OXFORD

Briefings in Bioinformatics, 2022, **23(4)**, 1–11
https://doi.org/10.1093/bib/bbac258
Advance access publication date: 30 June 2022
Problem Solving Protocol

Accurate identification of bacteriophages from metagenomic data using Transformer

Jiayu Shang, Xubo Tang, Ruocheng Guo and Yanni Sun
Corresponding author. Yanni Sun, Department of Electrical Engineering, City University of Hong Kong, Kowloon, Hong Kong (SAR), China. E-mail: yannisun@cityu.edu.hk

Abstract

Motivation: Bacteriophages are viruses infecting bacteria. Being key players in microbial communities, they can regulate the composition/function of microbiome by infecting their bacterial hosts and mediating gene transfer. Recently, metagenomic sequencing, which can sequence all genetic materials from various microbiome, has become a popular means for new phage discovery. However, accurate and comprehensive detection of phages from the metagenomic data remains difficult. High diversity/abundance, and limited reference genomes pose major challenges for recruiting phage fragments from metagenomic data. Existing alignment-based or learning-based models have either low recall or precision on metagenomic data.
Results: In this work, we adopt the state-of-the-art language model, Transformer, to conduct contextual embedding for phage contigs. By constructing a protein-cluster vocabulary, we can feed both the protein composition and the proteins' positions from each contig into the Transformer. The Transformer can learn the protein organization and associations using the self-attention mechanism and predicts the label for test contigs. We rigorously tested our developed tool named PhaMer on multiple datasets with increasing difficulty, including quality RefSeq genomes, short contigs, simulated metagenomic data, mock metagenomic data and the public IMG/VR dataset. All the experimental results show that PhaMer outperforms the state-of-the-art tools. In the real metagenomic data experiment, PhaMer improves the F1-score of phage detection by 27%.

Keywords: phage identification, protein cluster-based token, transformer, deep learning

Introduction

Bacteriophages (phages for short) are viruses infecting bacteria. They are highly ubiquitous and are widely regarded as the most abundant organisms on Earth [1]. There is accumulating evidence showing phages' significant impacts on various ecosystems [2, 3]. Phages play an essential role in regulating microbial system dynamics by limiting the abundance of their hosts and mediating gene transfer. For example, marine viruses can lyse 20–40% of bacteria per day in marine ecosystems [4]. In addition, by regulating the bacteria inhabiting human body sites, phages can also influence human health [5, 6]. An important application of phage is phage therapy, which uses phages as antimicrobial agents to treat bacterial infections [7]. It has gained a resurgence of attention because of the fast rise of antibiotic-resistant bacterial infections.

However, despite the importance of phages to both environmental and host-associated ecosystems, our knowledge about this vast, dynamic and diverse population is very limited. Previously, the limitation is partially caused by the need of the host cell cultivation in labs. Recently, metagenomic sequencing, which allows us to obtain all genetic materials directly from a wide range of samples regardless of cultivation [8–10], has largely removed this limitation and becomes the major means for new phage discovery. According to the NCBI Reference Sequence Database (RefSeq), the number of newly released phages is doubled from 2126 in 2019 to 4410 in 2021. Despite the rapid growth of the phage genomes in RefSeq, the number of known phages is only the tip of the iceberg compared with those in the biome [11]. The uncharacterized phages comprise a big portion of the 'dark matter' in metagenomic composition analysis. Due to the lack of universal marker genes, phages cannot be easily and comprehensively identified [12] using conventional methods.

There are two main challenges for phage identification in metagenomic data. First, both lytic and temperate phages can integrate the host genetic materials into

Jiayu Shang received his bachelor's degree from Sun Yat-sen University. He is now pursuing his Ph.D. degree at the City University of Hong Kong. His research interest is bioinformatics, with a focus on algorithm design for analyzing microbial sequencing data.
Xubo Tang is currently pursuing his Ph.D. degree at the City University of Hong Kong. His main research interests include computational biology and bioinformatics, particularly applying deep learning models to analyze genomic data.
Ruocheng Guo is currently a machine learning researcher at Bytedance AI Lab, London, UK. He got his Ph.D. from Arizona State University, USA. His research interests include causal inference and machine learning.
Yanni Sun is currently an associate professor in the Department of Electrical Engineering at the City University of Hong Kong. She got her Ph.D. in Computer Science and Engineering from Washington University in Saint Louis, USA. Her research interests are sequence analysis and metagenomics.
Received: March 10, 2022. **Revised:** May 22, 2022. **Accepted:** June 4, 2022

their genomes, leading to local sequence similarities between the genomes of phages and bacteria [13]. For example, ~76% phages with known hosts in the RefSeq database have detectable alignments (E-value $<1e$-5) with their host genomes [14]. These common regions pose challenges for distinguishing phages from their bacterial hosts. Second, although some previous works use phage structure-related genes as hallmark genes for phage identification, those genes only account for a small set of the proteins encoded by all phages. Using a small set of hallmark genes can lead to low recall of phage identification. The large gene set coded by phages is further compounded by the fact that many newly identified phages contain genes without any functional annotation. For example, *Caudovirales*, the largest order of phages containing about 93% of sequenced phages, has about 187 006 hypothetical proteins. It is not trivial to identify hallmark genes without functional annotation.

Related work

Many attempts have been made for phage identification [15]. According to the algorithm design, they can be roughly divided into two groups: alignment-based [16, 17] and learning-based [18–21]. The alignment-based methods utilize DNA or protein sequence similarity as the main feature to distinguish phages from other sequences. For example, MetaPhinder [17] uses BLAST hits against a phage reference database to identify phage sequences. VirSorter [16] constructs a phage protein family database and applies hidden Markov model-based search to identify the protein clusters in input contigs. Then, enrichment and depletion metrics are computed to estimate the likelihood of input contigs being phages. However, the limitations of alignment-based methods are apparent. First, bacterial contigs can have multiple alignments with phage genomes, which will lead to false-positive phage predictions. Second, novel or diverged phages might not have significant alignments with the chosen phage protein families (e.g. selected hallmark genes), leading to low sensitivity for new phage identification.

To overcome the limitations of alignment-based methods, several learning-based tools have been proposed for phage identification. These learning models are mainly binary classification models with their training data containing phages and bacteria. Some learning models use extracted sequence features such as k-mers, while others use automatically learned features in deep learning models. For example, VirFinder [18] utilized k-mers to train a logistic regression model for phage detection. Virtifier [22] uses the codon-based features to train a long short-term memory classifier for read-level phage identification. Seeker [20] and DeepVirFinder [19] encode the sequence using one-hot embedding and train a long short-term memory model and convolutional neural network, respectively. PPR-meta [21] is a three-class classification model with predictions as phages, plasmids, and choromosomes. It uses both one-hot embedding and k-mers to train a convolutional neural network. VirSorter2 [23] employs a random forest model on sequence features, such as HMM alignment scores and GC content.

Despite the promising results, a third-party review [15] shows that the precision of these tools drops significantly on real metagenomics data. Many bacterial contigs are misclassified as phages. There are two possible reasons behind this. First, current learning models did not carefully address the challenge that phages and bacteria can share common regions. As a result, the training data did not include sufficient hard cases to train the model. For example, to construct a balanced training data, these tools often randomly select a subset of bacterial segments as negative samples. These samples may not share any local similarities with the phages and thus the trained model cannot generalize to more complicated and heterogeneous data such as real metagenomic data. Second, current models need to crop the genomes into short segments for training. The extracted features are limited to the segment and larger context information from the phage genomes cannot be effectively incorporated.

Overview

In this work, we present a method, named PhaMer, to identify phage contigs from metagenomic data. Because previous works have shown the importance of protein composition for phage classification [24, 25], we employ a contextualized embedding model from natural language processing (NLP) to learn protein-associated patterns in phages. Specifically, by converting a sequence into a sentence composed of protein-based tokens, we employ the embedding model to learn both the protein composition and also their associations in phage sequences. First, we will construct the vocabulary containing protein-based tokens, which are essentially protein clusters with high similarities. Then, we apply DIAMOND BLASTP [26] to record the presence of tokens in training phage sequences (Figure 1 A). Then, the tokens and their positions will be fed into Transformer (Figure 1 B) for contextual-aware embedding. The embedding layer and the self-attention mechanism in Transformer enable the model to learn the importance of each protein cluster and the protein–protein associations. Although Transformer has been used for sequence embedding based on k-mers and motifs [27–29], we are the first in using protein clusters as tokens in Transformer for phage identification. In addition, using the phages' host genomes as the negative samples in the training data, the model can learn from the hard cases and thus is more likely to achieve high precision in real data. Finally, PhaMer can directly use the whole sequences for training, avoiding the bias of segmentation. We rigorously tested PhaMer on multiple datasets covering different scenarios including the RefSeq dataset, short contigs, simulated metagenomic data, mock metagenomic data and the public IMG/VR dataset. We compared PhaMer with four competitive learning-based tools (Seeker, DeepVirFinder, VirFinder and PPR-meta) and one alignment-based tool (VirSorter) based on a third-party review [15].

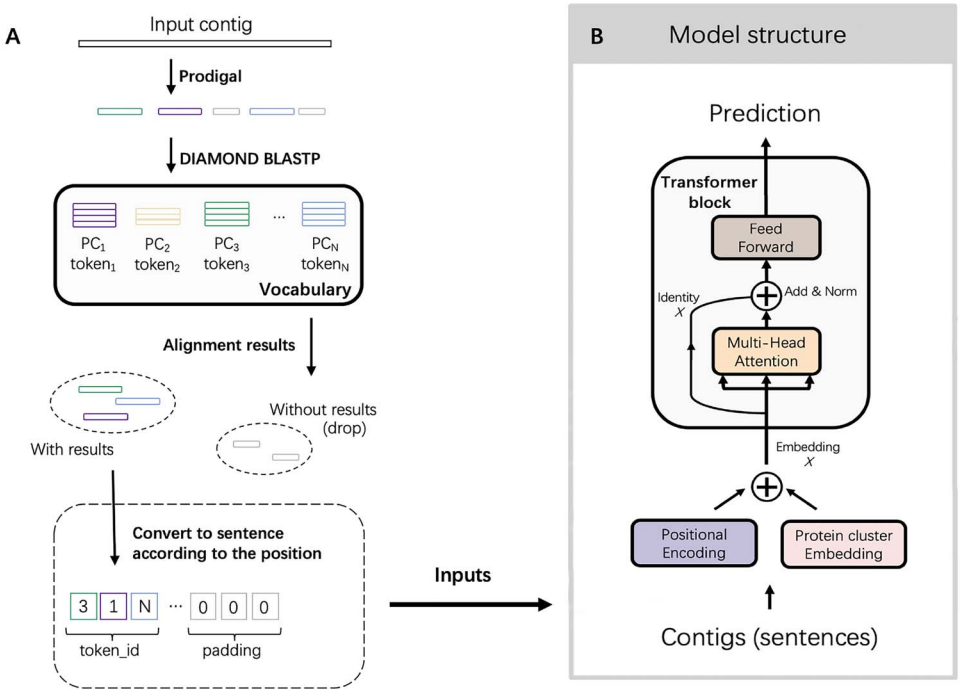

Figure 1. The pipeline of PhaMer. **(A)**: converting inputs into protein-token sentences. During training, we apply Prodigal to predict open reading frames (ORFs) from training phages and translate ORFs into proteins. Then a clustering method is applied to generate protein clusters (PCs), which are the tokens in Transformer. During the test/usage, PhaMer takes contigs as input and converts them into protein-token sentences. **(B)** Transformer network architecture. The converted sentences are fed into the Transformer model and a prediction is made.

Our experimental results show that PhaMer competes favorably against the existing tools. In particular, on the mock metagenomic data, the F1-score of PhaMer exceeds other tools by 27%.

Methods

Inspired by semantic analysis problems in NLP, we employ the state-of-the-art contextualized embedding model, Transformer, to automatically learn abstract patterns from the 'language' of phages. In this language, the contigs are regarded as sentences defined on a phage-aware vocabulary. There are three major advantages behind this formulation. First, some proteins play critical roles in phages' life cycle. For example, coat proteins and receptor-binding proteins can help us distinguish phages from bacteria. These proteins can act as strong signals similar to the words describing obvious emotions in human language. Second, proteins often interact with other proteins to carry out biological functions [30]. Similar to multiple words that can form phrases with different meanings, some protein combinations in the contigs can also provide important evidence for phage identification. Third, using protein-based tokens allows us to integrate much larger context, including the whole phage genome, into feature embedding. Unlike existing learning-based tools that often split the genomes into segments of fixed length, our model can effectively employ proteins in the whole genomes. These features prompt us to convert contigs into protein-based sentences.

In order to automatically learn the importance of proteins and their associations, we adapt the Transformer model to phage identification task. Transformer has achieved the state-of-the-art performance on a variety of NLP problems [31–33]. In particular, the positional encoding and self-attention mechanism enable the model to learn both the importance of each word and the relationships between words.

In the following sections, we will first introduce how we construct the protein-cluster tokens and encode the sequences into sentences. Then, we will describe the Transformer model optimized for phage identification.

Constructing the protein-cluster tokens

Each token in our model is derived from a protein cluster, which contains homologous protein sequences from sequenced phages.

Generating protein clusters Our protein clusters are constructed on the training data. Specifically, they are extracted from 2126 phage genomes released before December 2018 from the RefSeq database, which constitute our training phages. More recently sequenced phages are used as test samples. Constructing the protein vocabulary using only the training sequences allows us to rigorously test our method in scenarios where newly sequenced phages harbor novel proteins outside the vocabulary. Although there are available gene annotations and their corresponding proteins for the reference genomes, we did not use the annotation. Instead, in order to be consistent with the gene prediction process of the test sequences, we apply gene finding and protein translation for the downloaded DNA genomes. According to a recent review of gene finding in viruses [34], Prodigal outperforms other annotation

tools, especially for phages. Thus, we use Prodigal to predict ORF on both training and test genomes under its default parameters. Second, we will run all-against-all DIAMOND BLASTP [26] on the predicted proteins. Protein pairs with alignment E-value $\leq$ 1*e*-3 are used to create a protein similarity network, where the nodes represent proteins and the edges represent the recorded alignments. The edge weight encodes the corresponding alignment's E-value. Then, Markov clustering algorithm (MCL) [35] is employed to group similar proteins into the clusters using default parameters. All the clusters that contain fewer than two proteins are removed and finally we have 45 577 protein clusters. The size distribution of the protein clusters can be found in FigS. 1 in the [Supplementary file 1].

Encoding a contig into a protein token sentence With the generated protein clusters as the tokens in our vocabulary, we will use them to convert contigs into sentences. As shown in Figure 1 A, we will employ Prodigal for gene finding and translation. Then, we will identify the matched protein clusters for the translated proteins by conducting similarity search. Specifically, DIAMOND BLASP is applied to compare each translation against all the proteins in the clusters. We identify the reference protein incurring the smallest E-value and assign the query with this reference protein's cluster. We will record both the ID of the token (protein cluster) and the position of the protein in the query sequence. If an input sequence has no alignment with any token, we will not keep it for downstream analysis. Thus, if a new phage does not contain any of the tokens in our established vocabulary, it will be missed by our model and will be recorded as a false negative. In our experiments, we found that this type of phages is very rare. Most of them contain some tokens.

Because the lengths of the contigs can vary a lot, the converted sentences also contain different number of protein-cluster tokens. We follow the original paper of Transformer [31] and set the maximum length of the sentence to be 300. If the sequence contains more than 300 protein clusters, we will only keep the first 300. For sequences containing less than 300 tokens, we will pad zeros at the end of the sentence. Finally, we will generate a 300-dimensional vector for the input sequence and each dimension encodes a token ID. For example, in Figure 1 A, we show a sentence containing three tokens: $token_1$ (PC_1), $token_3$ (PC_3) and $token_N$ (PC_N). The other positions in this sentence are padded with zeros. The maximum length of the sentence is a hyperparameter and can be set by users.

The Transformer model

The model's inputs are the converted sentences, represented by 300-dimensional vectors, and the output is a score representing how likely the input contig is a phage. The main purpose of Transformer is to automatically learn whether these sentences contain essential features for phage identification: the marker tokens (important proteins) and phrases (protein–protein associations). Two main components in Transformer contribute to these aims: (1) the embedding layers and (2) the self-attention mechanism.

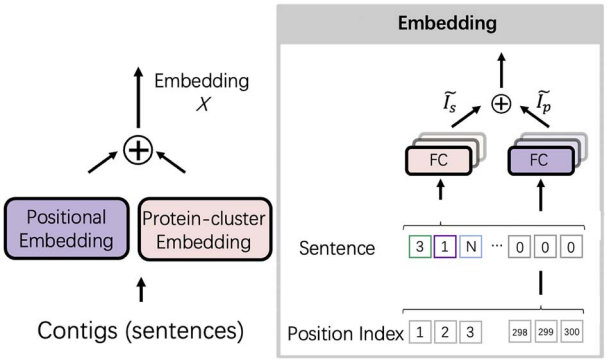

Figure 2. The embedding layer in PhaMer. There are two embedding layers in the model: protein-cluster embedding and positional embedding. The sum of these two embedding layers forms the input to the Transformer block.

The embedding layer As shown in Figure 2, before feeding the Transformer block, we embed the sentence and the position of the tokens via two embedding layers: protein-cluster embedding and positional embedding. The protein-cluster embedding layer resembles a look-up table and returns a numerical vector representing an input token. There are several ways to implement the protein-cluster embedding layer, such as the one-hot encoding used in DeepVirFinder [19] and Seeker [20]. However, because of the size of the vocabulary (45 577), using one-hot encoding can lead to very sparse vectors, which can make the model suffer from the curse of dimensionality [36]. Thus, we use a fully connected layer (FC layer) to conduct linear projection for computing a low-dimensional embedding vector for each token. Because the tokens in the sentences are IDs, the mapping from the ID to a vector by the FC layer functions as a learnable dictionary (look-up table). Given an ID of a token, it will return a corresponding embedding vector of the token.

Because Transformer contains no recurrence or convolution, it uses the positional embedding to encode the position information. The input to the positional embedding is the position index vector. The implementation is the same as the protein-cluster embedding layer with an individual learnable look-up table. The output of the embedding layer has the same dimension as the protein-cluster embedding so that the two embedded vectors can be summed.

$$\begin{cases} \widetilde{I_s} = FC(I_s, W_{I_s}) \\ \widetilde{I_p} = FC(I_p, W_{I_p}) \\ X = \widetilde{I_s} + \widetilde{I_p} \end{cases} \tag{1}$$

Mathematically, the embedding layers can be presented by Eqn. 1. I_s is the input sentence and I_p is the position index vector for the input tokens as shown

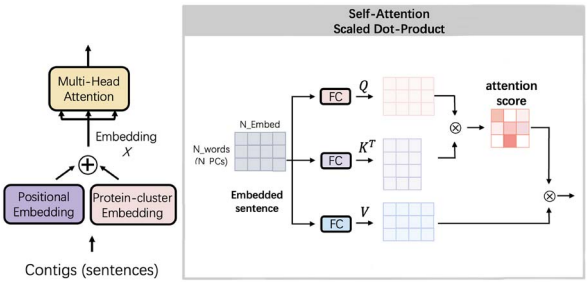

Figure 3. The self-attention mechanism in the Transformer model. The input of the self-attention is the embedded vector and the output is the weighted features with protein–protein relationships information.

in Figure 2. $W_{I_S} \in \mathbb{R}^{N\times embed}$ and $W_{I_P} \in \mathbb{R}^{len\times embed}$ are the learnable parameters of the look-up table for protein-cluster embedding and positional embedding, respectively. N is the number of protein clusters, which is 45 577 in our model, and *len* is the maximum length of the sentence, which is 300 by default. *embed* is a hyperparamter of the embedding dimension and it is set to 512 by default following the guideline in [31]. The padding tokens, which are fixed to zero, are not involved in the downstream computation. The outputs of the embedding are two matrices $\widetilde{I_S} \in \mathbb{R}^{300\times512}$ and $\widetilde{I_P} \in \mathbb{R}^{300\times512}$, where each row represents an embedded token and position vector, respectively. The final output $X \in \mathbb{R}^{300\times512}$ of the embedding layer is the sum of two matrices $\widetilde{I_S}$ and $\widetilde{I_P}$ Then, X will be fed into the Transformer block. Ideally, these embedding layers will capture some of the semantics of the input by placing semantically similar tokens close together in the embedding space [37]. Because the value of the embedding layers in our work represents the proteins of the phages and position of proteins, the embedding could help place proteins with similar functions, such as the proteins for constructing the capsid, in proximity in the embedding space.

Self-attention After embedding the sentences, each token is converted into a vector of size 512 and the embedded sentences will be a $\mathbb{R}^{300\times512}$ matrix. Then, we feed the matrix into the self-attention mechanism as shown in Figure 3. First, three FC layers are adopted to generate a query matrix Q, a key matrix K and a value matrix V. We want to train a model to learn: given a set of proteins (query), which proteins (key) are usually co-present in phage genomes (value). Then, when making a prediction for the contigs, the model will evaluate whether the co-occurrence of some proteins shows enough evidence for phage classification.

$$Attention(Q, K, V) = SoftMax(\frac{QK^T}{\sqrt{d_k}})V \quad (2)$$

Figure 3 and Eqn. 2 show how the self-attention mechanism works. First, the embedded matrix X is projected by three FC layers into Q, K and V, respectively. Second, we multiply Q and the transpose of K and obtain an attention score matrix of size *len* by *len*, where *len* is the length of the sentence. Thus, the value in the attention score matrix represents the strength of associations between two proteins. Then, the SoftMax function is employed to obtain normalized weights for each protein-cluster token and finally we multiply the weight with the value matrix V to score the protein clusters in the sentences. Because the matrix multiplication between Q and the transpose of K might grow large in magnitude when the dimension of the embedding increases, leading to extremely small gradients of the SoftMax function, we divided it with a scaling factor $\sqrt{d_k}$ to prevent gradient vanishing. Following the suggestion of [31], we set $d_k = embed = 512$.

Because the attention matrix only contains pairwise protein cluster information, to model different combinations of pairwise relationships, we use h FC layer groups for linear projections. Each group is called a head ($head_i$), and on each of these projected versions of queries Q_i, keys K_i and values V_i, we can perform the self-attention mechanism in parallel. To reduce computational complexity, in each FC layer we will reduce the dimension for the projected features. The dimension of the output will be $len \times d_s$, where d_s is calculated by $embed/h$. In this work, we choose $h = 8$ by default. Thus, the formula of each head attention can be written as in Eqn. 3.

$$\begin{cases} head_i & = Attention(Q_i, K_i, V_i) \\ Q_i & = FC(X, W_i^Q) \\ K_i & = FC(X, W_i^K) \\ V_i & = FC(X, W_i^V) \end{cases} \quad (3)$$

The parameters in the FC layers are projections matrices: $W_i^Q \in \mathbb{R}^{N\times d_s}$, $W_i^K \in \mathbb{R}^{N\times d_s}$ and $W_i^V \in \mathbb{R}^{N\times d_s}$. Finally, we will concatenate the output from each head and form the final output of the multi-head attention block as shown in Eqn. 4, where $W^O \in \mathbb{R}^{hd_s\times embed}$.

$$MultiHead(Q, K, V) = FC(Concat(head_1, ..., head_h), W^O) \quad (4)$$

While convolution and recurrence in CNN and RNN can record the relative positions directly, the attention mechanism is more suitable for biological data, especially for protein-cluster tokens, because the attention score can learn the remote interactions between proteins from the embedded feature. CNN and RNN can be limited by their architectures that only give them access to local context with a limited window size or receptive field. However, the self-attention mechanism of the Transformer grants access to all positions in the embedded sentence. In addition, the positional embedding enables the model to leverage the position of each protein for prediction. Then all the information can be used in the attention mechanism simultaneously.

Feed-forward networks The output of the multi-head attention block is fed to a two-layer neural network, which is called feed-forward network, as shown in Figure 1 B. We employ a residual connection [38] to the output of the multi-head, followed by layer normalization

[39] to prevent overfitting. Then, we employ the *sigmoid* function to the final output of the Transformer block to compute the probability of a contig being part of a phage.

Model training During training, we first generate protein clusters and vocabulary using the phage genomes released before December 2018 (i.e. our training phages). Then, we convert phage sequences into sentences using the sentence construction method. We also apply data augmentation by randomly generating short segments, ranging from 3 to 15 kbp, to enlarge the training set. These segments are used to improve the robustness to the short contigs, which might not contain many protein clusters. We use both the segments and the complete genomes to train PhaMer to prevent the model from overfitting to the complete genomes. The training data also include the host bacterial genomes of the training phage sequences. Because phages usually share local similarities with their hosts, we use the host genomes as the negative set to create harder negative samples for model training. Compared with using randomly selected bacterial sequences, using the hosts as the negative samples can help the model learn a more accurate classification surface and thus improves the model's generality. We download the host genomes from RefSeq database and both chromosomes and plasmids are included. The bacterial genomes go through the same sentence encoding process as the positive samples. Because some phages do not have known hosts, we sample segments from bacterial genomes to balance the training the positive and negative training data. We employ binary cross-entropy (BCE) loss and Adam optimizer with a learning rate of 0.001 to update the parameters. The model is trained on HPC with the GTX 3080 GPU unit to reduce the running time. Finally, the pre-trained model will be used to identify phages in input sequences.

Experimental setup

Metrics We use the same metrics as the previous works to ensure consistency and a fair comparison: precision, recall and F1-score. Their formulas are listed below (Eqn.5, Eqn. 6, and Eqn. 7):

$$precision = \frac{TP}{TP + FP} \tag{5}$$

$$recall = \frac{TP}{TP + FN} \tag{6}$$

$$F1{-}score = \frac{2 * precision * recall}{precision + recall} \tag{7}$$

TP, *FN* and *FP* represent the number of corrected identified phages, missed phages and falsely identified phages by PhaMer, respectively. If the input contigs do not contain any protein-based token, we will directly assign 'non-phage' label to them, which become part of the *FN*.

Dataset We rigorously tested PhaMer on multiple datasets with increasing complexity. The detailed information is listed in Table 1.

Results

In this section, we will show our experimental results on different datasets and compare PhaMer against the state-of-the-art tools. Because we want to include both alignment-based and learning-based methods in the experiments, we choose the tools that have top performance in each category based on a recent review [15]. Thus, we compared one alignment-based method: VirSorter [16] and four learning-based methods: Seeker [20], DeepVirfinder [19], VirFinder [18] and PPR-meta [21]. Because Virtifier [22] is designed for short reads (length <500bp), we exclude Virtifier from the comparison.

Experiments using the RefSeq dataset

Ten-fold cross-validation We trained our model using 10-fold cross-validation. First, we split our training set into 10 subsets. Second, we use nine subsets for training and the remaining one for validation. Finally, we repeated step two by iteratively choosing one subset for testing and recording the performance. The final results are shown in Figure 4. We also show how the positional embedding and attention mechanism affect the learning performance. *Without positional embedding* means that we only use the sentence as input (shown in Figure 2). *Without attention mechanism* means that we directly feed the embedding feature X into the feed-forward network (shown in Figure 3). The results clearly show that both strategies improve the performance. We also visualize the attention score matrix ($QK^T \in \mathbb{R}^{len \times len}$ in Figure 3) to show the self-attention mechanism can learn important protein associations with biological significance. Detailed information can be found in section *Visualization of the attention score* in the [Supplementary file 1]. After we conducted 10-fold cross-validation, we fixed the parameters of PhaMer using the model with the best performance on the validation set. The following experiments were conducted using this model, whose parameters are also available at our GitHub repository.

Performance on the test set and short contigs We run all the state-of-the-art methods using the pre-trained model with the default parameters on the test sequences. For complete input genomes as inputs, we draw an ROC curve using the output score of each tool in Figure 5. The area under the ROC curve reveals that PhaMer has more reliable results on the complete phages. Then, under the same score cutoff value (0.5) as all other tested tools, we recorded their precision and recall in Figure 6. Meanwhile, we tested the tools on the *short contig test set* described in Section *Experimental setup*. To reduce the bias of data generation, we repeated the short contig generation process for three times and reported the average performance in Figure 6. The comparison reveals that PhaMer can achieve the best performance across

Table 1. The detailed information of the datasets

Name	Description
RefSeq dataset	We split the training and test set by time. All the phage genomes released before December 2018 in RefSeq comprise the training set, while the genomes released after that comprise the test set. This dataset is a widely used benchmark dataset in phage identification task. For each phage, the host information is available based on the keywords '*isolate_host* =' or '*host* =' within each GenBank file. If no known host is available, we use this phage as a positive sample without a negative pair. Finally, 305 bacteria and 4410 phages were downloaded. The training set contains 106 bacteria and 2126 phages. The test set contains 194 bacteria and 2284 phages.
Short contig test set	We randomly cut the test phage genomes into segments of different lengths: 1, 2, 3, 5, 10 and 15 kbp. To balance the dataset, we randomly extract 10 segments from each test phage genome and 100 segments from each test bacterial genome. Finally, we have 22 840 phage segments and 19 400 bacterial segments for each given length. Then we use these segments to evaluate the performance of phage identification on short contigs.
Simulated metagenomic dataset	We use a sophisticated metagenomic simulator, CAMISIM [40], to generate simulated data using six common bacteria living in the human gut. Instead of adding random phages to this dataset, we add simulated reads from phages that infect these bacteria to create a harder case for distinguishing phages from bacteria with shared local similarities. Then, metaSPAdes [41] is applied to assemble the reads into contigs, which are fed into test phage detection tools. Finally, MetaQUAST [42] is used to map contigs to reference phage genomes in order to assign the labels to the contigs. The experimental results can be found in section *Experiments on the simulated data* in the [Supplementary file 1].
Mock metagenomic dataset	Nine shotgun metagenomic sequencing replicates of a mock community [43] are retrieved from the European Nucleotide Archive (BioProject PRJEB19901). We use metaSPAdes to assemble the reads into contigs, which are used for evaluation. Similarly, The label of the contigs is determined using MetaQUAST.
IMG/VR dataset	IMG/VR v3 database [12] contains 2314 129 viral contigs assembled from different environmental samples. We recruit 354 501 contigs with known bacterial hosts. With this dataset, we will compare the recall of different tools for identifying phages from different environments.

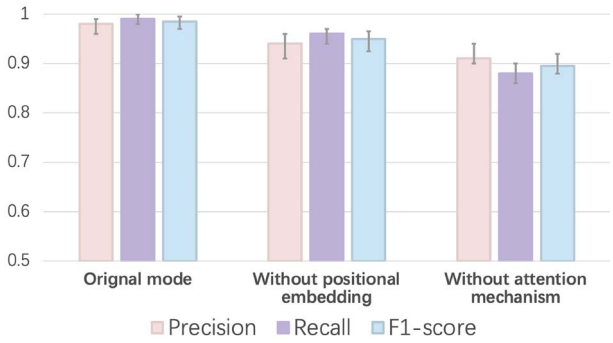

Figure 4. The 10-fold cross-validation performance on the training set. X-axis: training with different methods. Y-axis: the value of each metric (precision, recall and F1-score).

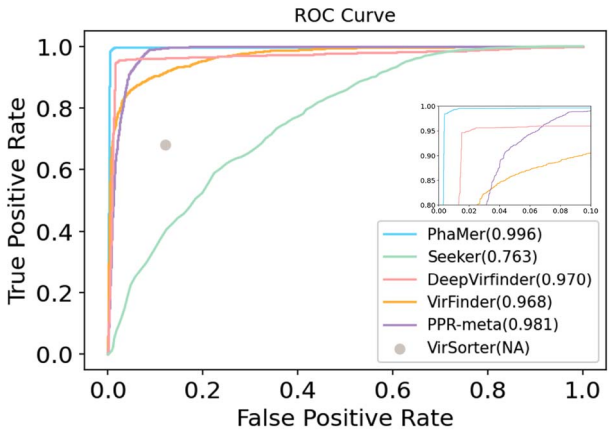

Figure 5. ROC curves on the complete test genomes. The value in the parentheses represents AUC for each tool. Because VirSorter does not provide a score associated with each prediction, we only have one data point. All later experimental results will be reported using the default cutoffs of each tool.

all length ranges. With the increase of contig length, the performance of all pipelines increases. This is expected because longer sequences may contain more information for phage identification. As Figure 6 shows, when the contigs are as short as 1kbp, PhaMer has precision around 0.8, while others have precision lower than 0.8. Thus, we do not suggest that users conduct phage identification for even shorter contigs, which can lead to unreliable results.

Because DeepVirFinder and Seeker support training a customized model, we also tried to retrain these methods with our training set. However, the recall of DeepVirFinder and Seeker dropped to 0.47 and 0.65 on the complete genomes, respectively. This indicates the possibility that their training set might contain some of our test genomes. To keep the better results, we only reported the predictions by the pre-trained models in all the experiments.

The similarity between the training and test set The performance of phage identification can be affected by the similarity between the training and test sequences. We used Dashing [44] to estimate the similarity between the training and test set. First, we ran all-against-all comparisons between sequences in the training and test set. Then, we recorded the largest similarity for each test phage in the test set. The mean value of the similarity is 0.41, indicating a relatively low similarity between the test and training set. To show how the similarity affects the phage identification, we divided the test set according to the dashing similarity. Because low similarity between training and test phages mainly affects the recall of phage

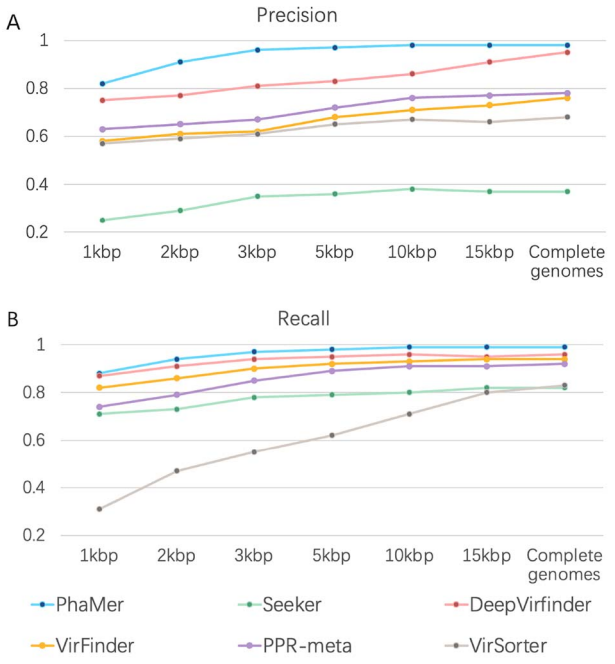

Figure 6. Precision and recall on the genomes and the simulated contigs from the test phages and bacteria. X-axis: contigs with different lengths. Y-axis: the precision on the test set **(A)** and the recall on the test set **(B)**. For simulated contigs, there are 22 840 phage contigs and 19 400 bacterial contigs for each length range. The reported performance is averaged on three such sets of contigs for each length range. The precision and recall of all tools correspond to their default score cutoffs (0.5).

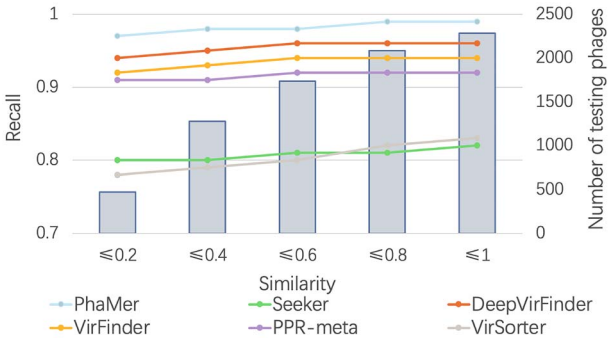

Figure 7. The impact of training-vs-test phage similarity on the recall of PhaMer. X-axis: the dashing similarity between test and training phages. The line plot: recall (left Y-axis). The bar plot: number of test phages (right Y-axis).

identification, we recorded the recall in Figure 8. X-axis stands for the maximum similarity between genomes in the training and test sets. For example, X-axis value 0.2 indicates that all the test phages have similarity ≤ 0.2 against the training phages. Figure 8 shows that with the increase of the similarity, the recall of all methods increases and PhaMer outperforms other tools on a wide range of similarities.

Experiments on the mock metagenomic data

After validating PhaMer on the RefSeq database and the simulated datasets, we compare all the methods on real shotgun-sequenced metagenomic datasets that are used for testing phage identification by the review [15]. The sequencing data are from a mock community [43], which contains 32 species or strains, including five phages. There are nine sequenced datasets with different properties of cell number abundance and protein biomass level from this mock community. These datasets are publicly available at European Nucleotide Archive (BioProject PRJEB19901). Following the guidelines of [15], we used the FASTQC (https://www.bioinformatics.babraham.ac.uk/projects/fastqc) to control the quality of the data and removed overrepresented reads with Cutadapt [46]. The cleaned paired-end reads were fed into metaSPAdes [41] and the output contigs were labeled by MetaQUAST [42]. Only the contigs with length >3kbp will be used for comparison and the prediction results of all methods are shown in Figure 7.

In general, the F1-scores of other tools were considerably lower on this dataset than on the RefSeq benchmark dataset, with an average F1-scores drop by ∼30%. A closer look shows that they commonly misclassified the bacterial contigs as phages in this metagenomic data. The precision of PhaMer is still much better than the state-of-the-art methods. Because PhaMer learns not only the importance of proteins but also the associations between proteins from phage sequences, it is able to make a fine distinction between bacteria and phages. In addition, we use the hard cases (the host genomes) for training, enabling the model to generalize to real metagenomic data.

Experiments on the IMG/VR data

Recently, IMG/VR published the largest public virus genome database IMG/VR v3 [12]. The viruses in this database are quality checked, taxonomically classified and annotated. This dataset provides a good test set to evaluate the recall of phage identification tools. Following the previous work [20], we downloaded 2314 129 viral contigs assembled from different environmental samples and recruited 354 699 viral contigs with known bacterial hosts. All contigs shorter than 3kbp were removed, resulting in 354 501 phage contigs. Such a broad coverage of phages from different environmental niches allow us to test the reliability of the phage identification tools. Because only phages were tested in the experiments, we reported the recall of different tools.

Figure 9 shows the recall of all six methods on the IMG/VR dataset. We split the dataset according to the living environments of the phages and show the performance of all tools on five of them with largest data samples (Figure 9 A): *human*, *plant*, *marine*, *freshwater* and *other*. The results reveal that PhaMer outperforms all other methods on these five domains. In particular, PhaMer significantly improved the recall of phage identification in plant-associated samples. Figure 9 B shows the summarized recall on all the datasets with PhaMer having a recall of 5% higher than the second best tool PPR-meta.

Running time comparison

The most resource-demanding components in PhaMer are the translation (Prodigal) and sequence alignment

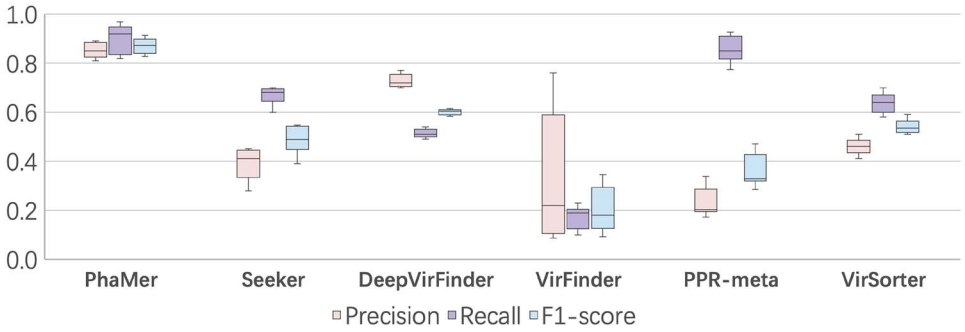

Figure 8. Results on the mock metagenomic data. X-axis: the names of the compared methods. Y-axis: the score of the metrics. Results on the mock metagenomic data. PhaMer outperforms the other tools regarding precision, recall and F1-score. PPR-meta achieves the second-best recall but shows low precision.

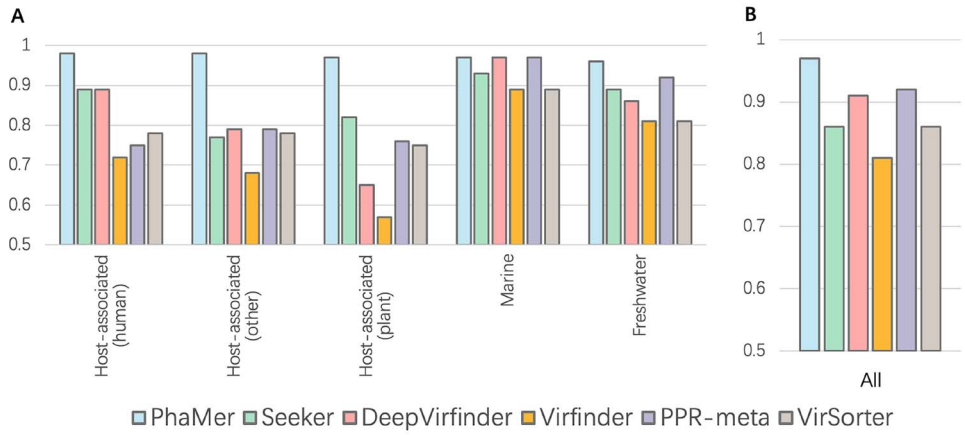

Figure 9. Results on the IMG/VR v3 database. Y-axis: the recall of PhaMer and the state-of-the-art tools for identifying phages in the IMG/VR database. **(A)**: Recall of different tools on phages living in different environments; **(B)**: the overall recall on the whole database.

Table 2. The average elapsed time to make predictions for the RefSeq test genomes. All the methods are run on Intel® Xeon® Gold 6258R CPU with eight cores.

Program	PhaMer	VirFinder	DeepVirFinder	Seeker	PPR-meta	VirSorter
Elapsed time(min)	67	31	23	58	46	214

(DIAMOND BLASTP). We used these two steps to convert input sequences into protein-based sentences. Both the protein cluster vocabulary and position information are utilized as input features of the Transformer model. Table 2 shows the average elapsed time of classifying the test set (2284 genomes) for each tool. PhaMer is not the fastest program, and ∼90% running time is used to run Prodigal and DIAMOND BLASTP.

Discussion

As shown in the experiments, existing approaches, such as Seeker, VirFinder, DeepVirFinder, PPR-meta and VirSorter, failed to achieve high precision or recall in phage identification, especially on metagenomic data. In this work, we demonstrate that PhaMer can render better performance for novel phage identification. The major improvement of our method stems from the adoption of the language model Transformer for bidirectional contextual protein embedding and our careful construction of negative samples. By constructing protein-based vocabulary, we can incorporate the similarity between phages. Using the positional embedding and the self-attention mechanism in Transformer, we can learn the importance of proteins and also their associations. In addition, we carefully construct our negative training samples using phages' host bacterial genomes. Because of the local similarities between phages and their hosts, this negative training set is more difficult to classify and tend to have direct bearing on learning the optimal decision surface. The benchmark experimental results on complete genomes, short contigs, simple and complicated metagenomic data show that our model outperforms others in different scenarios. Importantly, its performance is more robust than others on short contigs. And it increases the F1-score by 27% on the mock metagenomic data.

Considering that we first constructed protein clusters from phage proteins, it is a fair question to ask

whether we can achieve an optimal classification by controlling the protein matching criteria (E-value). When constructing our negative set (bacterial host sequences), we found that over 80% of bacterial genomes have multiple alignments with the protein clusters in our vocabulary. Thus, there is a trade-off between the recall and precision in terms of the E-value cutoff. Because phages are highly diverse and some protein clusters can have remote homologs in new phages, using a very stringent E-value cutoff can miss new pages. We found that if we use a very strict E-value cutoff (e.g. E-value close to 0), most of these bacterial genomes can be rejected. However, the recall of identifying phages will drop to 0.3. Thus, it is hard to strike an optimal balance between sensitivity and precision by adjusting E-value cutoffs.

Although PhaMer has greatly improved phage contig identification, we have several aims to optimize PhaMer in our future work. One possible extension is to incorporate prophage detection in PhaMer. There are a number of prophage annotation tools, such as Prophage Hunter [47] and Phage_Finder [48]. In our software, we supply a metric named *proportion* to record the ratio of matched tokens in our vocabulary to the predicted proteins by Prodigal for each input contig. Intuitively, for each input that is predicted as 'phage' by PhaMer, we can further check the value of *proportion*. If this value is very small, the input contig is likely to be a prophage rather than a phage. Users can filter the contigs according to their needs conveniently. In the future, we will incorporate prophage detection tools in our pipeline.

Like a majority of phage identification tools, PhaMer is a binary classification model with the goal of distinguishing phages from bacterial contigs, which can originate from either chromosomes or plasmids. In our current design and test, we have plasmid sequences in both the training and test sequences. For example, some contigs assembled from the mock metagenomic data are from plasmids. Because the plasmid-originated sequences are treated as 'negative' samples as the bacterial chromosomes, PhaMer does not distinguish them from chromosomes. There are other tools available to distinguish plasmids from the host genomes [49–51]. Users can run those tools for downstream analysis.

Key Points

- Phage identification is a key step for novel phage discovery in metagenomic data.
- We developed a phage identification tool named PhaMer that employs the Transformer and protein-based tokens to learn both the protein composition and their associations in phage sequences.
- Our rigorous test of PhaMer on both simulated and real sequencing data and the benchmark experiments against five recently published tools show that PhaMer is the most accurate phage identification tool.

Data Availability

All data and codes used for this study are available online or upon request to the authors. The source code of PhaMer is available via: https://github.com/KennthShang/PhaMer.

Funding

This work was supported by City University of Hong Kong (Project 9678241 and 7005453) and the Hong Kong Innovation and Technology Commission (InnoHK Project CIMDA).

References

1. McGrath S, van D. *Bacteriophage: genetics and molecular biology*. Wymondham: Caister Academic Press, 2007.
2. Zhong Z-P, Tian F, Roux S, *et al.* Glacier ice archives nearly 15,000-year-old microbes and phages. *Microbiome* 2021;**9**(1):1–23.
3. Nishimura Y, Watai H, Honda T, *et al.* Environmental viral genomes shed new light on virus-host interactions in the ocean. *Msphere* 2017;**2**(2):e00359–16.
4. Gregory AC, Zayed AA, Conceição-Neto N, *et al.* Marine DNA viral macro-and microdiversity from pole to pole. *Cell* 2019;**177**(5):1109–23.
5. Azimi T, Mosadegh M, Nasiri MJ, *et al.* Phage therapy as a renewed therapeutic approach to mycobacterial infections: a comprehensive review. *Infection and Drug Resistance* 2019;**12**:2943.
6. Loc-Carrillo C, Abedon ST. Pros and cons of phage therapy. *Bacteriophage* 2011;**1**(2):111–4.
7. Lee S-E, Lee D-Y, Lee W-G, *et al.* Osong Public Health and Research Perspectives. *Osong Public Health and Research Perspectives* 2020;**11**(3):118–27.
8. Moon K, Kang I, Kim S, *et al.* Genomic and ecological study of two distinctive freshwater bacteriophages infecting a Comamonadaceae bacterium. *Sci Rep* 2018;**8**(1):1–9.
9. Moon K, Jeon JH, Kang I, *et al.* Freshwater viral metagenome reveals novel and functional phage-borne antibiotic resistance genes. *Microbiome* 2020;**8**:1–15.
10. Moon K, Kim S, Kang I, *et al.* Viral metagenomes of Lake Soyang, the largest freshwater lake in South Korea. *Scientific Data* 2020;**7**(1):1–6.
11. Santiago-Rodriguez TM, Hollister EB. Human virome and disease: high-throughput sequencing for virus discovery, identification of phage-bacteria dysbiosis and development of therapeutic approaches with emphasis on the human gut. *Viruses* 2019;**11**(7):656.
12. Roux S, Páez-Espino D, Chen I-MA, *et al.* IMG/VR v3: an integrated ecological and evolutionary framework for interrogating genomes of uncultivated viruses. *Nucleic Acids Res* 2020;**49**(D1):D764–75.
13. Edwards RA, McNair K, Faust K, *et al.* Computational approaches to predict bacteriophage–host relationships. *FEMS Microbiol Rev* 2016;**40**(2):258–72.
14. Congyu L, Zhang Z, Cai Z, *et al.* Prokaryotic virus host predictor: a Gaussian model for host prediction of prokaryotic viruses in metagenomics. *BMC Biol* 2021;**19**(1):1–11.
15. Ho SFS, Millard AD, van Schaik W. Comprehensive benchmarking of tools to identify phages in metagenomic shotgun sequencing data. *bioRxiv* 2021;**1**:1–30.

16. Roux S, Enault F, Hurwitz BL, *et al.* VirSorter: mining viral signal from microbial genomic data. *PeerJ* 2015;**3**:e985.
17. Jurtz VI, Villarroel J, Lund O, *et al.* MetaPhinder-identifying bacteriophage sequences in metagenomic data sets. *PLoS One* 2016;**11**(9):e0163111.
18. Ren J, Ahlgren NA, Yang Young L, *et al.* VirFinder: a novel k-mer based tool for identifying viral sequences from assembled metagenomic data. *Microbiome* 2017;**5**(1):1–20.
19. Ren J, Song K, Deng C, *et al.* Identifying viruses from metagenomic data using deep learning. *Quantitative Biology* 2020;**8**:1–14.
20. Auslander N, Gussow AB, Benler S, *et al.* Seeker: alignment-free identification of bacteriophage genomes by deep learning. *Nucleic Acids Res* 2020;**48**(21):e121–1.
21. Fang Z, Tan J, Wu S, *et al.* PPR-Meta: a tool for identifying phages and plasmids from metagenomic fragments using deep learning. *GigaScience* 2019;**8**(6):giz066.
22. Yan Miao F, Liu TH, Liu Y. Virtifier: A deep learning-based identifier for viral sequences from metagenomes. *Bioinformatics* 2021;**38**:1216–22.
23. Guo J, Bolduc B, Zayed AA, *et al.* VirSorter2: a multi-classifier, expert-guided approach to detect diverse DNA and RNA viruses. *Microbiome* 2021;**9**(1):1–13.
24. Bolduc B, Jang HB, Doulcier G, *et al.* vConTACT: an iVirus tool to classify double-stranded DNA viruses that infect Archaea and Bacteria. *PeerJ* 2017;**5**:e3243.
25. Shang J, Sun Y. Predicting the hosts of prokaryotic viruses using GCN-based semi-supervised learning. *BMC Biol* 2021;**19**(1):1–15.
26. Buchfink B, Xie C, Huson DH. Fast and sensitive protein alignment using DIAMOND. *Nat Methods* 2015;**12**(1):59–60.
27. Nambiar A, Heflin M, Liu S, *et al.* (eds). Transforming the language of life: Transformer neural networks for protein prediction tasks. In: *Proceedings of the 11th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*. New York NY United States: Association for Computing Machinery, 2020, 1–8.
28. Wei D, Zhao X, Sun Y, *et al.* SecProCT: In Silico Prediction of Human Secretory Proteins Based on Capsule Network and Transformer. *Int J Mol Sci* 2021;**22**(16):9054.
29. Ma Y, Guo Z, Xia B, *et al.* Identification of antimicrobial peptides from the human gut microbiome using deep learning. *Nat Biotechnol* 2022;**40**:921–31.
30. Chaban Y, Lurz R, Brasilès S, *et al.* (eds). Structural rearrangements in the phage head-to-tail interface during assembly and infection. *Proc Natl Acad Sci* 2015;**112**(22):7009–14.
31. Vaswani A, Shazeer N, Parmar N, *et al.* Attention is all you need. In: Guyon I, Von Luxburg U, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R (eds). *Advances in Neural Information Processing Systems*. Long Beach, California, USA: Curran Associates Inc., 2017, 5998–6008.
32. Devlin J, Chang M-W, Lee K, *et al.* Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint* 2018;**1**:1–16.
33. Kitaev N, Kaiser Ł, Levskaya A. Reformer: The efficient transformer. In: *8th International Conference on Learning Representations(ICLR)*. Engineering and Technology organization, 2020.
34. González-Tortuero E, Krishnamurthi R, Allison HE, *et al.* Comparative analysis of gene prediction tools for viral genome annotation. *bioRxiv* 2021;**1**:1–17.
35. Enright AJ, Van Dongen S, Ouzounis CA. An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Res* 2002;**30**(7):1575–84.
36. Mikolov T, Sutskever I, Chen K, *et al.* Distributed representations of words and phrases and their compositionality. In: Burges CJ, Bottou L, Welling M, Ghahramani Z, Weinberger KQ (eds). *Advances in neural information processing systems*. Lake Tahoe, Nevada, USA: Curran Associates, Inc., 2013, 3111–9.
37. Cui P, Wang X, Pei J, *et al.* A survey on network embedding. *IEEE Transactions on Knowledge and Data Engineering* 2018;**31**(5):833–52.
38. He K, Zhang X, Ren S, *et al.* Deep residual learning for image recognition. In: Bajcsy R, Li F-F, Tuytelaars T (eds). *Proceedings of the IEEE conference on computer vision and pattern recognition*. Las Vegas, Nevada, USA: IEEE, 2016, 770–8.
39. Ba JL, Kiros JR, Hinton GE. Layer normalization. *arXiv preprint* 2016;**1**:1–14.
40. Fritz A, Hofmann P, Majda S, *et al.* CAMISIM: simulating metagenomes and microbial communities. *Microbiome* 2019;**7**(1):1–12.
41. Nurk S, Meleshko D, Korobeynikov A, *et al.* metaSPAdes: a new versatile metagenomic assembler. *Genome Res* 2017;**27**(5):824–34.
42. Mikheenko A, Saveliev V, Gurevich A. MetaQUAST: evaluation of metagenome assemblies. *Bioinformatics* 2016;**32**(7):1088–90.
43. Kleiner M, Thorson E, Sharp CE, *et al.* Assessing species biomass contributions in microbial communities via metaproteomics. *Nat Commun* 2017;**8**(1):1–14.
44. Baker DN, Langmead B. Dashing: fast and accurate genomic distances with HyperLogLog. *Genome Biol* 2019;**20**(1):1–12.
45. Andrews S, *et al.* FastQC: a quality control tool for high throughput sequence data. 2017;**2010**.
46. Martin M. Cutadapt removes adapter sequences from high-throughput sequencing reads. *EMBnet journal* 2011;**17**(1):10–2.
47. Song W, Sun H-X, Zhang C, *et al.* Prophage Hunter: an integrative hunting tool for active prophages. *Nucleic Acids Res* 2019;**47**(W1):W74–80.
48. Fouts DE. Phage_Finder: automated identification and classification of prophage regions in complete bacterial genome sequences. *Nucleic Acids Res* 2006;**34**(20):5839–51.
49. Krawczyk PS, Lipinski L, Dziembowski A. PlasFlow: predicting plasmid sequences in metagenomic data using genome signatures. *Nucleic Acids Res* 2018;**46**(6):e35–5.
50. Antipov D, Raiko M, Lapidus A, *et al.* Plasmid detection and assembly in genomic and metagenomic data sets. *Genome Res* 2019;**29**(6):961–8.
51. Andreopoulos WB, Geller AM, Lucke M, *et al.* Deeplasmid: Deep learning accurately separates plasmids from bacterial chromosomes. *Nucleic Acids Res* 2022;**50**(3):e17–7.