*Bioinformatics*, 38(5), 2022, 1216–1222
https://doi.org/10.1093/bioinformatics/btab845
Advance Access Publication Date: 15 December 2021
Original Paper



Genome analysis

# Virtifier: a deep learning-based identifier for viral sequences from metagenomes

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

College of Communication Engineering, Jilin University, Changchun 130022, China

*To whom correspondence should be addressed.

Associate Editor: Inanc Birol

Received on December 2, 2020; revised on November 13, 2021; editorial decision on December 10, 2021; accepted on December 13, 2021

## Abstract

**Motivation:** Viruses, the most abundant biological entities on earth, are important components of microbial communities, and as major human pathogens, they are responsible for human mortality and morbidity. The identification of viral sequences from metagenomes is critical for viral analysis. As massive quantities of short sequences are generated by next-generation sequencing, most methods utilize discrete and sparse one-hot vectors to encode nucleotide sequences, which are usually ineffective in viral identification.

**Results:** In this article, Virtifier, a deep learning-based viral identifier for sequences from metagenomic data is proposed. It includes a meaningful nucleotide sequence encoding method named Seq2Vec and a variant viral sequence predictor with an attention-based long short-term memory (LSTM) network. By utilizing a fully trained embedding matrix to encode codons, Seq2Vec can efficiently extract the relationships among those codons in a nucleotide sequence. Combined with an attention layer, the LSTM neural network can further analyze the codon relationships and sift the parts that contribute to the final features. Experimental results of three datasets have shown that Virtifier can accurately identify short viral sequences (<500 bp) from metagenomes, surpassing three widely used methods, VirFinder, DeepVirFinder and PPR-Meta. Meanwhile, a comparable performance was achieved by Virtifier at longer lengths (>5000 bp).

**Availability and implementation:** A Python implementation of Virtifier and the Python code developed for this study have been provided on Github https://github.com/crazyinter/Seq2Vec. The RefSeq genomes in this article are available in VirFinder at https://dx.doi.org/10.1186/s40168-017-0283-5. The CAMI Challenge Dataset 3 CAMI_high dataset in this article is available in CAMI at https://data.cami-challenge.org/participate. The real human gut metagenomes in this article are available at https://dx.doi.org/10.1101/gr.142315.112.

**Contact:** miaoyan17@mails.jlu.edu.cn

**Supplementary information:** Supplementary data are available at *Bioinformatics* online.

## 1 Introduction

Viruses, the most ubiquitous and abundant biological entities on earth, have many roles, such as controlling bacterial populations and altering host metabolism, thereby impacting the functions of microbial communities (human gut, soil and ocean microbiomes; Hurwitz *et al.*, 2016). It is believed that only a small fraction of viruses have been discovered (Dutilh *et al.*, 2017), many of which can cause serious diseases. Its frequency and infectivity pose a great threat to human health. Many studies have discovered the associations between the human gut microbiome (viruses and bacteria) and related diseases such as inflammatory bowel disease and colorectal cancer (Hannigan *et al.*, 2017; Mills *et al.*, 2013; Mirzaei and Maurice, 2017). Furthermore, recent emerging viral outbreaks, such as SARS in Asia, COVID-19 worldwide (Wei-Jie et al., 2020) and Ebola in West Africa (Carroll *et al.*, 2015; Gire *et al.*, 2014), have caused tremendous mortality and morbidity in human society. Our knowledge of microbial environments, especially virus communities, has been limited until recently with the development of next-generation sequencing (NGS; Schuster, 2008) methods and computing acceleration. NGS techniques can produce massive quantities of DNA fragments from all species of prokaryotic cells in microbial communities irrespective of the cultivability of these cells (Jie *et al.*, 2017). Viruses are unavoidably extracted at the same time from these samples. Identifying viral sequences from a plentiful mixture of viral and host sequences is an important step in the stream of virus discovery and virus taxonomy. However, this is a challenge because of the small proportion of viruses in the mixed metagenomes and the high mutation rates of viruses.

In order to distinguish viruses from other microorganisms in metagenomes, a variety of methods and tools have been introduced

recently. According to the discriminate criteria, they can be roughly categorized as alignment-based, gene-based, *k*-mer-based and deep learning-based methods. Alignment-based methods aim at matching the similarities between query sequences and known virus reference genomes, such as ProViDE (Tarini *et al.*, 2011), Metavir (Roux *et al.*, 2011), DIAMOND (Buchfink *et al.*, 2015), MetaPhlAn (Truong *et al.*, 2015), Centrifuge (Kim *et al.*, 2016) and Genome Detective (Vilsker *et al.*, 2019). These methods require extensive computation times because of the mechanism and the requirement of large reference databases (Bonnie *et al.*, 2016).

Gene-based methods, such as VIROME (Wommack *et al.*, 2012), VirSorter (Roux *et al.*, 2015) and MARVEL (Amgarten *et al.*, 2018), generally extract several genes from query sequences, and then map these genes against a previously established viral gene database. These methods determine viral sequences by showing that the query sequences have sufficient viral information (Paez-Espino et al., 2017). These methods are not reliable at making predictions for short sequences (<500 bp) because there may be no genes in a short sequence. Moreover, the gene features are usually designed manually based on their experience, and it is difficult to determine which features are more appropriate for the given task (Jie *et al.*, 2017; Lu *et al.*, 2013; Nguyen *et al.*, 2018).

The third type is the *k*-mer-based method. Kraken (Wood and Salzberg, 2014) and CLARK (Ounit *et al.*, 2015) match *k*-mers of query sequences to their pre-built *k*-mer databases. Taxonomer (Flygare *et al.*, 2016) and VirFinder (Jie *et al.*, 2017) utilize the *k*-mer frequency as the feature of a sequence to discriminate the virus from the host. Furthermore, *k*-mer-based methods can identify short viral sequences. However, many viruses can overcome the defense mechanisms of hosts by mimicking their sequences, and the *k*-mer frequency is too sparse to carry enough features when the lengths of sequences are short. Thus, the overall performance of identifying short viral sequences is low for *k*-mer-based methods. In addition, a larger *k* value indicates more dimensions of the features and more computing consumption, even though it appears to increase the performance of feature extraction (Jie *et al.*, 2017). In detail, the 4-mer frequency has a dimension of 136 and the 8-mer frequency has a dimension of 32 896.

The last category is the deep learning-based method. DeepVirFinder (Ren *et al.*, 2020), ViraMiner (Tampuu *et al.*, 2019), PPR-Meta (Zhencheng *et al.*, 2019) and CHEER (Shang et al., 2020) all establish a convolutional neural network (CNN) to automatically learn viral genomic features for the detection of viral sequences from assembled metagenomic sequences. Trained by a large number of sequences, these models achieve relatively high accuracy in identifying viral sequences, especially with short lengths. Although deep learning methods achieve relatively better performance in detecting short viral sequences by learning more high-level features from the sequences of limited lengths, their accuracy still needs to be further improved. Furthermore, the existing CNN-based viral identification tools suffer from insufficient important information of the sequences because of the skip window mechanism and the pooling layers in CNN (Ganapathy *et al.*, 2018).

Nearly all of the deep learning-based methods utilize one-hot coding to encode bases. However, each encoded entity in a nucleotide sequence is independent when encoded by one-hot vectors. This is not in line with the reality. For example, every three bases can be structured to a codon, and bases with a fixed length can be represented as a gene. The obtained features from one-hot encoding are discrete and sparse. The shortcomings of one-hot encoding have a negative effect on the accuracy and the application of identifying viral sequences. As a result, it is much better to express a nucleotide sequence by several meaningful dimensions, each of which contains a continuous real number to represent different degrees instead of binary values.

Considering the tremendous performance achieved by the long short-term memory (LSTM; Hochreiter *et al.*, 1997) neural network in the field of neural language processing (NLP), especially on the modeling of short sentences with hundreds of words (Zachary *et al.*, 2015), the LSTM network has the potential superiority to learn the features of nucleotide sequences with short lengths. To further improve the performance of identifying short viral sequences from metagenomic data, a deep learning-based method, namely Virtifier, is proposed in this article. It includes Seq2Vec, a novel nucleotide sequence encoding model, and an attention-based LSTM neural network for viral prediction. To our knowledge, this is the first time that an attention-based LSTM neural network input by embedded codon vectors is used to identify viral sequences from fragmented metagenomic datasets. The experimental results have shown that Virtifier can accurately identify short viral sequences from metagenomes, surpassing three widely used methods, VirFinder, DeepVirFinder and PPR-Meta.

## 2 Materials and methods

The workflow of Virtifier is shown in Figure 1. First, all sequences from databases are divided into several codon sentences and encoded to several one-hot matrices. The sequences from the whole RefSeq genome are utilized to train a skip-gram (SG) model by back propagation based on the stochastic gradient descent algorithm (Ruder, 2016), where weights between the hidden layer and the output layer are extracted as an embedding matrix. Second, the one-hot encoded sentences are multiplied from the left by the trained embedding matrix to be embedded. Subsequently, each element of the embedded sentence is successively input into the LSTM cells. Filtered by an attention layer and passed through a softmax layer, scores are generated to predict if the query sequence is viral or not. The details are described in the following section.

### 2.1 Virus and host RefSeq genomes for training and testing Virtifier

The performance of deep learning methods is usually sensitive to the data distribution. More bases in a sequence mean more real distribution it presents. When there is a big difference in length between the training and the testing data, the performance may be affected. All of the RefSeq genomes were split into a number of non-overlapped sequences with lengths of 300, 500, 1000, 3000, 5000 and 10 000 bp. Subsequently, 5000 viral sequences and 5000 host sequences of various lengths were randomly subsampled from the fragments above. Each baseline dataset was then split into two subsets with 9000 and 1000 sequences, where the number of viral and host sequences was the same. We referred to the 9000 sequence subset as the training dataset, and the 1000 sequence subset as the testing dataset. The six 9000 sequence subsets with fixed lengths of 300, 500, 1000, 3000, 5000 and 10 000 bp were used to train six models, respectively. The corresponding 1000 sequence subsets were used to test the Virtifier.

### 2.2 Construction of Virtifier

#### 2.2.1 Seq2Vec

Seq2Vec generally tries to map a nucleotide sequence using a codon dictionary to a matrix, similar to what has been done in word2vec technique (Felipe and Geraldo, 2019). The SG algorithm (Tomas et al., 2013a, b) trains a model to predict each word appearing in its context based upon the center word. Its embedding performance surpasses three types of frequency-based embedding vectors, namely, count vector (Deerwester *et al.*, 1990), TF-IDF (term frequency-

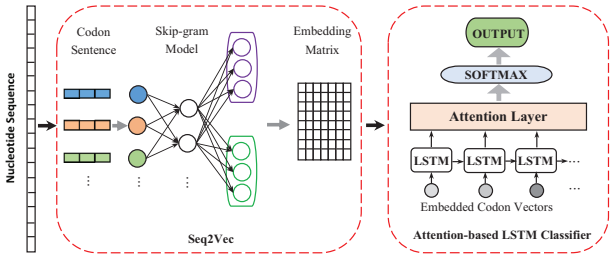


**Fig. 1.** The workflow of Virtifier

inverse document frequency) vector (Salton *et al.*, 1975) and co-occurrence vector (Lund and Burgess, 1996). Thus, the SG model is used to train the embedding matrix for Virtifier.

Before embedding, a nucleotide sequence is preprocessed into a string of codons with a stride of some bases. As an example, take a look at the nucleotide sequence 'AATGCAGTCCAT'. It is first converted into the format of a codon sentence 'AAT, ATG, TGC, GCA, CAG, AGT, GTC, TCC, CCA, CAT' in a stride of one base, where each codon can be regarded as a word in a sentence. The list of all 64 unique codons composes a complete codon dictionary as ['AAA', 'AAT', 'AAG', 'AAC', 'ATA', 'ATT', ...], which is used for codon embedding the latter. One-hot vectors are first used to encode each codon in a codon sentence, where 1 stands for the position where the codon exists in the dictionary and 0 everywhere else. The one-hot encoded codon sentences will then be multiplied from left by an embedding matrix.

In the process of training an embedding matrix, the nucleotide sequence, such as 'ATAGCCTGAAAGCTTGGATTG' in the training dataset with one-hot encoded codons is first transformed to a training set for the SG model with a context window of 1 as the format introduced in Supplementary Table S1. The first column contains all of the input codons for the SG, and the corresponding output codons (context codons) are in the second column. The matrix on the right contains the one-hot encoded form of the input. Next, the input codons in the formed training set are continuously input to the SG model (Supplementary Fig. S1), and then multiplied by the weights between the input layer and the hidden layer into the hidden activations, which are multiplied by the hidden-output weights to calculate the final outputs. Subsequently, the costs calculated by the negative log likelihood between the final outputs and the targets produced in the training set are back-propagated to learn the weights. Finally, the weights between the hidden layer and the output layer are taken as the vector representations of the codons, namely the embedding matrix.

In practice, after 300 000 training steps using the Adaptive Moment Estimation (Adam) (Kingma and Ba, 2014) algorithm with a mini-batch size of 64, the trained embedding matrix was obtained to embed each one-hot codon vector into an embedded vector. In particular, each row vector in the embedding matrix represents an embedded codon vector. Its position is where the codon exists in the established dictionary. The learning rate was selected as 0.0001 and $\beta_1$, $\beta_2$, $\epsilon$ (three hyperparameters in the Adam algorithm) were set as 0.9, 0.999, $10^{-8}$, respectively. The finally trained embedding matrix can be found in https://github.com/crazyinter/Seq2Vec/tree/master/supplementary_files, which can be applied to embed your own sequences.

#### 2.2.2 Attention-based LSTM classifier

LSTM has been demonstrated to be useful in the modeling of sequence data in the field of NLP (Razvan et al., 2014). To mitigate short-term memory (Bengio *et al.*, 1994; Hochreiter, 1991), a mechanism called 'gate' is introduced to constitute LSTM cells (Zachary *et al.*, 2014). It is easy to use LSTM and to choose which input information needs to be added to or deleted from the memory through the gate mechanism, which is realized by a sigmoid layer and a point-by-point multiplication operation. LSTM cells can allow relevant information to be passed down the long chain of sequences through the cell state. A simple LSTM layer consists of an array of recurrently connected memory blocks, each of which contains at least one recurrently connected memory cell that is usually combined by three multiplicative units: input, output and forget gates (Supplementary Fig. S2). As a result, even information from the earlier time steps can make their way to later time steps, reducing the effects of the short-term memory.

However, when an input is encoded into a vector of fixed length, every element of the input shares the same weight, which is inconformity in practice. Since the coding regions are discontinuously distributed in a sequence, each of which contains a set of bases, there is no discrimination to give the same weight to every base of the input, causing model degradation. Derived from human intuition, the attention mechanism (Supplementary Fig. S3) is used to encode sequence data by assigning the importance score for every element of the input. It has been widely applied and has achieved tremendous advancements in diversified applications in NLP, such as machine translation (Bahdanau et al., 2015), sentiment classification (Wang *et al.*, 2017) and question answering (Jiasen et al., 2016).

In practice, three bases in a 'word' (codons) with an embedding size of 20 were applied in the progress of embedding. (This will be discussed in detail in Supplementary Sections S2.1–S2.3.) Every codon in the codon sentences was multiplied by the embedding matrix trained previously. At the training stage, every embedded codon vector was then successively input into an LSTM cell continuously according to the order in every codon sentence. Combined with an attention layer before its final outputs, the LSTM network was trained by the input sequences step by step. All weights and biases were initialized by randomly extracting from the values assigned to the positive distribution. The model was trained using the Adam algorithm with a mini-batch size of 32 to minimize the average multi-task binary cross entropy loss function on the training set. In the Adam algorithm, the learning rate was set as 0.00015 and $\beta_1$, $\beta_2$, $\epsilon$ were set as 0.9, 0.999, $10^8$, respectively. The mini-batch loss was evaluated at the end of each training epoch to monitor convergence. The model contained 64 hidden units with a dropout rate of 0.8 and took 15 epochs to be fully trained.

For the prediction of every query sequence, the LSTM network first extracted each codon vector of the codon sentence as input and then generated two scores through the LSTM units, namely, the attention layer and the softmax units. The higher score indicates a higher possibility that the query sequence is viral or not.

### 2.3 Criteria

Universally, a confusion matrix is calculated to evaluate the performance of a classifier according to four statistics: true positives (TPs), false positives (FPs), true negatives (TNs) and false negatives (FNs). TPs are examples correctly labeled as positives; FPs refer to negative examples incorrectly labeled as positive; TNs correspond to negatives examples correctly labeled as negative; and FNs refer to positive examples incorrectly labeled as negative. Based on the four values, several high-level criteria are further calculated, such as the true positive rate (TPR), false positive rate (FPR), precision, recall and *F*1 score

$$\mathrm{TPR} = \mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}, \tag{1}$$

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}, \tag{2}$$

$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}, \tag{3}$$

$$F1\ \mathrm{Score} = \frac{2 \times \mathrm{Precision} \times \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}. \tag{4}$$

According to the TPR and FPR, a receiver operating characteristic (ROC) curve can be plotted to evaluate the performance of the proposed model. The ROC curve is created by plotting the TPRs against the FPRs at various threshold settings. The area under an ROC curve (AUROC) is utilized to evaluate the prediction performance, where a higher AUROC value indicates a better performance. When dealing with a highly imbalanced dataset, precision–recall (PR) curves give a more informative picture of the performance than ROC curves (Tarini *et al.*, 2011). The area under a PR curve (AUPRC) is also utilized to evaluate the prediction performance on an imbalanced dataset.

## 3 Experiments

### 3.1 Virus and host datasets built from the public metagenomes

The datasets built from the metagenomes used in this article are summarized in Supplementary Table S2.

#### 3.1.1 A public benchmark dataset from CAMI

The gold standard assembly of all five samples from a public benchmark dataset of CAMI (Critical Assessment of Metagenome Interpretation) Challenge Dataset 3 CAMI_high (https://data.cami-challenge.org/participate) were downloaded using the cami client java tool (https://data.cami-challenge.org/camiClient.jar). The sequences from the dataset were first compared to NT (Nucleotide Sequence Database) from the NCBI (National Center for Biotechnology Information) website using Basic Local Alignment Search Tool (BLAST) (Altschul *et al.*, 1990) with default parameters, and these shorter than 500 bp were extracted from the comparison result. These with short lengths were first mapped to the virus RefSeqs and the remaining unmapped ones were mapped to the host RefSeqs using BLASTn with default parameters. The blast comparison resulted in 46 viral sequences and 1051 host sequences with *E*-values lower than $10^{-5}$. All of the 1097 sequences were input to the previously trained models successively to identify the viruses. The NCBI numbers, sequence lengths and their *E*-values can be found at https://github.com/crazyinter/Seq2Vec/tree/master/supplementary_files/CAMI_blast_result.xlsx.

#### 3.1.2 A real human gut metagenomes

Virtifier was applied to a real human gut metagenomic sample, where reads were downloaded from the NCBI short-read archive (accession ID: SRA052203; Sharon *et al.*, 2013). The sequences in the sample were processed according to what was in the CAMI Challenge Dataset 3 CAMI_high. Using BLAST comparisons, 301 of 17 466 sequences shorter than 500 bp and 1270 of 9668 sequences longer than 5000 bp were mapped to viral genomes. After being mapped to the host RefSeq genome, 7462 of 17 165 sequences shorter than 500 bp and 9083 of 26 716 sequences longer than 5000 bp were considered as the host. For short sequences, as there were many sequences being mapped to a single host species, one with the lowest *E*-value in each species (253 in total) was selected in the dataset. The distribution of the viral and host sequences with various lengths (<500 bp) in the real metagenomic dataset was plotted as a histogram in Supplementary Figure S4. The abundance profiles are supported at https://github.com/crazyinter/Seq2Vec/tree/master/supplementary_files. For long sequences, as there were more sequences being mapped to a single host species, sequences with the nine lowest *E*-value in each species were selected into the test dataset. If there were not enough sequences in a specie, all of the sequences with *E*-values lower than $10^{-5}$ were selected. Finally, 1267 host sequences were subsampled. These host sequences combined with the 1270 viral sequences were contained in the long-sequence testing dataset.

### 3.2 Performance on the sequences of different lengths in the training and testing dataset

Figure 2 shows the impact of sequence length on the identification performance of Virtifier. The query test sequences in general receive the highest AUROC scores when the sequence length in the testing set matched those whose network was trained. The performance dropped dramatically when the sequence length in the training stage is longer than the query sequence length. Thus, in the following experiments, sequences of 300- and 500-bp length were, respectively, used to train Virtifier. The two trained models were separately utilized to predict query sequences <300 and 300–500 bp. For sequences in all of the datasets shorter than 300 and 500 bp, zero padding was used to fill in the input units.

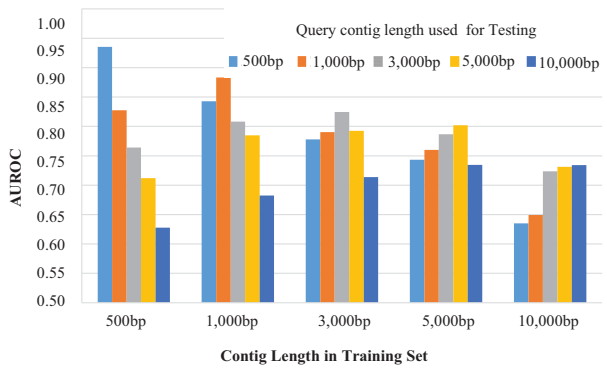


**Fig. 2.** The impact of sequence length from training and testing dataset on the performance of Virtifier

### 3.3 Performance on testing virus and host RefSeq genomes

Virtifier was evaluated by the two groups of the testing datasets with different lengths of short sequences (300 and 500 bp). The ROC curves were plotted as shown in Figure 3. It is clear that the AUROC scores created by Virtifier exceeded these of VirFinder, DeepVirFinder and PPR-Meta in both length (300 bp in Fig. 3a and 500 bp in Fig. 3b), achieving 0.9129 for Virtifier, 0.8290 for VirFinder, 0.8886 for DeepVirFinder, 0.8900 for PPR-Meta in the 300 bp dataset, 0.9354 for Virtifier, 0.8931 for VirFinder, 0.9128 for DeepVirFinder, 09275 for PPR-Meta in the 500 bp dataset.

Recalls, precisions and *F*1 scores of the four methods were calculated to arrive at further comparisons on their performance. Virtifier outperformed VirFinder, DeepVirFinder and PPR-Meta at all three criteria on both 300- and 500-bp lengths. This is shown in Supplementary Table S3.

Some experiments determined that identifying short viral sequences directly from metagenomes could supplement many viral sequences and genes to the long ones and supplement several species that are omitted through the strategy of binning, which is detailed in Supplementary Section S2.6.

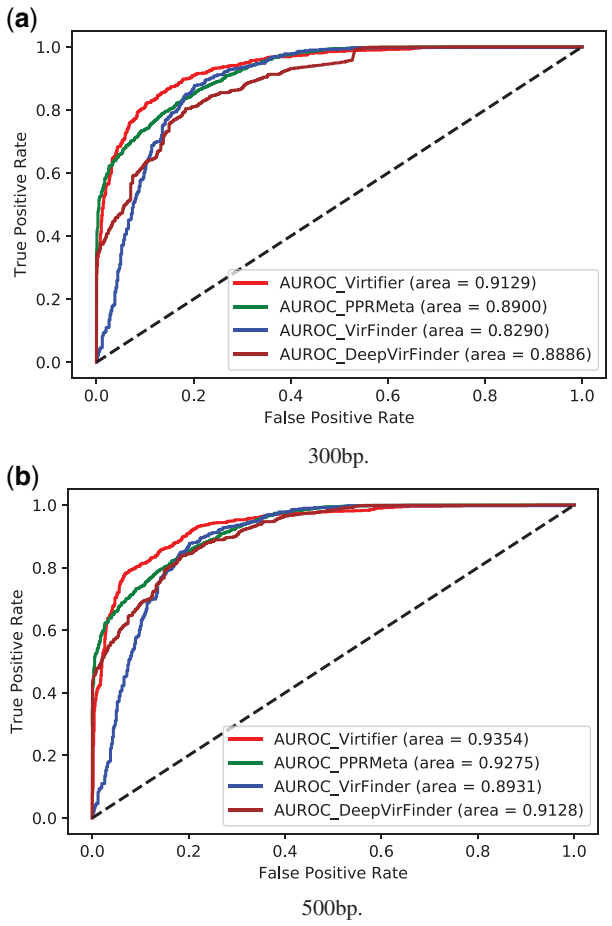


**Fig. 3.** The ROC curves of the classification results on the testing datasets (**a**) 300 bp and (**b**) 500 bp

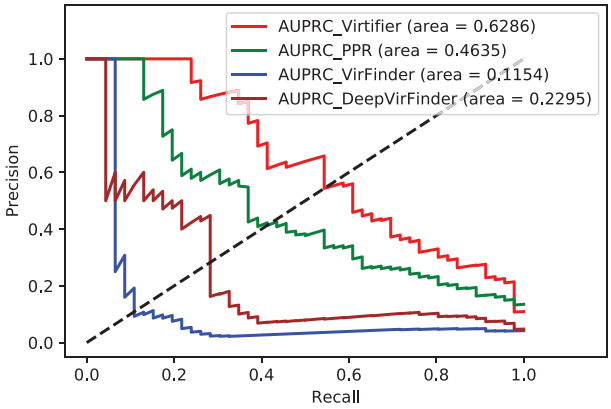


**Fig. 4.** The PR curves of the classification results on the CAMI Challenge Dataset 3 CAMI_high

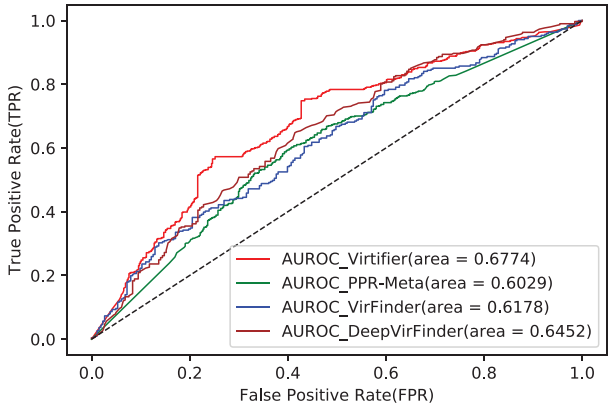


**Fig. 5.** The ROC curves of the classification results on a real human gut metagenomic dataset

### 3.4 Performance on CAMI challenge dataset 3 CAMI_high

Dynamic RNN (Recurrent Neural Network) was used to deal with the inputs of various lengths, where the zero-padded part was omitted and the remaining unpadded part was input. PR curves were plotted under the condition of highly imbalanced datasets (Fig. 4). Nearly all parts of the curve of Virtifier are above the other three. The AUPRC is 0.6286 for Virtifier, which is 5.45, 2.74 and 1.36 times higher than VirFinder, DeepVirFinder and PPR-Meta, respectively. The recalls, precisions and *F*1 scores of the four methods were further calculated, and comparisons are made as shown in Supplementary Table S4. Virtifier outperforms VirFinder, DeepVirFinder and PPR-Meta in all of the three criteria.

### 3.5 Performance on a real human gut metagenomic dataset

The ROC curves of the four methods were plotted as shown in Figure 5. Nearly all parts of the curve of Virtifier are above the other three. The proportion of correctly identified viral sequences is shown in Supplementary Figure S7. For all the different lengths of the sequences shorter than 500 bp, Virtifier identified more viral sequences than VirFinder and DeepVirFinder. Although Virtifier's accuracy of identifying viral sequences with lengths between 100 and 299 bp is a little lower than PPR-Meta's, it performs much better for lengths between 300 and 500 bp, which achieves better results in general. The recalls, precisions and *F*1 scores of the four methods are compared as shown in Supplementary Table S5. Virtifier achieves better performance than VirFinder, DeepVirFinder and PPR-Meta in all of the three criteria.

### 3.6 Sensitivity of Virtifier in the presence of sequencing errors

Usually, sequencing errors include base substitutions and base insertions or deletions (Zhencheng *et al.*, 2019). The tolerances of Virtifier and related tools (VirFinder, PPR-Meta and DeepVirFinder) to the two kinds of sequencing errors were compared. MetaSim (Richter *et al.*, 2008), a reliable sequencing simulator, was used to process the test fragments in the real human gut dataset with 3% base substitutions and 3% base insertions or deletions separately (similar to what was done in PPR-Meta). The same models and settings introduced previously were utilized to test the performance of the four methods in the presence of both types of errors. The proportion of correctly identified viral sequences in terms of sequencing errors in the real human gut metagenome are shown in Supplementary Figure S8. The decrease of the four methods in the 3% base substitutions and 3% base insertions or deletions is shown in Supplementary Figure S9. In each range of sequence length, in terms of 3% base substitutions, the four methods identified fewer viruses than those without sequencing errors. Virtifier still performed the best of the four tools. In the presence of 3% base insertions or deletions, a significant decrease in every sequence length occurred for VirFinder, PPR-Meta and DeepVirFinder. However, there was merely a slightly decrease for Virtifier. In our view, this may be caused by the nature of the Virtifier-codon features. When there are base insertions or deletions in a nucleotide sequence, the order of the divided codons is not changed greatly. For example, when one base is substituted, only three codons are substituted, infecting three LSTM cells. However, for the other three methods, larger *k*-values in VirFinder and larger sizes of the filters in PPR-Meta and DeepVirFinder cause a strong influence on the feature extraction. The influence of the base insertions or deletions is the same as that of the base substitutions.

### 3.7 The performance of Virtifier on identifying long viral sequences

When the sequence length is longer, a long input chain of LSTM in Virtifier may result in memory impairment. To render the function of Virtifier complete, when identifying a long query sequence (>500 bp), it will be divided into several non-overlapped subsequences of 500 bp. If the length of the last part in the sequence is shorter than 500 bp, the last bases of the sequence will be zero-padded and regarded as a single subsequence. Subsequently, all of the subsequences are input to the 500-bp trained Virtifier one after another to get their own scores, the average of which will be the final score and determining whether the query long sequence is viral or not.

All of the models were tested using sequences longer than 5000 bp from the real human gut metagenomic dataset. The identification results are shown in Supplementary Table S6. Virtifier identified more viral sequences than both VirFinder and DeepVirFinder. Although Virtifier identified 4.80% less viral sequences than PPR-Meta, the gap from Virtifier to PPR-Meta for identifying long viral sequences is acceptable.

## 4 Discussions

Almost all existing methods have low accuracy in identifying short viral sequences because the short sequences contain too little information to extract enough features. There are three reasons why Virtifier works. First, Seq2Vec creates a new representation for nucleotide sequences. Each codon is encoded to an embedded vector from the one-hot vector by an embedding matrix. The new representation for nucleotide sequences by Seq2Vec further enriches the features. Second, the loop mechanism in LSTM can easily get the order of every codon in the sequence. Based on this, higher-level features of the sequence will automatically be more precisely extracted, which is intuitively consistent with the expression of the DNA sequence in nature. Third, the combination of several 'gates' in an LSTM cell together with the attention mechanism over LSTM

decides which part of the information from the sequence should be remembered or forgotten, which is really important not only to solving the problem of memory loss but also determining which parts of the sequence contributed more to distinguishing.

To verify the effectiveness of Seq2Vec and LSTM in Virtifier, the sequences from the same training and testing datasets (500 bp) were one-hot encoded by two separate ways: each base in a sequence was one-hot encoded to [1,0,0,0], [0,1,0,0], [0,0,1,0] or [0,0,0,1]; each codon in a sequence was one-hot encoded to a 64-dimension vector. Subsequently, the two one-hot encoded training datasets were used to train two LSTM models, namely base one-hot encoded model (BOEM) and codon one-hot encoded model (COEM). The two models both have the same LSTM model as Virtifier. Another CNN model was trained and tested by the same training and testing datasets. The sequences in the datasets were embedded by Seq2Vec. Every model had the same hyperparameters as the corresponding deep learning model in Virtifier. All parameters were finetuned during the training strategy. The three models were tested by the testing datasets and compared with Virtifier as shown in Supplementary Table S9. The recall, precision and *F*1 score of Virtifier for identifying viral sequences exceeded these of BOEM and COEM by 6.40%, 3.89%, 5.14% and 11.00%, 10.94%, 9.35%, respectively, representing the effectiveness of Seq2Vec. Virtifier was also superior to the single CNN-based model. The gap was 6.20%, 4.54% and 5.61% on recall, precision and *F*1 score, respectively, showing the availability of the LSTM model in Virtifier.

Virtifier could accurately identify short viral sequences (<500 bp) from metagenomes with a comparable performance at longer lengths (>500 bp). It supplements viral analysis since a high proportion of the existing tools tend to deal with long sequences generated from assembling and binning, which misses many viral species. Moreover, identifying viral sequences is the very first step in viral analysis, the effectiveness of which could have an impact on downstream work. Virtifier will play an important role in the realm of virus taxonomy and virus-derived diseases detection.

## 5 Conclusions

A deep learning-based viral identifier, namely Virtifier, is proposed for short sequences from metagenomic data. It includes a novel nucleotide sequence encoding method named Seq2Vec and a variant viral sequence predictor with an attention-based LSTM neural network. The application of Virtifier to the CAMI dataset and real human gut metagenomes prove its outperformance than VirFinder, DeepVirFinder and PPR-Meta in identifying viral sequences of lengths shorter than 500 bp, which is a significant advancement in the realm of virus identification in metagenomes. The identification of viral sequences is the first step in viral analysis, which has an impact on downstream work. Virtifier will play an important role in the realm of virus taxonomy and virus-derived diseases detection.

## Funding

This work was supported by the Youth Science and Technology Talent Support Project of Jilin Province [QT202109] and the project funded by China Postdoctoral Science Foundation [2019M651204].

*Conflict of Interest*: none declared.

## References

Altschul,S.F. *et al.* (1990) Basic local alignment search tool. *J. Mol. Biol.*, **215**, 403–410.

Amgarten,D. *et al.* (2018) MARVEL: a tool for prediction of bacteriophage sequences in metagenomic bins. *Front. Genet.*, **9**, 304.

Bahdanau,D. *et al.* (2015) Neural machine translation by jointly learning to align and translate. *arXiv,* International Conference on Learning Representations, San Diego, Chile.

Bengio,Y. *et al.* (1994) Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on. Neural Networks*, **5**, 157–166.

Bonnie,L.H. *et al.* (2016) Computational prospecting the great viral unknown. *FEMS Microbiol. Lett.*, **363**, fnw077.

Buchfink,B. *et al.* (2015) Fast and sensitive protein alignment using DIAMOND. *Nat. Methods*, **12**, 59–60.

Carroll,M.W. *et al.* (2015) Temporal and spatial analysis of the 20142015 Ebola virus outbreak in West Africa. *Nature*, **524**, 97–101.

Deerwester,S. *et al.* (1990) Indexing by latent semantic analysis. *J. Am. Soc. Inform. Sci.*, **41**, 391–407.

Dutilh,B.E. *et al.* (2017) Editorial: virus discovery by metagenomics: the (im)possibilities. *Front. Microbiol.*, **8**, 1710.

Felipe,A. and Geraldo,X. (2019) Word embeddings: a survey. *arXiv*, https://arxiv.org/abs/1901.09069.

Flygare,S. *et al.* (2016) Taxonomer: an interactive metagenomics analysis portal for universal pathogen detection and host mRNA expression profiling. *Genome Biol.*, **17**, 111.

Ganapathy,N. *et al.* (2018) Deep learning on 1-D biosignals: a taxonomy-based survey. *Yearb. Med. Inform.*, **27**, 98–109.

Gire,S.K. *et al.* (2014) Genomic surveillance elucidates Ebola virus origin and transmission during the 2014 outbreak. *Science*, **345**, 1369–1372.

Hannigan,G.D. *et al.* (2017) Viral and bacterial communities of colorectal cancer. *BioRxiv*, https://doi.org/10.1101/152868.

Hochreiter,S.G. (1991) Untersuchungen zu dynamischen neuronalen netzen. Master's Thesis, Institut fur Informatik, Technische Universitat, Munchen, Germany.

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

Hurwitz,B.L. *et al.* (2016) Computational prospecting the great viral unknown. *FEMS Microbiol. Lett.*, **363**, fnw077.

Jiasen,L. *et al.* (2016) Hierarchical question-image co-attention for visual question answering. The thirtieth Conference on Neural Information Processing Systems, Barcelona, Spain, pp. 289–297.

Jie,R. *et al.* (2017) VirFinder: a novel k-mer based tool for identifying viral sequences from assembled metagenomic data. *Microbiome*, **5**, 69.

Kim,D. *et al.* (2016) Centrifuge: rapid and sensitive classification of metagenomic sequences. *Genome Res.*, **26**, 1721–1729.

Kingma,D.P. and Ba,J. (2014) Adam: a method for stochastic optimization. *arXiv*, https://arxiv.org/abs/1412.6980v8.

Lu,S. *et al.* (2013) Genomic and proteomic analyses of the terminally redundant genome of the *Pseudomonas aeruginosa* phage PaP1: establishment of genus PaP1-like phages. *PLoS One*, **8**, e62933.

Lund,K. and Burgess,C. (1996) Producing high-dimensional semantic spaces from lexical co-occurrence. *Behav. Res. Methods Instrum. Comput.*, **28**, 203–208.

Mills,S. *et al.* (2013) Movers and shakers: influence of bacteriophages in shaping the mammalian gut microbiota. *Gut Microbes*, **4**, 4–16.

Mirzaei,M.K. and Maurice,C.F. (2017) Menage a trois in the human gut: interactions between host, bacteria and phages. *Nat. Rev. Microbiol.*, **15**, 397–408.

Nguyen,T.H. *et al.* (2018) Deep learning for metagenomic data: using 2D embeddings and convolutional neural networks. *arXiv*, https://arxiv.org/abs/1712.00244.

Ounit,R. *et al.* (2015) CLARK: fast and accurate classification of metagenomic and genomic sequences using discriminative k-mers. *BMC Genomics*, **16**, 236.

Paez-Espino,D. *et al.* (2017) Nontargeted virus sequence discovery pipeline and virus clustering for metagenomic data. *Nat. Protoc.*, **12**, 1673–1682.

Razvan,P. *et al.* (2014) Word embeddings: a survey. https://arxiv.org/abs/1312.6026.

Ren,J. *et al.* (2020) Identifying viruses from metagenomic data using deep learning. *Quant. Biol.*, **8**, 64–77.

Richter,D.C. *et al.* (2008) MetaSim-a sequencing simulator for genomics and metagenomics. *PLoS One*, **3**, e3373.

Roux,S. *et al.* (2011) Metavir: a web server dedicated to virome analysis. *Bioinformatics*, **27**, 3074–3075.

Roux,S. *et al.* (2015) VirSorter: mining viral signal from microbial genomic data. *PeerJ*, **3**, e985.

Ruder,S. (2016) An overview of gradient descent optimization algorithms. https://arxiv.org/abs/1609.04747.

Salton,G. *et al.* (1975) A vector space model for automatic indexing. *Commun. ACM*, **18**, 613–620.

Schuster,S.C. (2008) Next-generation sequencing transforms today's biology. *Nat. Methods*, **5**, 16–18.

Shang,J. *et al.* (2020) CHEER: HierarCHical taxonomic classification for viral mEtagEnomic data via deep leaRning. *Methods*, **189**, 95–103.
Sharon,I. *et al.* (2013) Time series community genomics analysis reveals rapid shifts in bacterial species, strains, and phage during infant gut colonization. *Genome Res.*, **23**, 111–120.
Tampuu,A. *et al.* (2019) ViraMiner: deep learning on raw DNA sequences for identifying viral genomes in human samples. *PLoS One*, **14**, e0222271.
Tarini,S.G. *et al.* (2011) ProViDE: a software tool for accurate estimation of viral diversity in metagenomic samples. *Bioinformation*, **6**, 91–94.
Tomas,M. *et al.* (2013a) Efficient estimation of word representations in vector space. *arXiv,* https://arxiv.org/abs/1301.3781.
Tomas,M. *et al.* (2013b) Distributed representations of words and phrases and their compositionality. arXiv. https://arxiv.org/abs/1310.4546.
Truong,D.T. *et al.* (2015) MetaPhlAn2 for enhanced metagenomic taxonomic profiling. *Bioinformatics*, **12**, 902–903.
Vilsker,M. *et al.* (2019) Genome detective: an automated system for virus identification from high-throughput sequencing data. *Bioinformatics*, **35**, 871–873.
Wang,W. *et al.* (2017) Coupled multi-layer attentions for co-extraction of aspect and opinion terms. Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, San Francisco California, USA.
Wei-Jie,G. *et al.* (2020) Clinical characteristics of 2019 novel coronavirus infection in China. *medRxiv*, doi.org/10.1101/2020.02.06.20020974.
Wommack,K.E. *et al.* (2012) VIROME: a standard operating procedure for analysis of viral metagenome sequences. *Stand. Genomic Sci.*, **6**, 421–433.
Wood,D.E. and Salzberg,S.L. (2014) Kraken: ultrafast metagenomic sequence classification using exact alignments. *Genome Biol.*, **15**, R46.
Zachary,C.L. *et al.* (2014) A critical review of recurrent neural networks for sequence learning. arXiv. https://arxiv.org/abs/1506.00019v4.
Zhencheng,F. *et al.* (2019) PPR-Meta: a tool for identifying phages and plasmids from metagenomic fragments using deep learning. *GigaScience*, **8**, 1–14.