RNN-VirSeeker: A Deep Learning Method for Identification of Short Viral Sequences From Metagenomes

Fu Liu, Yan Miao, Yun Liu, and Tao Hou

Abstract—Viruses are the most abundant biological entities on earth, and play vital roles in many aspects of microbial communities. As major human pathogens, viruses have caused huge mortality and morbidity to human society in history. Metagenomic sequencing methods could capture all microorganisms from microbiota, with sequences of viruses mixed with these of other species. Therefore, it is necessary to identify viral sequences from metagenomes. However, existing methods perform poorly on identifying short viral sequences. To solve this problem, a deep learning based method, RNN-VirSeeker, is proposed in this paper. RNN-VirSeeker was trained by sequences of 500bp sampled from known Virus and Host RefSeq genomes. Experimental results on the testing set have shown that RNN-VirSeeker exhibited AUROC of 0.9175, recall of 0.8640 and precision of 0.9211 for sequences of 500bp, and outperformed three widely used methods, VirSorter, VirFinder, and DeepVirFinder, on identifying short viral sequences. RNN-VirSeeker was also used to identify viral sequences from a CAMI dataset and a human gut metagenome. Compared with DeepVirFinder, RNN-VirSeeker identified more viral sequences from these metagenomes and achieved greater values of AUPRC and AUROC. RNN-VirSeeker is freely available at https://github.com/crazyinter/RNN-VirSeeker.

Index Terms—Deep learning, human gut, LSTM, metagenomes, virus identification

1 Introduction

Viruses, the most ubiquitous and abundant biological entities on earth [1], play important roles in functions of microbial communities, such as human gut, soil, and ocean microbiomes [2], [3]. Viruses have caused huge mortality and morbidity in human history, such as smallpox [4] and COVID-19 [5]. Different with the traditional virus isolation techniques by laboratory cultures, metagenomic sequencing methods could capture DNA fragments of all species from microbiota, including culturable and unculturable microorganisms [6]. In a metagenome, viral sequences are mixed with these of various microorganisms [7], [8]. Therefore, identifying viral sequences from the plentiful mixture of viral and host sequences is a crucial step in the downstream of viral analysis, such as viral taxonomy, novel viruses discovery and discriminating human-infecting viruses. However, identifying viral sequences from metagenomes is still a rather tough problem because of the small proportion of viral sequences.

To accurately identify viral sequences in metagenomes, a few methods and tools have been developed, which can be roughly categorized into three types: alignment based, gene based and k-mer based methods. Alignment based methods generally aim at finding the similarities between query sequences and known virus reference genomes, such as ViromeScan [9], Kraken [10], MetaPhlAn [11], Metavir [12], Centrifuge [13] and Genome Detective [14]. These methods, however, suffer from the long execution time and memory consumption during the database construction and sequences mapping.

Gene based methods universally identify viral sequences mainly by comparing genes in a query sequence against viral gene databases, and see if there is enough evidence showing that the query sequences carry viral information [15], [16], [17]. For example, VirSorter [18] uses three or more detected genes within a contig to make a prediction. MARVEL [19] extracts three informative genomic features, gene density, strand shifts, and fraction of genes with significant hits against pVOGS database [20], from contig sequences and uses a machine learning method for prediction of bacteriophage sequences in metagenomic bins.

The last category is k-mer based approaches. CLARK [21], a tool based on reduced sets of k-mers, accurately and efficiently classifies viruses from host. VirFinder [7] also extracts the k-mer frequency from contigs, and then inputs them to a previously trained machine learning classifier named "glmnet" to automatically learn k-mer patterns.

Although gene based and k-mer based methods have achieved relatively good performance on identifying viruses from metagenomes, they suffer from low accuracy of identifying short viral sequences. Gene based methods perform poorly on short sequences (<500bp) for the reason that there may not be sufficient information of features in these short sequences. More importantly, gene features are typically designed by human engineers with extensive domain expertise and it remains a difficult task to determine

• *The authors are with the College of Communication Engineering, Jilin University, Changchun, Jilin 130012, China. E-mail: {liufu, liuyun313, ht_happy}@jlu.edu.cn, miaoyan17@mails.jlu.edu.cn.*

Manuscript received 31 May 2019; revised 10 Oct. 2020; accepted 10 Dec. 2020. Date of publication 14 Dec. 2020; date of current version 3 June 2022.
(Corresponding author: Yun Liu.)
Digital Object Identifier no. 10.1109/TCBB.2020.3044575

which features are more appropriate for a given task. K-mer based methods do not rely on gene detecting nor similarity search so that they can predict short viral contigs that contain few or even only partial genes. While as being known that viruses try to mimic host genome sequences to overcome their defense, and k-mer frequency is sparse for short sequences, k-mer based methods suffer a decrease of overall accuracy when identifying short viral sequences.

Recently, several deep learning based methods have been proposed to improve the performance of identifying viruses of short length in metagenomes during the past two years, such as DeepVirFinder [22], ViraMiner [23], PPR-Meta [24], and VirNet [25]. These tools utilize different deep learning neural networks to establish virus identification models, which can learn more high-level features from rather limited sequences and overcome shortcomings of gene based and k-mer based methods. Thus, better performance on detecting short viral sequences has been achieved. However, the accuracy still needs to be further improved.

In the past few years, Long Short-Term Memory (LSTM) [26] network has been achieved tremendous performance in the field of Neural Language Processing (NLP), especially on modeling short sentences with hundreds of words [27]. In view of this superiority, LSTM network is suitable to learn the features of nucleotide sequences of short length. Therefore, to improve the performance on identifying short viral sequences from metagenomes, a novel LSTM based method, namely RNN-VirSeeker, is developed in this paper. First, sequences are encoded by converting (A, T, G, C) to (1, 2, 3, 4). Then each sequence in the training dataset is successively input to LSTM model. The cross entropy loss is calculated by the gap between labels and current output scores generated by a softmax layer. Until the end of training stage, the loss is back propagated in every iteration to update parameters in the model. After several batches of training, RNN-VirSeeker is used to automatically learn the high-level features of query sequences and predict if the sequence is viral or not based on these scores from the softmax layer. Experimental results have shown that RNN-VirSeeker can accurately identify short viral sequences from metagenomes, surpassing three most widely used methods VirFinder [7], VirSorter [18], and DeepVirFinder [22].

The rest of this paper is organized as follows: Section 2 describes the construction of the proposed RNN-VirSeeker and the evaluation criteria used in this paper. Section 3 introduces the details of the datasets and shows experimental results on the testing data set, a CAMI metagenome, and a human gut metagenome. Section 4 discusses the superiority and future improvements of RNN-VirSeeker. A brief conclusion is given in Section 5.

2 Methods

The workflow of RNN-VirSeeker is shown in Fig. 1. It contains three main procedures: preprocessing, recurrent and prediction. First, 9,000 sequences (containing 4,500 viral sequences and 4,500 host sequences) subsampled from NCBI RefSeq genomes were established as a training database. Then sequences from this database were encoded to a vector by transforming (A, T, G, C) to (1, 2, 3, 4). Next each element of the vector was input into LSTM cells successively. A pair of scores for each sequence were calculated by a softmax layer. Loss generated by comparison between scores and labels was back propagated to learn parameters of the model. In testing stage, each query sequence was put through the model, generating two prediction scores by the softmax layer. The higher score indicated the more likely the query sequence was virus or host.

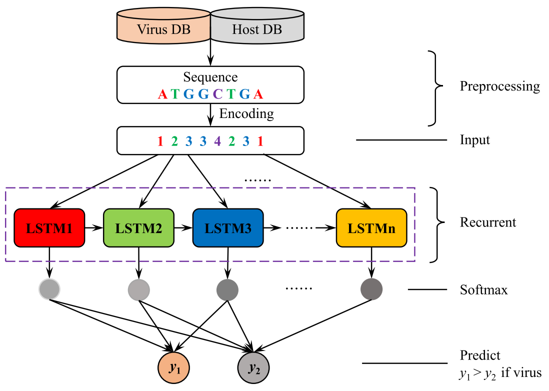

Fig. 1. The workflow of RNN-VirSeeker.

2.1 Construction of RNN-VirSeeker

RNN-VirSeeker is implemented as a LSTM neural network, which is adapted to model complex sequences and has been widely used for many sorts of sequence modeling problems [28], [29], [30], [31], [32]. LSTM can mitigate the short-term memory thanks to the mechanism called “gates” (Fig. 2) that is resistant to the vanishing gradient problem [33], [34]. These gates are different tensor operations that can learn what information to add to or remove from the hidden state. More specifically, the linear temporal path in LSTM that allows gradients to flow more freely across time steps prevents gradients from exploding or vanishing. By doing that, it can pass relevant information down the long chain of sequences. So even information from the earlier time steps can make its way to later time steps, reducing the effects of short-term memory.

According to previous output $\boldsymbol{h}^{(t-1)}$, old state $\boldsymbol{C}^{(t-1)}$ and current input $\boldsymbol{X}^{(t)}$, forget gate $\boldsymbol{f}^{(t)}$, input gate $\boldsymbol{i}^{(t)}$, output gate $\boldsymbol{o}^{(t)}$ and new memory cell $\tilde{c}^{(t)}$ are calculated by following formulas:

$$\boldsymbol{f}^{(t)} = \sigma(\boldsymbol{W}^{(f)}\boldsymbol{X}^{(t)} + \boldsymbol{U}^{(f)}\boldsymbol{h}^{(t-1)}) \tag{1}$$

$$\boldsymbol{i}^{(t)} = \sigma(\boldsymbol{W}^{(i)}\boldsymbol{X}^{t} + \boldsymbol{U}^{(i)}\boldsymbol{h}^{(t-1)}) \tag{2}$$

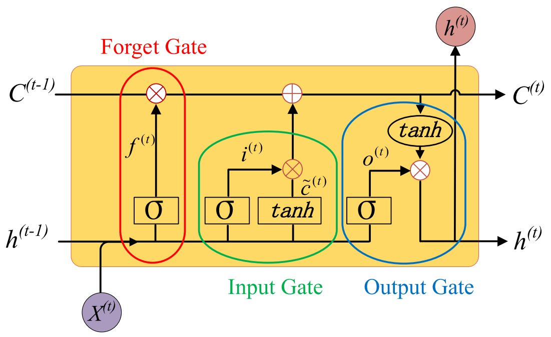

Fig. 2. A memory cell in LSTM.

$$\boldsymbol{o}^{(t)} = \sigma(\boldsymbol{W}^{(o)}\boldsymbol{X}^{(t)} + \boldsymbol{U}^{(o)}\boldsymbol{h}^{(t-1)}) \tag{3}$$

$$\tilde{\boldsymbol{c}}^{(t)} = tanh(\boldsymbol{W}^{(c)}\boldsymbol{X}^{(t)} + \boldsymbol{U}^{(c)}\boldsymbol{h}^{(t-1)}), \tag{4}$$

where $\boldsymbol{W}^{(f)}$, $\boldsymbol{W}^{(i)}$, $\boldsymbol{W}^{(o)}$, $\boldsymbol{W}^{(c)}$ and $\boldsymbol{U}^{(f)}$, $\boldsymbol{U}^{(i)}$, $\boldsymbol{U}^{(o)}$, $\boldsymbol{U}^{(c)}$ are corresponding weights of input $\boldsymbol{X}^{(t)}$ and previous output $\boldsymbol{h}^{(t-1)}$ for forget gate, input gate, output gate and new memory cell, respectively. $\boldsymbol{f}^{(t)}$ decides what information is going to be thrown away from the cell state. $\boldsymbol{i}^{(t)}$ decides which values will be updated. $\boldsymbol{o}^{(t)}$ decides what parts of the cell state are going to be output. $\tilde{\boldsymbol{c}}^{(t)}$ is a vector of new candidate values that could be added to the state. Then the final state values are updated by forget gate, input gate and new memory cell:

$$\boldsymbol{C}^{(t)} = \boldsymbol{f}^{(t)} \times \boldsymbol{C}^{(t-1)} + \boldsymbol{i}^{(t)} \times \tilde{\boldsymbol{c}}^{(t)}. \tag{5}$$

The old state $\boldsymbol{C}^{(t-1)}$ is multiplied by $\boldsymbol{f}^{(t)}$, representing what was decided to forget earlier. Then $\boldsymbol{i}^{(t)} \times \tilde{\boldsymbol{c}}^{(t)}$, a new candidate value, is added. It is scaled by how much we decide to update each state value. The output of the LSTM cell is a filtered version of the cell state:

$$\boldsymbol{h}^{(t)} = \boldsymbol{o}^{(t)} \times tanh(\boldsymbol{C}^{(t)}). \tag{6}$$

The cell state $\boldsymbol{C}^{(t)}$ is put through *tanh* to push the values to be between -1 and 1 and then it is multiplied by the output gate $\boldsymbol{o}^{(t)}$, so that we only output the parts we decided to. $\boldsymbol{h}^{(t)}$ here works as a hidden state vector, representing a short-term memory vector. It is calculated by the output gate to decide which information will be transformed to the next LSTM cell and a higher level LSTM layer [26], [35].

2.2 Evaluation Criteria

The decision made by the classier can be represented in format of confusion matrix, which has four categories: True Positives (TP) are examples correctly labeled as positives; False Positives (FP) refer to negative examples incorrectly labeled as positive; True Negatives (TN) correspond to negatives correctly labeled as negative; and False Negatives (FN) refer to positive examples incorrectly labeled as negative. According to the four values, True Positive Rate (TPR), False Positive Rate (FPR), Precision, Recall and F1 Score are calculated as following standard formulas:

$$TPR = Recall = \frac{TP}{TP + FN} \tag{7}$$

$$FPR = \frac{FP}{FP + TN} \tag{8}$$

$$Precision = \frac{TP}{TP + FP} \tag{9}$$

$$F1\ Score = \frac{2 \times Precision \times Recall}{Precision + Recall}. \tag{10}$$

On the basis of TPR and FPR, receiver operating characteristic (ROC) curve was utilized to evaluate the performance of RNN-VirSeeker. The ROC curve is a graphical plot typically used to illustrate the diagnostic ability of a binary classifier system as its discrimination threshold is varied. The ROC curve was created by plotting TPR against FPR at various threshold settings. Contigs with scores higher than the threshold were deemed to be viral. The area under the receiver operating characteristic curves (AUROC) was used to evaluate the goodness of our predictor in the binary classification task, whereby a higher value indicated a better performance. The value of AUROC is equal to the probability that a predictor will rank a randomly chosen positive instance higher than a randomly chosen negative one. If AUROC is equal to 0.5, the corresponding predictor makes a random classification, while an AUROC score of 1 represents a perfect classifier. The labeled CAMI dataset is highly skewed (the proportion of viral and host sequences is about 1:23). When dealing with highly imbalanced datasets, precision-recall (PR) curves could give a more informative picture of the performance than ROC curves [36]. So PR curve instead of ROC curve was utilized here to compare the performance of four methods. The area under the precision-recall curves (AUPRC) were also utilized to evaluate the prediction performance.

In order to prove the significant improvement of RNN-VirSeeker over the other three methods more comprehensively, Friedman test [37], [38] and Nemenyi [39] test were utilized as statistical measures, which compared the performance of multiple algorithms on multiple datasets. Friedman test ranks the algorithms for every dataset from 1 to *n* separately (*n* represents the number of algorithms). Then average ranks are assigned. The Friedman test compares the average ranks of algorithms as

$$R_j = \frac{1}{N}\sum_j r_i^j, \tag{11}$$

where r_i^j is the rank of the *j*th of *k* algorithms on the *i*th of *N* dataset. When under the hypothesis of all algorithms being equivalent, the Friedman statistic is distributed according to τ_{χ^2}:

$$\tau_{\chi^2} = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right]. \tag{12}$$

Then the Friedman value can be calculated as:

$$\tau_F = \frac{(N-1)\tau_{\chi^2}}{N(k-1) - \tau_{\chi^2}}. \tag{13}$$

The Nemenyi test is used when all classifiers are compared to each other. The performance of two classifiers is significantly different if the corresponding average ranks differ by at least the critical difference:

$$CD = q\alpha\sqrt{\frac{k(k+1)}{6N}}, \tag{14}$$

where critical values $q\alpha$ are based on the Studentized range statistic divided by $\sqrt{2}$ (Table 1) [40].

TABLE 1
Critical Values for the Two-Tailed Nemenyi Test

#classifiers	2	3	4	5	6	7	8	9	10
$q_{0.05}$	1.960	2.343	2.569	2.728	2.850	2.949	3.031	3.102	3.164
$q_{0.10}$	1.645	2.052	2.291	2.459	2.589	2.693	2.780	2.855	2.920

3 Results

3.1 Datasets

The datasets used in this paper are summarized in Table 2. The training and testing datasets contain 9,000 and 1,000 sequences of 500bp, respectively. CAMI dataset is structured by 46 viral sequences and 1,051 host sequences with length between 100bp and 500bp. The real human gut metagenomic datasets with length shorter than 500bp and longer than 5,000bp are respectively utilized to evaluate the performance on identifying short and long viral sequences.

3.1.1 *Virus and Host Genomes Used for Training and Testing*

RefSeq genomes from NCBI as supplied in VirFinder [7], which contain 1,562 virus RefSeq genomes infecting prokaryotes and 4,410 prokaryotic host RefSeq genomes were collected. Since contigs shorter than 500bp were always hard or low-accurately to be identified, all of the RefSeq genomes were split into a number of non-overlapped contigs with a length of 500bp so as to mimic fragmented short metagenomic sequences. Due to the limitation of personal computer memory, 5,000 viral contigs and 5,000 host contigs were randomly subsampled from fragments above respectively. The baseline dataset was then split into two subsets with 9,000 contigs and 1,000 contigs respectively, where the number of viral and host contigs were the same. The 9,000-contigs subset was referred as the training set, and the 1,000-contigs subset as the testing set. The NCBI accession numbers of the viral and prokaryotic RefSeqs can be found in the supplementary materials, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TCBB.2020.3044575, of VirFinder [7].

3.1.2 *A Public Benchmark Dataset From CAMI*

To provide full confidence on the objectivity of the comparison, RNN-VirSeeker was compared on a public benchmark dataset from CAMI Challenge Dataset 3 CAMI_high (https://data.cami-challenge.org/participate). The gold standard assembly of all five samples was downloaded using the cami client java tool (https://data.cami-challenge.org/camiClient.jar). As the superiority of RNN-VirSeeker is to identify viral sequences shorter than 500bp, sequences from the dataset were first compared to NT (Nucleotide Sequence Database) from NCBI website using BLAST (Basic Local Alignment Search Tool) [41] and these shorter than 500bp were extracted from the comparison result. Then these with short length were first mapped to virus RefSeqs and the rest unmapped ones were following mapped to host RefSeqs using program "blastn.exe". Sequences whose E-values lower than 10^{-5} were considered as viral sequences and host sequences respectively. The blast comparison resulted 46 of 8,463 viral sequences and 1,051 of 23,579 host sequences. All of the 1,097 sequences were input to the previously trained models one by one to differentiate viruses. The NCBI numbers, contig lengths and their E-values can be found at https://github.com/crazyinter/RNN-VirSeeker/blob/master/supplementary file/CAMI_blast_result.xlsx.

TABLE 2
The Construction of the Datasets Used in This Paper

Dataset	Number of sequences		Length
	Virus	Host	
Training	4,500	4,500	500bp
Testing	500	500	500bp
CAMI_high	46	1,051	100-500bp
Real human gut samples for short sequences	301	253	100-500bp
Real human gut samples for long sequences	267	282	>5,000bp

3.1.3 *Real Human Gut Metagenomic Dataset*

To further prove the ability of RNN-VirSeeker, a simple real metagenomic dataset were established from a real human gut metagenomic sample, where reads were downloaded from the NCBI short-read archive (accession ID: SRA052203) [42]. Reads from the sample were preprocessed as what had been down for CAMI Challenge Dataset 3 CAMI_high. After the blast comparison, 301 in 17,466 (about 1.73 percent) sequences shorter than 500bp were mapped to viral genomes, which was a little bit lower than the range of viral fraction 4-17 percent [43], previously evaluated for human gut metagenomes. This is mainly because of the incomplete set of reads (only shorter than 500bp) in the human gut metagenomic sample. After being mapped to host RefSeq dataset that contains much more contigs than viral sequences, 7,462 in 17,165 (about 43.47 percent) sequences shorter than 500bp were considered as host. As there were a lot of sequences being mapped to a single host species, one with lowest E-value in each species (253 totally) was selected into test dataset respectively. The distribution of viral and host sequences with various length in test dataset was plotted as a histogram in Fig. 3. The abundance profiles are supported at https://github.com/crazyinter/RNN-VirSeeker/blob/master/supplementaryfile/abundance profiles.xlsx.

3.1.4 *Implementation for Training and Predicting*

Before training, sequences of 500bp were encoded by transforming (A, T, G, C) to (1, 2, 3, 4), respectively. At the training stage, every sequence was successively imported to the model, where each element in a sequence was input to a single LSTM cell. All weights and biases were initialized by randomly extracting from the values assigned to the positive distribution. The model was trained using Adam algorithm [44] with a mini-batch size of 256 to minimize the average multi-task binary cross entropy loss function on the training set. More details about the cross entropy loss function can be read from https://sefiks.com/2017/12/17/a-gentle-introduction-to-cross-entropy-loss-function/. In Adam algorithm, learning rate was set as 0.00022 and β_1, β_2, ϵ (three hyperparameters in Adam algorithm) were set as 0.9, 0.999, 10^{-18} respectively. Mini-batch loss was evaluated

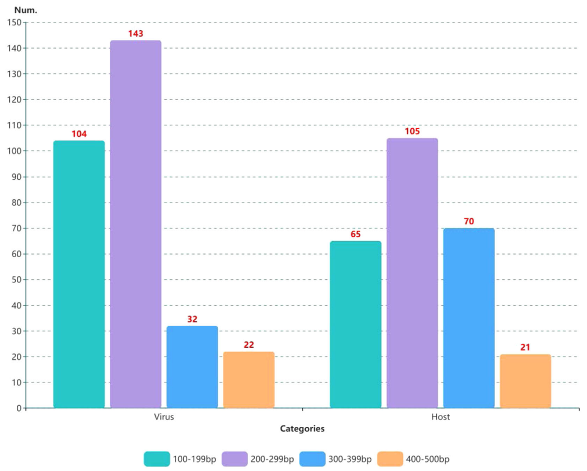

Fig. 3. The distribution of viral and host sequences with various length extracted from a real human gut metagenome.

at the end of each training epoch to monitor convergence. By grid search technique, a traditional technique for hyper-parameters optimization, our model finally contains 64 hidden units and takes 70,000 epochs to be fully trained.

For predicting sequences <500bp, zero padding was used to fill in the input units. Dynamic LSTM was used to deal with inputs of various lengths, where the part zero-padded was omitted in the process of LSTM. The padded sequences were then input to the LSTM cells to generate two scores for each sequence, the higher of which was used to predict whether it was virus or not.

The ability of RNN-VirSeeker was assessed to correctly identify viral sequences of short length (<500bp) in comparison to VirFinder, a well-performed k-mer based viral classification tool at present, VirSorter, a state-of-the art gene-based viral identifier, and DeepVirFinder, a novel deep learning method for viruses detecting, respectively. When identifying short viral sequences, all of the four models were trained using the same dataset containing 9,000 mixed contigs with a length of 500bp as described above. All of the four trained models were tested on the test dataset, CAMI Challenge Dataset 3 CAMI_high and a real human gut metagenome to make a comparison. Our implementation utilized Python 3.6.5 with Tensorflow 1.3.0.

3.2 Performance Comparison on Testing Dataset

Fragments in test dataset containing 500 viral contigs and 500 host contigs were given as input to fully trained RNN-VirSeeker successively. Scores were then calculated for each input sequence. On the basis of these scores and their true labels, all sets of values of FPR and corresponding values of TPR were calculated and a ROC curve was plotted as a red line shown in Fig. 4. To make a comparison to VirFinder, VirSorter and DeepVirfinder, the same testing data as mentioned above were given as input to all of the three trained models and in accordance with scores, their ROC curves were plotted as a blue line, brown line (VirSorter was assessed the TPR and FPR by category I and II VirSorter predictions.), and a green line respectively shown in Fig. 4. We plotted the four ROC curves together so that it was clear to see that AUROC score created by RNN-VirSeeker surpassed these of the others, 0.9175 for RNN-VirSeeker, 0.8931 for

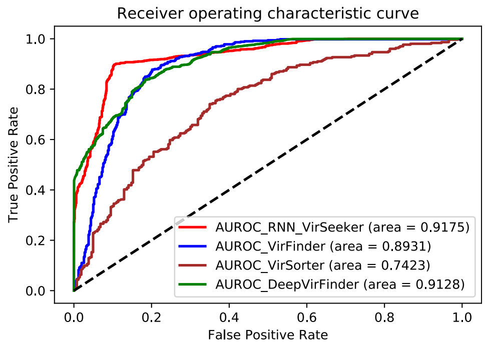

Fig. 4. The ROC curves for VirFinder, VirSorter, DeepVirFinder and RNN-VirSeeker prediction results based on test dataset.

VirFinder, 0.7423 for VirSorter, and 0.9128 for DeepVirFinder. Based on the number of correctly and incorrectly identified viral and host sequences, recalls, precisions and F1 scores of the four methods were calculated (Table 3) to make a comparison on their performance. RNN-VirSeeker outperformed VirSorter and VirFinder at all three criteria, 0.1620 and 0.0220 higher at recall, 0.2342 and 0.0924 higher at precision, 0.1972 and 0.0563 higher at F1 score, respectively. Although there was 0.014 lower than DeepVirFinder at recall, RNN-VirSeeker exceeded DeepVirFinder 0.0378 and 0.0110 at precision and F1 score respectively.

In addition, we compared the TPRs under the condition of the four methods having equivalent FPRs. In order to realize this, we first assessed the TPRs and FPRs for the other three methods. Then the score threshold was selected that produced the same FPR level as the corresponding results to determine the TPR for RNN-VirSeeker. As shown in Table 4, TPRs of RNN-VirSeeker far exceeded that of VirSorter at comparable FPRs. VirSorter only had a TPR value of 0.008 in case of 0.01 FPR. RNN-VirSeeker performed better than VirFinder, correctly identifying 14.75, 16.2, 9.65, 8.87 times more viral contigs when FPR value was 0, 0.002,

TABLE 3
Comparison of Recalls, Precisions and F1 Scores of VirSorter, VirFinder, DeepVirFinder and RNN-VirSeeker on Test Dataset

Criteria	Methods			
	VirSorter	VirFinder	DeepVirFinder	RNN-VirSeeker
Recall	0.7020	0.8420	**0.8780**	0.8640
Precision	0.6869	0.8287	0.8833	**0.9211**
F1 Score	0.6944	0.8353	0.8806	**0.8916**

Bold values represent the best performance.

TABLE 4
The Fraction of True Viral Contigs (TPRs) Identified by RNN-VirSeeker, VirFinder, VirSorter and DeepVirFinder at the Same FPRs of 0, 0.002, 0.006, and 0.01

FPR	TPR			
	VirSorter	VirFinder	DeepVirFinder	RNN-VirSeeker
0	0	0.008	0.096	**0.118**
0.002	0	0.020	0.288	**0.324**
0.006	0	0.040	0.342	**0.386**
0.010	0.008	0.046	0.380	**0.408**

Bold values represent the best performance.

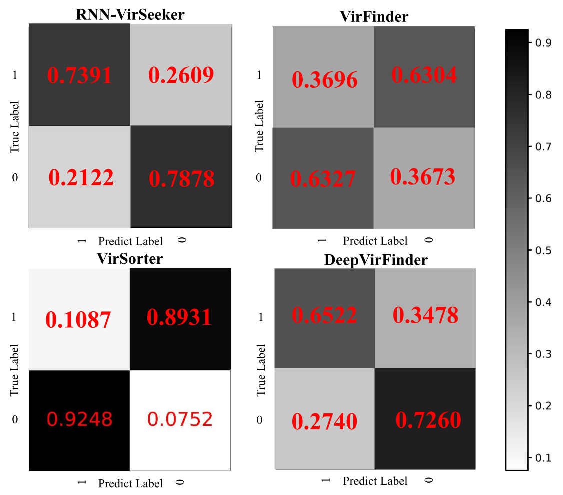

Fig. 5. The normalized 2-class confusion matrices of the four methods. (“1” represents virus and “0” represents host).

0.006 and 0.01 respectively. DeepVirFinder nearly matched the performance with RNN-VirSeeker. The gap of TPR between DeepVirFinder and RNN-VirSeeker was smaller than 0.05 at every level of FPR. All of above represented the effectiveness RNN-VirSeeker on identifying viral contigs of short length.

3.3 Experiments on CAMI Challenge Dataset 3 CAMI_high

We used normalized 2-class confusion matrices (shown in Fig. 5) to show the performance of the four methods. RNN-VirSeeker identified 34 of 46 viral sequences, 17, 29, 4 more than VirFinder, VirSorter, DeepVirFinder respectively. We further calculated the Recalls, Precisions and F1 Scores of the four methods, and made a comparison in Table 5. Of all three criteria from Table 5, RNN-VirSeeker surpassed VirSorter, VirFinder and DeepVirFinder, 0.6304, 0.3595, 0.0869 higher at recall, 0.1272, 0.1074, 0.0380 higher at precision, and 0.2147, 0.1777, 0.0596 higher at F1 Score, respectively.

To further prove the identification performance, PR curves were plotted under the condition of highly imbalanced datasets (Fig. 6). Most parts of the curve of RNN-VirSeeker were above the other three. And AUPRC was 0.4362 for RNN-VirSeeker, 1.90, 3.78 and 9.67 times higher than DeepVirFinder, VirFinder and VirSorter, respectively.

3.4 Performance on a Real Human Gut Metagenomic Dataset

The models of RNN-VirSeeker, VirFinder, VirSorter and DeepVirFinder were respectively used to predict the real

TABLE 5
Comparison of VirSorter, VirFinder, DeepVirFinder and RNN-VirSeeker on the Recalls, Precisions and F1 Scores Based on CAMI Benchmark Dataset

Criteria	Methods			
	VirSorter	VirFinder	DeepVirFinder	RNN-VirSeeker
Recall	0.1087	0.3796	0.6522	**0.7391**
Precision	0.0051	0.0249	0.0943	**0.1323**
F1 Score	0.0097	0.0467	0.1648	**0.2244**

Bold values represent the best performance.

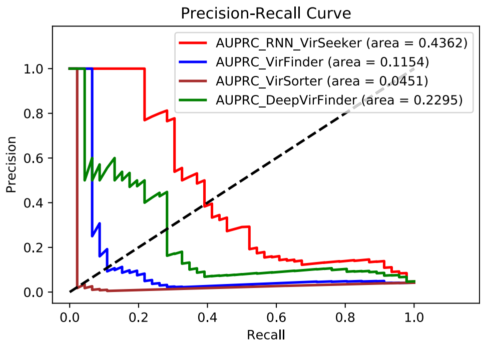

Fig. 6. PR curves of the four methods.

human gut metagenomic contigs of length shorter than 500bp. Totally, RNN-VirSeeker identified 225 of all 301 viral contigs, a little more than DeepVirFinder which identified 214 viral contigs. And RNN-VirSeeker exceeded that of VirFinder identifying 186 viral contigs. VirSorter, a gene-based method, had an extremely poor performance on identifying viruses of short length (totally 6 viral contigs correctly detected). This was probably because of the insufficient complete genes to be detected in short fragments for VirSorter. The ROC curves of RNN-VirSeeker, VirFinder and DeepVirFinder were plotted as red line, blue line and green line in Fig. 7, respectively. Since VirSorter almost identified no virus from the dataset, its ROC curve was excluded here. The AUROC value of RNN-VirSeeker surpassed VirFinder and DeepVirFinder by 0.1183 and 0.0267 respectively. The proportion of correctly identified and misclassified viral sequences was shown in Fig. 8. For all different length of contigs <500bp, RNN-VirSeeker identified more viral sequences than all of the other three methods. The shorter of the contigs, the more contigs were identified correctly. RNN-VirSeeker correctly identified 78 of 104 in length of 100bp-199bp, 111 of 143 in length of 200bp-299bp, 24 of 32 in length of 300bp-399bp and 12 of 22 in length of 400bp-500bp, which were 27.87, 18.09, 20.00, 9.09 percent more viral contigs than VirFinder and 5.41, 3.74, 9.09, 9.09 percent more than DeepVirFinder respectively.

To evaluated the sensitivities of RNN-VirSeeker, recalls, precisions and F1 scores of the four methods were also calculated by the identification results of viral and host sequences in the real human gut metagenomic dataset (Table 6). As expected, RNN-VirSeeker exceeded VirSorter and VirFinder at all three criteria, 0.7276 and 0.1296 higher

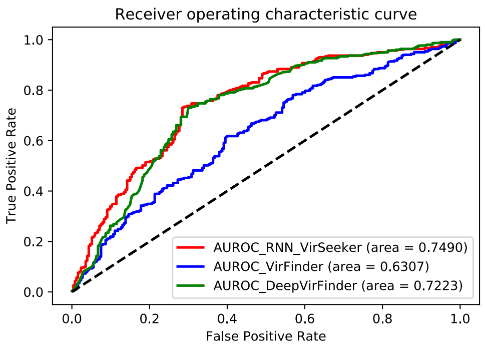

Fig. 7. The ROC curves for VirFinder, DeepVirFinder and RNN-VirSeeker identification results based on the real human gut metagenome.

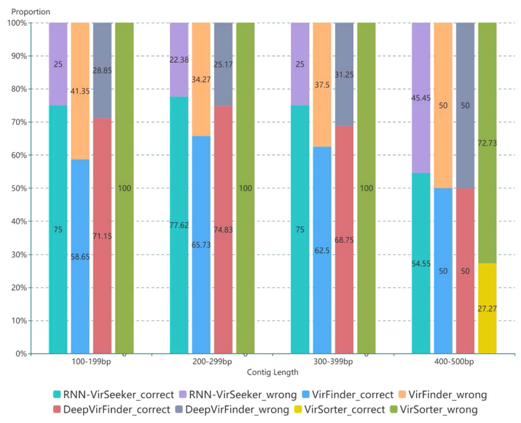

Fig. 8. The proportion of correctly identified viral contigs using the four methods.

at recall, 0.7336 and 0.1096 higher at precision, 0.7307 and 0.1199 higher at F1 score, respectively. DeepVirFinder surpassed RNN-VirSeeker at precision, while 0.0365 and 0.0183 lower at recall and F1 score. This was mainly because RNN-VirSeeker misclassified 4 more host sequences than DeepVirFinder.

3.5 Evaluation of RNN-VirSeeker for Viral Sequences Longer Than 5,000bp

To evaluated the performance of RNN-VirSeeker on identifying viral sequences longer than 5,000bp, the same real human gut metagenomic sample (accession ID: SRA052203) [42] was used to test the ability of our RNN-VirSeeker for long viral sequence identification. Sequences from the sample were preprocessed as what was done in Section 3.1.3. After the comparison by BLASTn, 267 in 2,116 sequences longer than 5,000bp were mapped to viral genomes, and 914 in 2,116 sequences longer than 5,000bp were considered as host. As there were more sequences being mapped to a single host species (47 species totally), sequences with the six lowest E-value in each species (282 sequences totally) were selected into test dataset respectively.

For RNN-VirSeeker, when the sequence length got longer, too long input chain of LSTM might result in memory impairment. For a long sequence (such as 5,000bp), when the tail of the sequence was input to the current LSTM cell, the memory of previously input head part became too weak to play a role in feature contribution. To avoid the memory loss of LSTM, when identifying a long query sequence, the sequence will be divided into several non-overlapped subsequences of 500bp. If the length of the last part in the sequence is shorter than 500bp, the last bases of the sequence will be zero-padded to 500bp and be regarded as a single subsequence. Then all of the subsequences were input to the LSTM cells one after another to get their own scores, the average of which will be the final scores and were contributed to identify whether the query long sequence was viral or not.

To make a comparison, we trained the three benchmark models using fixed length sequences of 3,000bp subsampled from RefSeq genomes, where the number of viruses and host were both 4,500 for training and 500 for testing. The identification results were shown in Table 7.

It is clear to be seen that the performance of RNN-VirSeeker on viral identification for long contigs (longer than 5,000bp) became worse than the others. In our opinion, the diminishment for RNN-VirSeeker may be caused by the segmentation of long sequences, which led to three negative effects. First, when a long sequence was cut off, some useful regions may be separated at the same time. If those regions had relative high weights to contribute to the features of the whole sequence, the operation of cutting off led to losing significant information for identifying viruses. Second, there may be relationships between regions in different parts of a long sequence. When short sequences generated from a long sequence were input into RNN-VirSeeker separately, those relationships were naturally omitted. Lastly, unbalanced feature distribution in a long sequence led to misclassifying according to the average score from short fragments. Although the accuracy of RNN-VirSeeker reduced a little for identifying long contigs, the test performance in the real human gut metagenomic dataset showed that RNN-VirSeeker performed not such worse than VirFinder, VirSorter and DeepVirFinder, 9, 4, 16 less identified viruses and 3.37, 1.50, 6.00 percent lower precision respectively. Considering its better performance on identifying short viral sequences, the gap from RNN-VirSeeker to the other three methods on detecting long viral sequences was acceptable enough.

TABLE 6
Comparison on the Recalls, Precisions and F1 Scores calculated by VirSorter, VirFinder, DeepVirFinder and RNN-VirSeeker on the Real Human Gut Metagenome

Criteria	Methods			
	VirSorter	VirFinder	DeepVirFinder	RNN-VirSeeker
Recall	0.0199	0.6179	0.7110	**0.7475**
Precision	0.0240	0.6480	**0.7589**	0.7576
F1 Score	0.0218	0.6326	0.7342	**0.7525**

Bold values represent the best performance.

TABLE 7
Comparison of Four Methods on Identification Viral Sequences of 5,000bp

	Methods			
	VirSorter	VirFinder	DeepVirFinder	RNN-VirSeeker
Num.	201/267	206/267	**213/267**	197/267
Recall	0.7528	0.7715	**0.7978**	0.7378
Precision	0.7336	0.7574	**0.8068**	0.7011
F1 Score	0.7431	0.7644	**0.8023**	0.7190

Num. represents the number of correctly identified viral sequences. Bold values represent the best performance.

3.6 Friedman Test and Nemenyi Test for the Four Methods on Four Datasets

To evaluated the performance of RNN-VirSeeker on identifying short viral sequences through statistical measures, Friedman test and Nemenyi test were used to compared the performance of the four methods on the three datasets with short sequences ($\leq$ 500bp). First, the F1 Scores of RNN-VirSeeker, VirSorter, VirFinder and DeepVirFinder on the three datasets were collected to rank the four methods. Then the

TABLE 8
Ranks of the Four Methods on Three Datasets With Short Sequences

Datasets	VirSorter	VirFinder	DeepVirFinder	RNN-VirSeeker
Testing	4	3	2	1
CAMI	4	3	2	1
Real human gut samples	4	3	2	1
Average ranks	4.00	3.00	2.00	1.00

The values of F1 Score were used to rank the methods.

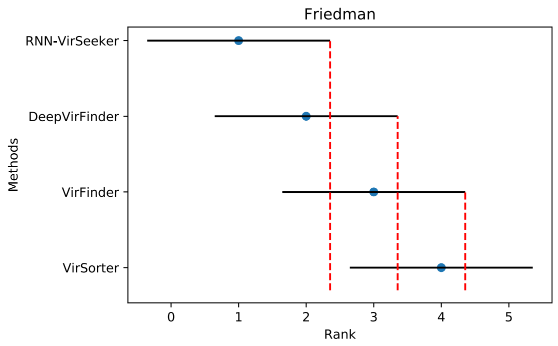

Fig. 9. The result of Friedman test and Nemenyi test on the three datasets with short sequences. The dots represent the average ranks of the four methods, and the length of each horizontal line is equal to the value of CD.

average sorts of the four methods for every dataset were ranked (Table 8). Finally, the Friedman value τ_F was calculated as ∞, which means the assumption that all the four algorithms perform the same is rejected. Thus the critical difference (CD) of Nemenyi test was calculated as 2.7080 with $\alpha = 0.05$. The results of Friedman test and Nemenyi test were plotted in Fig. 9. There is no overlapped region between RNN-VirSeeker and VirSorter, which means that RNN-VirSeeker works significantly better than VirSorter on identifying short viral sequences. Although the differences between RNN-VirSeeker and DeepVirFinder and VirFinder are not significant, the performance of RNN-VirSeeker is better than all of them.

3.7 Comparison of Running Time for RNN-VirSeeker, VirFinder, VirSorter and DeepVirFinder

We timed RNN-VirSeeker and the the other three benchmark methods (VirFinder, VirSorter and DeepVirFinder) in the strategies of training 9,000 contigs of 500bp and testing 1,000 contigs of 500bp from RefSeq genomes. The time consuming of the two strategies were shown at Table 9. It is worth mentioning that RNN-VirSeeker and DeepVirFinder are both deep learning methods, the running time of which are determined extremely by training iterations. The number of epochs when making a comparison in this paper were 70,000 for RNN-VirSeeker and 50 for DeepVirFinder. RNN-VirSeeker took 748 seconds for all 70,000 training iterations and 19 seconds for 1,000 sequences testing. Overall, there were not too much differences among the four methods. The gap between RNN-VirSeeker and the least training time consuming method, VirFinder, was merely 159 seconds. And the testing time consuming of the four methods were very close to each other.

TABLE 9
The Comparison on Time Consuming of RNNVirSeeker, VirFinder, VirSorter and DeepVirFinder

		Methods			
		VirSorter	VirFinder	DeepVirFinder	RNN-VirSeeker
Running Time(s)	Training	748	**589**	651	894
	Testing	19	**18**	**18**	21

Bold values represent the best performance.

4 Discussion

We have shown the architecture, application, and performance of RNN-VirSeeker, a deep learning based method that uses Recurrent Neural Network with LSTM cells, to identify viral contigs of short length (<500bp) in a remarkable accuracy. There are two reasons why RNN-VirSeeker works. First, the loop mechanism in LSTM model can easily get the order of every base in the sequence. Based on this, higher-level features of the sequence will automatically be more exactly extracted, which is intuitively consistent with the expression of DNA sequence in nature. Second, the combination of several “gates” in a LSTM cell decide which part of information from the sequence should be remembered or forgot, which is really important to not only solve the problem of memory loss but also determine which parts of the sequence have more contribution to distinguishing.

Although RNN-VirSeeker is dependent on reference database, the LSTM model could learn higher-level features from limited reference sequences in training set. The learned features may represent much more sequences than those currently existing. As a result, RNN-VirSeeker could correctly identify more viral sequences by restricted reference database.

Although RNN-VirSeeker achieved a better performance than VirSorter, VirFinder and DeepVirFinder, the criteria of precision and F1 Score were relatively low. In our opinion, it may be caused by the limitation of training dataset. As mentioned above, RNN-VirSeeker was trained by 9,000 viral and host sequences, merely a small fraction of RefSeq genomes. Some of more abundant feature information needed to be further learned from a broader training set. If computational consuming were acceptable and having enough time to train the model, RNN-VirSeeker may achieve a better performance by being trained by the whole RefSeq genomes downloaded from NCBI. Furthermore, LSTM in RNN-VirSeeker only considered one direction of the sequences, the another of which may carry some more features. For this reason, Bidirectional LSTM (Bi-LSTM) model appended by Attention Mechanism (AM) can be utilized to further improve the performance of RNN-VirSeeker in our future work.

5 Conclusion

In this paper, we developed a RNN-based method, RNN-VirSeeker, to identify viral sequences from metagenomic data. In comparison to VirFinder, VirSorter and DeepVirFinder on test dataset, CAMI dataset and real human gut metagenomic dataset, RNN-VirSeeker had outstanding performance on short contigs (<500bp) with a competitive result on long ones (>5,000bp). The improvement is a significant progress in the realm of viral identification in metagenomes. RNN-VirSeeker will play crucial roles in expanding our knowledge of natural virus communities as a result that these short contigs usually dominate metagenomic assemblies. We anticipate this method will play a positive role in the downstream viral analysis such as identifying pathogens and viral taxonomy.

Acknowledgments

The authors would like to thank editors, reviewers for the valuable comments. This work has been supported by the National Natural Science Foundation of China (61503151), the Natural Science Foundation of Jilin Province (20160520100JH) and the project funded by China Postdoctoral Science Foundation (2019M651204).

References

[1] Z. Zhang *et al.*, “Rapid identification of human-infecting viruses,” *Transboundary Emerg. Dis.*, vol. 66, pp. 2517–2522, 2019.
[2] B. L. Hurwitz, J. M. U’Ren, and K. Youens-Clark, “Computational prospecting the great viral unknown,” *FEMS Microbiol. Lett.*, vol. 363, no. 10, 2016, Art. no. fnw077.
[3] M. Kimura, Z.-J. Jia, N. Nakayama, and S. Asakawa, “Ecology of viruses in soils: Past, present and future perspectives,” *Soil. Sci. Plant Nutr.*, vol. 54, no. 1, pp. 1–32, 2008.
[4] S. Riedel, “Edward jenner and the history of smallpox and vaccination,” *Proc. Baylor Univ. Med. Center*, vol. 18, no. 1, pp. 21–25, 2005.
[5] W.-J. Guan *et al.*, “Clinical characteristics of 2019 novel coronavirus infection in China,” *New Engl. J. Med.*, vol. 382, pp. 1708–1720, 2020.
[6] S. El-Metwally, T. Hamza, M. Zakaria, and M. Helmy, “Next-generation sequence assembly: Four stages of data processing and computational challenges,” *PLoS Comput. Biol.*, vol. 9, 2013, Art. no. e100345.
[7] J. Ren, N. A. Ahlgren, Y. Y. Lu, J. A. Fuhrman, and F. Sun, “VirFinder: A novel k-mer based tool foridentifying viral sequences from assembled metagenomic data,” *Microbiome*, vol. 5, 2017, Art. no. 69.
[8] J. M. Labonté *et al.*, “Single-cell genomics-based analysis of virus—host interactions in marine surface bacterioplankton,” *ISME J.*, vol. 9, no. 11, pp. 2386–2399, 2015.
[9] S. Rampelli *et al.*, “ViromeScan: A new tool for metagenomic viral community profiling,” *BMC Genomics*, vol. 17, no. 1, pp. 1–9, 2016.
[10] D. E. Wood and S. L. Salzberg, “Kraken: Ultrafast metagenomic sequence classification using exact alignments,” *Genome Biol.*, vol. 15, 2014, Art. no. R46.
[11] D. T. Truong *et al.*, “MetaPhlAn2 for enhanced metagenomic taxonomic profiling,” *Nat. Methods*, vol. 12, pp. 902–903, 2015.
[12] S. Roux *et al.*, “Metavir: A web server dedicated to virome analysis,” *Bioinformatics*, vol. 27, no. 21, pp. 3074–3075, 2011.
[13] D. Kim, L. Song, F. P. Breitwieser, and S. L. Salzberg, “Centrifuge: Rapid and sensitive classification of metagenomic sequences,” *Genome Res.*, vol. 26, no. 12, pp. 1721–1729, Oct. 2016.
[14] M. Vilske *et al.*, “Genome detective: An automated system for virus identification from high-throughput sequencing data,” *Bioinformatics*, vol. 35, pp. 871–873, 2019.
[15] K. E. Wommack *et al.*, “VIROME: A standard operating procedure for analysis of viral metagenome sequences,” *Stand Genomic Sci.*, vol. 6, no. 3, pp. 421–433, Jun. 2012.
[16] B. Buchfink, C. Xie, and D. H. Huson, “Fast and sensitive protein alignment using DIAMOND,” *Nat. Methods*, vol. 12, no. 1, pp. 59–60, Jan. 2015.
[17] D. Paez-Espino, G. A. Pavlopoulos, N. N. Ivanova, and N. C. Kyrpides, “Nontargeted virus sequence discovery pipeline and virus clustering for metagenomic data,” *Nat. Protocols*, vol. 12, pp. 1673–1682, 2017.
[18] S. Roux, F. Enault, B. L. Hurwitz, M. B. Sullivan, “VirSorter: Mining viral signal from microbial genomic data,” *PeerJ*, vol. 3, May 2015, Art. no. e985.
[19] D. Amgarten, L. P. P. Braga, A. M. da Silva, and J. C. Setubal, “MARVEL, a tool for prediction of bacteriophage sequences in metagenomic bins,” *Front. Genet.*, vol. 9, 2018, pp. 304.
[20] A. L. Grazziotin, E. V. Koonin, and D. M. Kristensen, “Prokaryotic virus orthologous groups (pVOGs): A resource for comparative genomics and protein family annotation,” *Nucleic Acids Res.*, vol. 45, pp. D491–D498, 2017.
[21] R. Ounit, S. Wanamaker, T. J. Close, and S. Lonardi, “ CLARK: Fast and accurate classification of metagenomic and genomic sequences using discriminative k-mers,” *BMC Genomics*, vol. 16, 2015, Art. no. 236.
[22] J. Ren *et al.*, “Identifying viruses from metagenomic data using deep learning,” *Quant. Biol.*, vol. 8, no. 1, pp. 64–77, 2020.
[23] A. Tampuu, Z. Bzhalava, J. Dillner, and R. Vicente, “ViraMiner: Deep learning on raw DNA sequences for identifying viral genomes in human samples,” *PLoS One*, vol. 14, no. 9, 2019, Art. no. e0222271.
[24] Z. Fang *et al.*, “PPR-Meta: A tool for identifying phages and plasmids from metagenomic fragments using deep learning,” *Gigascience*, vol. 8, 2019, Art. no. giz066.
[25] A. O. Abdelkareem, M. I. Khalil, M. Elaraby, H. Abbas, and A. H. A. Elbehery, “VirNet: Deep attention model for viral reads identification,” in *Proc. 13th Int. Conf. Comput. Eng. Syst.*, 2018, pp. 623–626.
[26] H. Sepp and S. Jurgen, “Long short-term memory,” *Neural Computation*, vol. 9, pp. 1735–1780, 1997.
[27] Z. C. Lipton, J. Berkowitz, and C. Elkan, “A critical review of recurrent neural networks for sequence learning,” *Comput. Sci.*, vol. 28, no. 1, pp. 1–38, 2015.
[28] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *Proc. 27th Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 3104–3112.
[29] A. Graves, “Generating sequences with recurrent neural networks,” 2013, *arXiv:1308.0850*.
[30] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.
[31] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber, “A novel connectionist system for improved unconstrained handwriting recognition,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 5, pp. 855–868, May 2009.
[32] A. Graves, A. Mohamed, and G. Hinton, “Speech recognition with deep recurrent neural networks,” in *Proc. IEEE Int. Conf. Acoust.*, 2013, pp. 6645–6649.
[33] H. Sepp, “Untersuchungen zu dynamischen neuronalen netzen,” Master’s thesis, Josef Hochreiter Institut fur Informatik, Technische Universitat, Munchen, 1991.
[34] Y. Bengio, P. Simard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.
[35] A. Karpathy, J. Johnson, and L. Fei-Fei, “Visualizing and understanding recurrent networks,” 2015, *arXiv: 1506.02078*.
[36] J. Davis and M. Goadrich, “The relationship between precision-recall and ROC curves,” in *Proc. 23rd Int. Conf. Mach. Learn.*, 2006, pp. 233–40.
[37] M. Friedman, “The use of ranks to avoid the assumption of normality implicit in the analysis of variance,” *J. Amer. Statist. Assoc.*, vol. 32, pp. 675–701, 1937.
[38] M. Friedman, “A comparison of alternative tests of significance for the problem of m rankings,” *Ann. Math. Statist.*, vol. 11, pp. 86–92, 1940.
[39] P. B. Nemenyi, “ Distribution-free multiple comparisons,” PhD thesis, Dept. Mathematics, Princeton Univ., Princeton, NJ, 1963.
[40] J. Demiar and D. Schuurmans, “Statistical comparisons of classifiers over multiple data sets,” *J. Mach. Learn. Res.*, vol. 7, no. 1, pp. 1–30, 2006.
[41] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, “Basic local alignment search tool,” *J. Mol. Biol.*, vol. 215, no. 3, pp. 403–410, 1990.
[42] I. Sharon *et al.*, “Time series community genomics analysis reveals rapid shifts in bacterial species, strains, and phage during infant gut colonization,” *Genome Res.*, vol. 23, no. 1, pp. 111–120, 2013.
[43] S. Minot *et al.*, “The human gut virome: Inter-individual variation and dynamic response to diet,” *Genome Res.*, vol. 21, no. 10, pp. 1616–1625, 2011.

[44] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2015, *arXiv 1412.6980*.

Fu Liu received the BS and MS degrees from the Jilin University of Technology, in 1991 and 1994, respectively, and the PhD degree from the Department of Control Science and Engineering, Jilin University, in 2002. He is currently a professor with Jilin University. His research interests include machine vision, pattern recognition, bioinformatics, and biometrics.

Yan Miao received the BSc degree in automation from the Harbin Institute of Technology (HIT), and the MSc degree in control engineering from Northeast Forestry University (NEFU). He is currently working toward the PhD degree at Jilin University. His research interests include deep learning applications, virus identification, and text classification.

Yun Liu (Member, IEEE) received the BS and PhD degrees from the College of Communication Engineering, Jilin University, in 2011 and 2016, respectively. He is currently with Jilin University. In 2017, he joined the College of Communication Engineering, Jilin University, as a lecturer. He has supervised/participated in several research projects, including the National Natural Science Funds of China and the China postdoctoral Science Foundation. He is the author of more than 20 articles and his areas of research interests include machine learning and bioinformatics.

Tao Hou received the BS degree from the College of Mathematics, and the MS and PhD degrees from the College of Communication Engineering, Jilin University. She is currently a lecturer with Jilin University. Her areas of research interests include machine learning and bioinformatics.