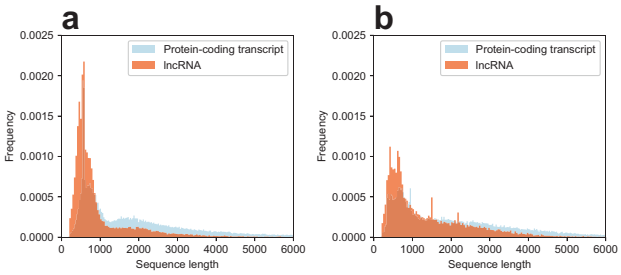

Fig. 2. Length distribution of the protein-coding transcripts and lncRNAs in the (a) human and (b) mouse data

Table 2. Effects of hyperparameter variations through a 5-fold cross validation in terms of prediction accuracy

parameter	# of stacked RNN layers (n_l)			# of RNN hidden units (n_h)				dropout probability (p_d)			
	0	**1**	2	25	50	**100**	200	0	0.25	**0.5**	0.75
Training	0.8642	**0.9857**	0.9057	0.8658	0.9545	**0.9857**	0.8760	0.9854	0.9920	**0.9857**	0.9386
Test	0.8410	**0.9018**	0.8575	0.8629	0.8862	**0.9018**	0.8513	0.8924	0.8901	**0.9018**	0.8729

The total numbers of human protein-coding transcripts and lncRNAs were 94,127 and 27,692, respectively, and those for the mouse data were 60,474 and 14,226, respectively. In our experiment, both the human protein-coding transcripts and lncRNAs were down-sampled to 24,500, where 21,000 were used for training (Model-Training) and 3,500 were used for testing (HT). Those of the mouse sequences were randomly down-sampled to 3,500 only for testing (MT). The details of the numbers of sequences used in our experiments are listed in Table 1. Model-Training was used for model selection through a 5-fold cross validation, the HT dataset was used for model generalization for the human data, and the MT was used for cross-species experiments independent of the training species.

For diverse, cross-species experiments, we prepared transcripts from frogs, zebrafish, cows, and pigs obtained from RefSeq (O'Leary *et al.*, 2015), and transcripts of other species obtained from ENSEMBL (protein-coding transcripts), and NONCODE (Bu *et al.*, 2011) (lncRNAs). 1,000 protein-coding transcripts, and 1,000 lncRNA transcripts of each species were selected as a test dataset.

In addition to compare the performance on the complete reference sequence data (*i.e.*, HT, MT, and other cross-species datasets), we prepared additional real RNA-seq data to evaluate the tools in a real environment. We performed experiments with RNA-seq from SRR1817386 TH1 Primary_2695 data (Spurlock III *et al.*, 2015). According to Tuxedo protocol (Trapnell *et al.*, 2012), RNA-seq data were aligned using TopHat2 (Kim *et al.*, 2013) and assembled using Cufflinks (Trapnell *et al.*, 2012). Assembled transcripts under one Fragments Per Kilobase of transcript per Million mapped reads (FPKM) were filtered because of low expression. Filtered transcripts were then annotated with lncRNA and protein-coding transcripts from GENCODE. Through the RNA-seq pipeline, 1,210 lncRNA transcripts and 5,233 protein coding transcripts were labeled. The 6,443 labeled transcripts were used as a new test dataset to compare the performance of RNA-seq data.

4.2 Prediction Performance

The empirical optimal hyperparameters were obtained from various combinations based on baseline parameters. The baseline parameters for the number of stacked RNN layers (n_l), the number of RNN hidden units (n_h), and RNN dropout probability (p_d) were 1, 100, and 0.5, respectively. Experiments were performed on Model-Training with a 5-fold cross validation in terms of the average train and test accuracy, and trained by minimizing the categorical cross-entropy with the Adam optimizer (Kingma and Ba, 2014) for 200 epochs. Maximum bucket lengths of 500, 1000, 1500, 2000, 2500, and 3000 were used.

Table 3. Effects of feature information in terms of prediction accuracy

Features	$S, O, l_{\text{ORF}}, r_{\text{ORF}}$	S, O	$S, l_{\text{ORF}}, r_{\text{ORF}}$	S
Training	**0.9857**	0.9077	0.9668	0.8105
Test	**0.9018**	0.8899	0.8331	0.7204

S: transcript sequence, O: ORF indicator, l_{ORF}: ORF length, r_{ORF}: ORF ratio

Table 2 shows the experimental results of the hyperparameter combinations. The number of stacked RNNs layers (n_l) was changed to 0, 1, and 2. The n_l of 0 indicates that the outputs of RNN^S and RNN^O were fed directly into the fully connected layer without additional stacked RNN layers. The n_l of 1, which had one stacked RNN layer, showed the best performance in both training and test accuracy.

The number of RNN hidden units (n_h) was changed to 25, 50, 100, and 200. As the number of RNN hidden units increased, the LSTM memory capacity increased. The n_h of 100 showed the best performance, indicating that the sequence memory capacity of 100 was the most suitable size for distinguishing the lncRNAs from protein-coding transcripts in our experiments.

We changed the dropout probability (p_d) to 0, 0.25, 0.5, and 0.75. The p_d of 0.5 showed the second-highest accuracy in training, but showed the best performance in the generalization test.

Table 3 shows the prediction performance according to the certain features. The model with all features, S, O, l_{ORF}, and r_{ORF}, showed the best performance in terms of average training and test accuracy. The model trained with sequences S and O and the model with ORF scalar features, l_{ORF}, and r_{ORF}, followed. The vanilla RNN, which accepted only preprocessed transcript sequences, S, was not trained properly. According to the results, the ORF features, O, l_{ORF}, and r_{ORF}, are crucial feature information for identifying lncRNAs.

4.3 Performance Comparison between Tools

LncRNAnet was compared with the following existing tools: Coding Potential Calculator (CPC) (Kong *et al.*, 2007), CPAT (Wang *et al.*, 2013), Coding-Non-Coding Index (CNCI) (Sun *et al.*, 2013b), and Predictor of

Table 4. Comparison of prediction performance (data: HT)

	Sensitivity	Specificity	Accuracy	F1-score	AUC
lncRNAnet	0.9591	**0.8766**	**0.9179**	**0.9211**	**0.9672**
CPAT	0.9229	0.8129	0.8679	0.8747	0.9264
CNCI	0.9771	0.7214	0.8493	0.8664	0.8205
CPC	**0.9911**	0.5134	0.7523	0.8000	0.9320
PLEK	0.9840	0.5294	0.7567	0.8018	0.9004

Table 5. Comparison of prediction performance (data: MT)

	Sensitivity	Specificity	Accuracy	F1-score	AUC
lncRNAnet	0.9463	**0.8903**	**0.9183**	**0.9205**	**0.9667**
CPAT	0.9646	0.8157	0.8901	0.8978	0.9530
CNCI	0.9689	0.7883	0.8786	0.8886	0.8610
CPC	**0.9897**	0.5940	0.7919	0.8262	0.9508
PLEK	0.9180	0.5643	0.7411	0.7800	0.8300

Table 6. Accuracy comparison of tools in cross-species

Species	Chicken	Frog	Fruitfly	Zebrafish	Chimpanzee	Cow	Gorilla	Orangutan	Pig	Platypus	Rhesus
lncRNAnet	0.9300	0.8965	0.9085	0.8980	**0.9165**	0.9320	0.9085	**0.9335**	0.9335	0.9050	**0.9270**
CPAT	0.9105	**0.9460**	**0.9545**	**0.9465**	0.8900	**0.9530**	**0.9095**	0.9095	**0.9530**	0.8910	0.9035
CNCI	0.9155	0.8790	0.9510	0.9055	0.9030	0.9165	0.8955	0.8955	0.9215	0.8960	0.9000
CPC	**0.9330**	0.8365	0.9155	0.7985	0.8330	0.9290	0.8745	0.8745	0.9515	**0.9570**	0.8890
PLEK	0.7905	0.7675	0.7320	0.7835	0.7740	0.8255	0.7545	0.7545	0.8035	0.7000	0.7500

long noncoding RNAs and messenger RNAs based on an improved k-mer scheme (PLEK) (Li *et al.*, 2014).

CPC, CNCI, and PLEK are all SVM-based tools using different features. CPC uses sequence features, such as sequence hits from the protein database search, high-scoring segment pairs, and unique manufactured features. CNCI uses adjoining nucleotide triplets (ANT) frequencies. PLEK uses the *k*-mer scheme. On the other hand, CPAT is a logistic regression-based tool that uses four features: ORF size, ORF coverage, Fickett statistic, and hexamer usage bias. The performance analysis of lncRNAnet was compared to the above four tools.

The model of lncRNAnet was trained with the human dataset (Model-Training). The generalization test was performed on the human dataset (HT), the mouse dataset (MT), other cross-species datasets, and the RNA-seq dataset.

4.3.1 Performance Comparison in the Human Dataset

Table 4 shows the prediction performance on dataset HT. The proposed lncRNAnet outperformed the best alternative results in terms of specificity, accuracy, F1-score, and AUC by 7.83%, 5.76%, 5.30%, and 3.78%, respectively. CPC showed the highest sensitivity, 3.34% higher than that of lncRNAnet, and the second-highest AUC, but it showed the lowest performance in terms of specificity, accuracy, and F1-score.

Additionally, we analyzed the effects of sequence length ranges. Fig. 3(a) shows that the performance of lncRNAnet was the most accurate in all length ranges and consistently improved as the length of the sequences increased. On the other hand, the accuracy of other tools fluctuated according to the length variations.

4.3.2 Performance Comparison in a Cross-species Dataset

The dataset MT was used to evaluate the cross-species performance. The models of lncRNAnet and other tools except CPAT were trained on human data, while the model of CPAT was trained on mouse logistic regression.

Table 5 shows that lncRNAnet still showed the best performance, 9.14%, 3.16%, 2.54%, and 1.44% higher than the second-best results in terms of specificity, accuracy, F1-score, and AUC, respectively. These results indicate that the model of lncRNAnet built from human data performed reasonably in a cross-species test. PLEK showed the lowest performance in all metrics.

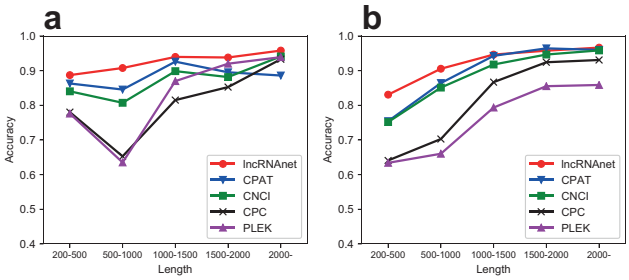

Fig. 3. Accuracy of tools according to length variation in (a) HT and (b) MT

In the length variation analysis, Fig. 3(b) shows that the accuracy of all tools increased steadily as the sequence length increased. However, in terms of overall prediction accuracy, lncRNAnet showed the best performance.

The proposed lncRNAnet showed the best performance in terms of specificity, accuracy, F1-score, and AUC score in both independent test sets, HT, and MT. Furthermore, lncRNAnet performed robustly regardless of the length variations.

Additional cross-species experimental results with other species are listed in Table 6. LncRNAnet only used human data for training. Despite the biased training data, lncRNAnet performed well in regard to non-mammalian species in terms of accuracy, at 0.9300, 0.8965, 0.9085, and 0.8989 for chickens, frogs, fruitflies and zebrafish, respectively. An additional experiment was run using mammalian data. As listed in Table 6, lncRNAnet successfully predicted lncRNAs from mammalian species at 0.9165, 0.9320, 0.9085, 0.9335, 0.9335, 0.9050, and 0.9270 for chimpanzees, cows, gorillas, orangutans, pigs, platypuses, and rhesus monkeys, respectively. Based on these experimental results, we can conclude that lncRNAnet was not overfitted.

4.3.3 Performance Comparison in the RNA-seq dataset

The RNA-seq dataset of SRR1817386 TH1 Primary_2695 data was used to evaluate the performance in the real experimental data. As listed in Table 7, the overall performance of lncRNAnet in terms of accuracy and AUC was the best with 0.8120 and 0.7955, respectively. The F1-score was second highest among the tools, at 0.5411, followed by PLEK. Based on the experimental results listed in Table 7, which followed the Tuxedo

Table 7. Performance comparison of tools in real NGS data

	Sensitivity	Specificity	Accuracy	F1-score	AUC
lncRNAnet	0.5901	0.8634	**0.8120**	0.5411	**0.7955**
CPAT	0.5818	0.8626	0.8099	0.5348	0.7941
CNCI	0.6678	0.8026	0.7773	0.5297	0.7390
CPC	0.2785	**0.9306**	0.8082	0.3529	0.7742
PLEK	**0.7066**	0.7980	0.7808	**0.5477**	0.7775

protocol, we can conclude that lncRNAnet successfully performed in the real NGS dataset.

5 Discussion

Deep learning-based approaches have been successfully applied in the bioinformatics domain (Min *et al.*, 2016). RNN-applied tools based on shorter ncRNAs outperformed existing tools (Lee *et al.*, 2016; Park *et al.*, 2017; Kim *et al.*, 2018). Although the length of the lncRNA transcripts was relatively longer, lncRNAnet also outperformed existing tools.

In the study, there were issues to consider while identifying lncRNAs. One was a sensitivity-specificity trade-off. In a binary classification, it is important to reach both high sensitivity and high specificity. Currently, the number of identified lncRNA transcripts is fairly small compared to the number of protein-coding transcripts. It is important to disregard false lncRNAs, but CPAT, CNCI, CPC, and PLEK had low specificity. They focus more on detecting lncRNAs and misclassify protein-coding transcripts, which require additional steps to filter protein-coding transcripts from predicted lncRNAs. In contrast, lncRNAnet with a balanced and high performance in terms of sensitivity and specificity can help reduce the additional filtering process.

Another issue was the accuracy change depending on the sequence length. When the sequence length increased, the accuracy of the tools, especially CPC and PLEK, increased dramatically. The reason the performance of the other tools increased as the length increased is that the ORF features are dependent on the transcript length. An ORF of a non-coding transcript can be interpreted as a random sequence that is irrelevant to the function of coding. Briefly, a start codon and a stop codon may appear on average every 64 codons. Thus, the length of the ORF has a limit. As a result, the high performance on longer sequences benefits from the length difference between the ORF length and the transcript length. However, a large proportion of lncRNAs are shorter than protein-coding mRNAs, and the performance with shorter lncRNAs must be guaranteed. The proposed lncRNAnet robustly predicted lncRNAs on both short sequences and long sequences, which will contribute to discovering novel lncRNAs.

This study showed that RNNs perform successfully compared to the other approaches. RNNs can be improved by newly developed architectures, such as stack-augmented recurrent nets (Joulin and Mikolov, 2015) and ByteNet (Kalchbrenner *et al.*, 2016). Despite the outstanding performance of deep learning, it has weaknesses. Compared to the traditional machine learning approaches, a neural network is like a black box, which can identify only its input and output. The trained weights in the model can be identified, while the meaning of the features is difficult to interpret. There have been many efforts to analyze deep learning features. If the feature analysis is completed, we can discover new characteristics of lncRNAs.

6 Conclusion

In this study, we proposed an RNN-based method for classifying lncRNAs from protein-coding transcripts. We used a novel feature, an ORF indicator, which is identified by one-dimensional CNNs, to reflect biological knowledge in our model. The proposed lncRNAnet showed 7.83%, 5.76%, 5.30%, and 3.78% improvements over the alternatives in terms of specificity, accuracy, F1-score, and AUC, respectively, on the test set HT. Furthermore, lncRNAnet successfully detected shorter lncRNAs and showed robust performance regardless of sequence length variations. Our method will contribute to the identification of novel lncRNAs from the abundant transcriptome data.

Acknowledgment

This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (Ministry of Science, ICT and Future Planning) [2014M3C9A3063541, 2018R1A2B3001628], in part by a grant of the Korea Health Technology R&D Project through the Korea Health Industry Development Institute (KHIDI), funded by the Ministry of Health & Welfare [HI15C3224], and in part by Samsung Research Funding Center of Samsung Electronics [SRFC-IT1601-05].

References

Achawanantakun, R., et al. (2015). LncRNA-ID: Long non-coding RNA IDentification using balanced random forests. *Bioinformatics*, **31**(24), 3897–3905.

Alexander, R. P., et al. (2010). Annotating non-coding regions of the genome. *Nature Reviews Genetics*, **11**(8), 559–571.

Baldi, P. et al. (2001). Chapter 6. Neural Networks: applications. In *Bioinformatics: The Machine Learning Approach*. MIT press.

Bhan, A. et al. (2014). Long noncoding RNAs: emerging stars in gene regulation, epigenetics and human disease. *ChemMedChem*, **9**(9), 1932–1956.

Bishop, C. M. (2006). Pattern recognition. *Machine Learning*, **128**.

Bu, D., et al. (2011). NONCODE v3. 0: integrative annotation of long noncoding RNAs. *Nucleic acids research*, **40**(D1), D210–D215.

Cho, K., et al. (2014). On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Collobert, R. et al. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Derrien, T., et al. (2012). The GENCODE v7 catalog of human long noncoding RNAs: analysis of their gene structure, evolution, and expression. *Genome research*, **22**(9), 1775–1789.

Dinger, M. E., et al. (2008). Differentiating protein-coding and noncoding RNA: challenges and ambiguities. *PLoS Comput Biol*, **4**(11), e1000176.

Guttman, M., et al. (2009). Chromatin signature reveals over a thousand highly conserved large non-coding RNAs in mammals. *Nature*, **458**(7235), 223–227.

Harrow, J., et al. (2012). GENCODE: the reference human genome annotation for The ENCODE Project. *Genome research*, **22**(9), 1760–1774.

Hochreiter, S. et al. (1997). Long short-term memory. *Neural computation*, **9**(8), 1735–1780.

Joulin, A. et al. (2015). Inferring algorithmic patterns with stack-augmented recurrent nets. In *Advances in Neural Information Processing Systems*, pages 190–198.

Kalchbrenner, N., et al. (2014). A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Kalchbrenner, N., et al. (2016). Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099*.

Kapranov, P. et al. (2012). Dark matter RNA: existence, function, and controversy. *Genomic âŁœDark MatterâŁž: Implications for*

Understanding Human Disease Mechanisms, Diagnostics, and Cures, pages 7–15.

Khomenko, V., et al. (2016). Accelerating recurrent neural network training using sequence bucketing and multi-GPU data parallelization. In *Data Stream Mining & Processing (DSMP), IEEE First International Conference on*, pages 100–103. IEEE.

Kim, D., et al. (2013). TopHat2: accurate alignment of transcriptomes in the presence of insertions, deletions and gene fusions. *Genome biology*, **14**(4), R36.

Kim, H. K., et al. (2018). Deep learning improves prediction of CRISPR–Cpf1 guide RNA activity. *Nature biotechnology*, **36**(3), 239.

Kim, Y. (2014). Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Kingma, D. et al. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kong, L., et al. (2007). CPC: assess the protein-coding potential of transcripts using sequence features and support vector machine. *Nucleic acids research*, **35**(suppl 2), W345–W349.

Kung, J. T., et al. (2013). Long noncoding RNAs: past, present, and future. *Genetics*, **193**(3), 651–669.

LeCun, Y., et al. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, **86**(11), 2278–2324.

Lee, B., et al. (2016). deepTarget: End-to-end Learning Framework for microRNA Target Prediction using Deep Recurrent Neural Networks. *arXiv preprint arXiv:1603.09123*.

Lee, J. T. (2012). Epigenetic regulation by long noncoding RNAs. *Science*, **338**(6113), 1435–1439.

Lertampaiporn, S., et al. (2014). Identification of non-coding RNAs with a new composite feature in the Hybrid Random Forest Ensemble algorithm. *Nucleic acids research*, page gku325.

Li, A., et al. (2014). PLEK: a tool for predicting long non-coding RNAs and messenger RNAs based on an improved k-mer scheme. *BMC bioinformatics*, **15**(1), 1.

Lin, M. F., et al. (2011). PhyloCSF: a comparative genomics method to distinguish protein coding and non-coding regions. *Bioinformatics*, **27**(13), i275–i282.

Liu, J., et al. (2006). Distinguishing protein-coding from non-coding RNAs through support vector machines. *PLoS Genet*, **2**(4), e29.

Mattick, J. S. (2001). Non-coding RNAs: the architects of eukaryotic complexity. *EMBO reports*, **2**(11), 986–991.

Mattick, J. S. et al. (2006). Non-coding RNA. *Human molecular genetics*, **15**(suppl 1), R17–R29.

Mercer, T. R., et al. (2009). Long non-coding RNAs: insights into functions. *Nature Reviews Genetics*, **10**(3), 155–159.

Min, S., et al. (2016). Deep learning in bioinformatics. *Briefings in Bioinformatics*, page bbw068.

O'Leary, N. A., et al. (2015). Reference sequence (RefSeq) database at NCBI: current status, taxonomic expansion, and functional annotation. *Nucleic acids research*, **44**(D1), D733–D745.

Pang, K. C., et al. (2006). Rapid evolution of noncoding RNAs: lack of conservation does not mean lack of function. *Trends in Genetics*, **22**(1), 1–5.

Park, S., et al. (2017). Deep Recurrent Neural Network-Based Identification of Precursor microRNAs. In *Advances in Neural Information Processing Systems*, pages 2895–2904.

Pian, C., et al. (2016). LncRNApred: Classification of Long Non-Coding RNAs and Protein-Coding Transcripts by the Ensemble Algorithm with a New Hybrid Feature. *PloS one*, **11**(5), e0154567.

Ponting, C. P., et al. (2009). Evolution and functions of long noncoding RNAs. *Cell*, **136**(4), 629–641.

Quinn, J. J. et al. (2016). Unique features of long non-coding RNA biogenesis and function. *Nature Reviews Genetics*, **17**(1), 47–62.

Simonyan, K. et al. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Spurlock III, C. F., et al. (2015). Expression and functions of long noncoding RNAs during human T helper cell differentiation. *Nature communications*, **6**, 6932.

Srivastava, N., et al. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, **15**(1), 1929–1958.

Struhl, K. (2007). Transcriptional noise and the fidelity of initiation by RNA polymerase II. *Nature structural & molecular biology*, **14**(2), 103–105.

Sun, K., et al. (2013a). iSeeRNA: identification of long intergenic non-coding RNA transcripts from transcriptome sequencing data. *BMC genomics*, **14**(2), 1.

Sun, L., et al. (2013b). Utilizing sequence intrinsic composition to classify protein-coding and long non-coding transcripts. *Nucleic acids research*, page gkt646.

Trapnell, C., et al. (2012). Differential gene and transcript expression analysis of RNA-seq experiments with TopHat and Cufflinks. *Nature protocols*, **7**(3), 562.

Tripathi, R., et al. (2016). DeepLNC, a long non-coding RNA prediction tool using deep neural network. *Network Modeling Analysis in Health Informatics and Bioinformatics*, **5**(1), 1–14.

Van den Oord, A., et al. (2013). Deep content-based music recommendation. In *Advances in neural information processing systems*, pages 2643–2651.

Wang, K. C. et al. (2011). Molecular mechanisms of long noncoding RNAs. *Molecular cell*, **43**(6), 904–914.

Wang, L., et al. (2013). CPAT: Coding-Potential Assessment Tool using an alignment-free logistic regression model. *Nucleic acids research*, **41**(6), e74–e74.

Wapinski, O. et al. (2011). Long noncoding RNAs and human disease. *Trends in cell biology*, **21**(6), 354–361.

Wilusz, J. E., et al. (2009). Long noncoding RNAs: functional surprises from the RNA world. *Genes & development*, **23**(13), 1494–1504.

Yates, A., et al. (2016). Ensembl 2016. *Nucleic acids research*, **44**(D1), D710–D716.