2025 International Conference on Intelligent Computing
July 26-29, Ningbo, China
https://www.ic-icc.cn/2025/index.php

MetaCleaner: A Deep Neural Network for Phage Recognition with Denoising

Yingqi Liu[1], Yong Wang[2] and Ying Wang[3]

[1] Guangdong University of Technology, Guangzhou, China
[2] Guangdong University of Technology, Guangzhou, China
wangyong@gdut.edu.cn
[3] Guangdong University of Technology, Guangzhou, China

Abstract. At present, the phage genome sequence recognition model based on deep learning technology faces two problems, namely, the pollution of human genome sequence and noise interference. To address this problem, we propose MetaCleaner, a phage genome sequence recognition model. MetaCleaner uses the k-mer count as the classification basis for genome sequences, and uses a parallel convolution filter and average pooling method to extract the k-mer count features of genome sequences. The denoising module implemented by the transformer architecture is used to predict the difference between the k-mer count feature of the noisy sequence and the k-mer count feature of the noise-free sequence, and the denoising operation is completed by subtracting the difference between the k-mer count feature of the noisy sequence. Finally, the denoised k-mer count feature is input into the fully connected layer to obtain the probability of the sequence belonging to phage and human. Our experiments on test sets with noise show that MetaCleaner is robust to noise, and experiments on real metagenomic datasets show that MetaCleaner outperforms recent proposed phage recognition models.

Keywords: Deep learning, Metagenomics, Denoising, phage identification, Noise.

1 Introduction

Bacteriophages are viruses that infect bacteria and archaea, and are considered to be the most abundant and diverse biological entities on Earth. Bacteriophages play a key role in constructing the human gut microbiota [1]. In the past few decades, phages were discovered by culture-dependent genome sequencing methods, which are time-consuming and expensive. Metagenomic sequencing technology is a new approach to genome sequencing that is able to sequence phages directly from gut samples.

At present, there are three types of phage identification methods: sequence alignment based methods, machine learning based methods and deep learning based methods. For example, VirSorter [2] builds a viral reference database and compares sequences with more than two tested genes to the database to determine viral assemblies (contigs).

MetaPhinder [3] constructed a phage whole genome sequence database and used BLAST (Basic Local Alignment Search Tool) to align with the database and identify phage assemblies (contigs). The methods based on contrast have the problems of long execution time and large memory consumption in the process of database construction and sequence mapping.

Machine learning-based methods extract gene features from predefined sequences and train a classifier for phage identification, achieving better performance than the methods based on sequence alignment. For example, VirFinder [4] calculated k-mer frequencies of sequences and trained logistic regression models to identify viral sequences. VIBRANT [5] extracts protein features and uses a hybrid machine learning model for virus identification. VirSorter2[6] is an extension of VirSorter and random forest classifiers trained on gene features to identify viruses. Yet genetic signatures are often designed by human engineers with broad domain expertise, which remains a difficult task to determine.

In recent years, deep learning has achieved great success in many fields, and some deep learning-based methods have been proposed for metagenomic phage DNA identification, which can automatically extract and learn features for classification. For example, DeepVirFinder [7] and CHEER [8] use CNNS to predict viral sequences. Virtifier [9] uses attention-based LSTM for short virus sequence recognition. DETIRE [10] used graph-based nucleotide sequence em- bedding strategy to enrich the expression of DNA sequences by training embedding matrices. Then, the trained Convolutional Neural Network (CNN) and Bidirectional Long Short-Term Memory (Bi-LSTM) network were used to ex- tract spatial features and sequence features, respectively, to enrich the features of short sequences. MetaPhaPred [11] uses convolutional neural network (CNN) to capture sequence feature mapping, uses bidirectional Long Short-Term memory network (Bi-LSTM) to capture the long-term dependence between features from forward and backward, and uses self-attention mechanism to enhance the representation of important features. Although these deep learning models have achieved good prediction performance, they all ignore two problems: the interference of human DNA sequences and the interference of noise.

Background contamination is currently unavoidable [12]. Because the hu- man genome is roughly one thousand times larger than an average bacterial genome ($\sim 3\times10^9$ *versus* $\sim 3\times10^6$ *bp*), host DNA can quickly drown out microbial reads in samples containing even a relatively small number of human cells.[13] The presence of human DNA sequences can cause interference in phage analysis, and the exclusion of human DNA sequences is a necessary step to per- form phage analysis. However, the phage recognition models proposed in recent years do not have the ability to distinguish phage DNA sequences from hu- man DNA sequences. Our experiments on real metagenomic datasets observe that MetaPhaPred, CHEER, DeepVirFinder, Virtifier, and DETIRE achieve no more than 52.42% accuracy in identifying human DNA sequences.

Noise is an important property of metagenomic data, which manifests as insertion, deletion and substitution of bases on DNA sequences. [14] The availability of more affordable sequencing technologies has been accompanied by noisier reads. Oxford Nanopore's MinION[15] comes with error rates close to 10%, orders of magnitude higher than typical noise levels for Illumina, a more expensive technology. Sequence alignment based methods suffer from increasing ambiguity as noise increases These noises can affect downstream applications[16]. We demonstrate experimentally that noise affects the performance of deep learning models.

To address the above two issues, we propose a phage recognition model, MetaCleaner. MetaCleaner contains three modules: feature extraction module, denoising module, and linear classification module. The feature extraction mod- ule uses multiple parallel convolutional filters to extract the feature maps of k-mers with different lengths. The average pooling layer is used to extract k-mer counts of DNA sequences. The denoising module is implemented by a trans- former encoding block, which is used to capture the dependencies between k- mers of different lengths and predict abnormal k-mer counts. The k-mer count minus the k-mer count of the abnormal k-mer achieves the purpose of denoising. The linear classification module is a fully connected layer with two hidden layer units, which is used to produce the probability that the DNA sequence belongs to phage and non-phage.

2 Materials and methods

2.1 Problem Definition

Let $s_i = (a_1, a_2, \ldots a_m)$ denote the ith DNA sequence in the Metagenomic dataset, where $a_j \in \{A, T, C, G\}$, $0 \le j \le m$, m is the length of the sequence. Let y_i be the label of s_i, where $y_i = 0$ denotes that s_i is a non-phage DNA sequence and $y_i = 0$ denotes that the s_i is a phage DNA sequence. MetaCleaner consists of three modules, feature extraction module $f_\theta(z|s)$, classification module $C_\sigma\ (y|z)$ and denoising module $N_\emptyset(d|z)$, where θ , σ and $\emptyset$ represent the trainable parameters of feature extraction module, classification module and denoising module,respectively.The structural diagram of MetaCleaner is shown in **Fig. 1**.

2.2 Model structure

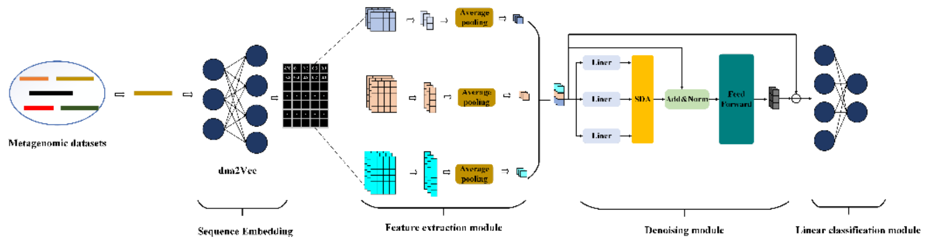

Fig. 1. Model structure. MetaCleaner is divided into three modules: feature extraction module, denoising module, and linear classification module. MetaCleaner first converts the DNA sequence into an embedding matrix with dna2Vec, and then the feature extraction module extracts the k-mer count features of the DNA sequence using a convolutional filter and an average pooling layer. The k-mer count features are input to the denoising module, which denoises the k-mer counts, and then the denoised k-mer counts are input to the linear classification module to obtain the probability that the DNA sequence belongs to phage and non-phage.

Sequence Embedding. Most of the deep learning-based methods mainly used the one-hot encoding technique to represent DNA sequences. The distance be- tween any two one-hot vectors is equal and the correlation information between nucleotides is ignored, which may affect the prediction accuracy[11]. There- fore, we use the word embedding technique commonly utilized in natural language processing to learn the distributed representations of k-mers by applying dna2Vec[18].

K-mer is a biologically meaningful feature of DNA sequences. A k-mer is a length k subsequence of genomic sequences; for any sequence of length L, there exist a maximum of $L - K + 1$ possible k-mers. s_i can be represented as an ordered set of k-mers.

$$s_i^{kmer} = \{w_1, w_2, w_3, ..., w_n\} \tag{1}$$

$$n = m - k + 1 \tag{2}$$

$$w_t = s_i[k(t-1) + q : (kt + q)] \tag{3}$$

where $t = 1,2,3, ..., n$. $s_i[(k(t - 1) + q) : (kt + q)]$ denotes the element in s_i whose index position is from $k(t - 1) + q$ to $kt + q$. We set $k = 3$, s_i is preprocessed into a string of codons with a stride of some bases [9], and then learn the word embedding vector of the 3-mer using dna2Vec. Dna2Vec first encodes the 3-mer into a "One-hot" vector using the "One-hot" encoding technique. "One-hot" is one of the most widely used encodings in the field of natural language processing, "One-hot" represents the vocabulary word w_t using a vector v_{w_t} of size n_v, where n_v is the size of the vocabulary:

$$v_{w_t} = \left[z_1^t, z_2^t, ..., z_{n_v}^t\right]^T \tag{4}$$

For each value z_l^t in v_{w_t}, $0 \le l \le n_v$, $n_v = 64$. z_l^t is encoded as 1 when l is equal to the index of the word in the vocabulary, and as 0 otherwise, i.e

$$z_l^t = \begin{cases} 0, if \;\; index(w_t) != l \\ 1, if \;\; index(w_t) = l \end{cases} \tag{5}$$

$index(w_t)$ is the index of the word in the vocabulary. Next, dna2Vec uses a two-layer fully connected neural network to learn the word embedding vector v'_{w_t} from v_{w_t}. In this step, dna2Vec follows the principle of the Skip-gram algorithm, which maximizes the prediction of the words in the context window based on the previous words. More specifically, let W and W' be the weights of the first and second layers in the two-layer fully connected neural network, s be the size of the context window, and $v_{w_{t+h}}$ is the "One-hot" encoding vector in the context window of w_t, where $-s \le h \le s$. Then the training process of dna2Vec is driven by the following loss function:

$$Loss = \frac{1}{m-2} \sum_{h=1}^{m-2} \sum_{-s \le h \le s, h \ne 1} v_{w_{t+h}} \log\left[\alpha\left(v_{w_t}^T \cdot W \cdot W^T\right)\right] \tag{6}$$

Where α represents the activation function softmax.

After dna2Vec is trained, the second layer of the fully connected neural network is discarded and the first layer of the fully connected neural network is retained, then the conversion to the word embedding vector is calculated as follows:

$$v'_{w_t} = v_{w_t}^T \cdot W \tag{7}$$

s_i will be encoded as s_i^{emb} :

$$s_i^{emb} = \left(v'_{w_1}, v'_{w_2}, ..., v'_{w_{m-2}} \right) \tag{8}$$

Feature extraction module. K-mers can be seen as subsequences extracted sequentially from the first base on the genomic sequence by a sliding window of length a step of 1, in a process similar to how the filters of convolutional neural networks work. When k is sufficiently large, the majority of k-mers are unique to the species carrying them. These species-specific k-mers may serve as signatures, directly implicating the appropriate taxonomic classification [17].So the feature extraction module uses convolution filter to extract the features of k-mer on DNA sequence. Assume that the size of the convolution filter is u ,the convolution filter is computed as follows:

$$x_q^{C_u} = f\left(W^{C_u} * v'_{w_{t:t+u-1}} + b^{C_u} \right) \tag{9}$$

$$v_{w_{t:t+u-1}} = \left\{ v'_{w_1}, v'_{w_2}, v'_{w_3}, ..., v'_{w_{t+u-1}} \right\} \tag{10}$$

$$x^{C_u} = \left[x_1^{C_u}, x_2^{C_u}, x_3^{C_u}, ..., x_{m-u-1}^{C_u} \right] \tag{11}$$

where v'_{w_t} is the encoding vector corresponding to the t th 3-mer in s_i^{emb} and m is the length of s_i . W^{C_u} and b^{C_u} are the trainable weights and bias values of the convolutional filter. $*$ denotes the convolution operation. f denotes the activation function Relu, and $x_q^{C_u}$ denotes the value of the q th feature pattern, where $1 \le q \le m-u-1$.

The feature extraction module sets convolution filters of size 1, 2 and 13, respectively. The convolution filter size is set according to the biological characteristics of the genome. [27] shows that k-mers with length between 15 and 20 are significantly more compact in the sequence space of the human genome than the genomes of other species, and it is proposed that 15-mers can be used as probes to detect non-human gene sequences in samples. So the feature extraction module sets a convolutional filter of size 13 to extract the 15-mer feature map. DeepVirFinder [7] and ViraMiner [22] demonstrated that codon usage preferences of DNA sequences can help distinguish viral sequences from non-viral sequences. So the feature extraction module sets convolution filters of size 1 to capture the codon usage preference of DNA sequences. [20] indicates that tetranucleotide frequencies provide information about the structure of the microbial genome and can help identify the genomic regions of phages, so we set a convolution filter of size 2.

Let convolutional filters of size 1, 2 and 13 produce feature maps x^{C_1} , x^{C_2} and $x^{C_{13}}$,respectively. MeatCleaner uses the average pooling method to downsample these feature maps. [22] demonstrated that the average pooling method provides important information about k-mer counts, and methods based on k-mer counts are effective in separating viral samples from non-viral samples, The average pooling method can produce a distinguishing feature map when there are multiple specific k-mers in the DNA sequence. The average pooling method is calculated as follows:

$$x^{avg_u} = \frac{1}{m-u-1}\sum_{q=1}^{m-u-1} x_q^{C_u} \tag{12}$$

x^{C_1} , x^{C_2} and $x^{C_{13}}$ after average pooling layer of feature mapping for x^{avg_1} , x^{avg_2} and $x^{avg_{13}}$, The output of the final feature extraction module is the concatenation of three average pooling feature maps:

$$x^{avg} = Concat\left(x^{avg_1}, x^{avg_2}, x^{avg_{13}}\right) \tag{13}$$

x^{avg} will be input to the denoising module.

Denoising module. The genome sequencing process is imperfect and can result in reads containing various types of noise, including base substitutions, insertions, and deletions (INDELs). Base substitutions, insertions, and deletions can lead to abnormal k-mer counts in DNA sequences, which can affect the performance of the model. To solve this problem MetaCleaner sets the denoising module.

In the high-throughput sequencing setting the channel input is a single, noise-free sequence and the output are numerous overlapping, short, noisy sequences [28]. Let the noise-free sequence corresponding to s_i be g_i . The purpose of the denoising module is to train a residual map $N_\phi\left(d \mid z\right) \approx v$ and then we have

$$f_\theta\left(z \mid g_i\right) = f\left(z \mid s_i\right) - v \tag{14}$$

where v is the difference between the k-mer count of s_i and the k-mer count of g_i . In brief we use the denoising module to learn the difference between the k-mer counts of noiseless and noisy DNA sequences. This idea is borrowed from [23].

The feature extraction module extracts the k-mer counts of 3-mer, 4-mer and 15-mer. The 4-mer can be represented as an ordered set of 3-mer, the 15-mer can be represented as an ordered set of 3-mer or 4-mer. So there is a dependency between the k-mer counts for 3-mer, 4-mer and 15-mer in the same DNA sequence. Changes in the 3-mer count will be reflected in the 4-mer count and 15-mer count.The same goes for 4-mer and 15-mer.

Therefore, we adopt the structure of transformer encoding block [31] to implement the denoising module. The self-attention mechanism in transformer encoding block is often used to extract the dependencies between different features, examples include [24], [25] and [26]. The output of the feature extraction module x^{avg} is used as the input of the denoising module, and the calculation process of x^{avg} through the self-attention mechanism is as follows:

$$Q = x^{avg}W^Q, K = x^{avg}W^K, V = x^{avg}W^V \tag{15}$$

$$X = SDA(Q,K,V) = \alpha\left(\frac{QK^T}{\sqrt{d_{model}}}\right) \tag{16}$$

α represents the activation function softmax. SDA denotes the Scaled Dot-Product Attention, and W^Q, W^K and W^V are the trainable parameters of the attention mechanism, respectively. The scale factor $\frac{1}{\sqrt{d_{model}}}$ is similar to the standard dot product note. When the value of d_{model} is large, the dot product produces large results, which may lead to extreme values after the $softmax$ function is applied. These values in turn may cause the gradient to vanish. To solve this problem, the value of the dot product is scaled by d_{model} . $head_i$ denotes the i th attention head. X represents the output of SDA. The residual join and layer normalization (LN) are then applied to X :

$$X_{LN} = LN\left(X + x^{avg}\right) \tag{17}$$

X_{LN} is then fed to the feedforward network containing the full perceptron, which consists of two linear transformations with a ReLU activation function in between. Finally we obtain d_i, the difference between the k-mer count of s_i and the k-mer count of g_i :

$$d_i = f\left(X_{LN}W_1 + b_1\right)W_2 + b_2 \tag{18}$$

f denotes the activation function Relu. W_1, W_2, b_1 and b_2 are the trainable parameters of the feedforward network. Then we obtain the k-mer counts of the noise-free sequence as follows:

$$x^{clean} = x^{avg} - d_i \tag{19}$$

Linear classification module. x^{clean} is input to the fully connected layer containing two hidden layer units, and then the $softmax$ activation function generates the probability that s_i belongs to the phage DNA sequence and the non-phage DNA sequence

$$p_i = \alpha\left(x^{clean}W_3 + b_3\right) \tag{20}$$

W_3 and b_3 are the trainable parameters of the Linear classification module.

3 Learning

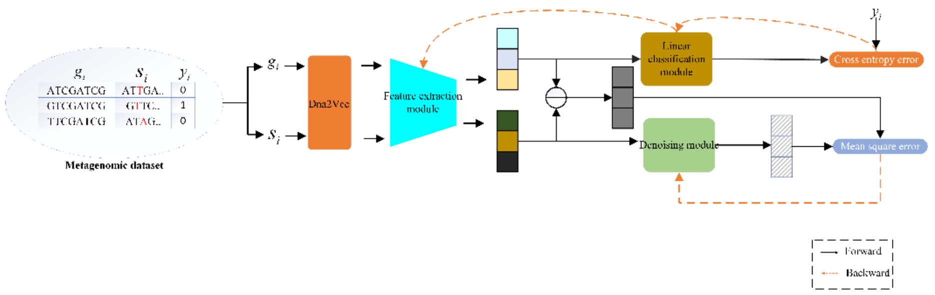

Fig. 2. Training flow. The reference sequence g_i and the noise subsequence s_i of g_i are input into the model together, and the k-mer count feature is extracted by the feature extraction module. The k-mer count feature of g_i is input into the linear classification module to obtain the probability distribution and calculate the cross-entropy error, and then backpropagate to update the parameters of the linear classification module and the feature extraction module. The feature count of s_i is input to the denoising module, and then the denoising module predicts the difference between the k-mer count of g_i and the feature count of s_i, and then calculates the mean square error of the predicted value and the actual difference, and then backpropagates to update the parameters of the denoising module.

The training flow of MetaCleaner is shown in **Fig. 2**, where MetaCleaner takes reference sequence and noise sequence as input during training. The noise sequence is generated from the reference sequence by randomizing subsequences and replacing r % of the bases within these subsequences to effectively simulate substitution noise. Given a training dataset consisting of reference sequence, noise sequence, and classification label, denoted as $\{(g_1,s_1,y_1),(g_2,s_2,y_2),\dots,(g_n,s_n,y_n)\}$. n denotes the total number of samples contained in the training set. The reference sequence and noise sequence are processed through the feature extraction module to obtain the global pooling feature vectors z_i^g and z_i^s.

$$z_i^g = f_\theta(z \mid g_i), z_i^s = f_\theta(z \mid s_i) \tag{21}$$

Then z_i^g and z_i^s are fed into the linear classification module and denoising module to obtain the prediction score p_i and vector d_i, respectively.

$$p_i = C_\sigma\left(y \mid z_i^g\right), d_i = N_\phi\left(d \mid z_i^s\right) \tag{22}$$

The learning process is driven by optimizing two objectives:

$$L_{noise} = \frac{1}{n}\sum_{i=0}^{n}\left\| d_i - \left(z_i^s - z_i^g\right)\right\|^2 \tag{23}$$

$$L_c = \frac{1}{n}\sum_{i=0}^{n} y_i \log\left(p_i\right) \tag{24}$$

where L_{noise} is the objective for training of the denoising module, and L_c is the objective for training of the Feature extraction module and Linear classification module. The two functions are combined as the overall objective in learning:

$$L = L_{noise} + L_c \tag{25}$$

The parameter Settings of MetaCleaner are shown in **Table 1**. MetaCleaner is implemented in Python 3.1.1 based on tensoflow 2.1.0 with Nvidia RTX 4090 GPU.

Table 1. Parameters for MetaCleaner

Parameter	Value
k-mer length	3
k-mer embedding size	100
Convolution kernel size	1,2,13
Convolutional filter number	256,256,256
Neurons of the attention layer	256
Neurons of the feedforward layer	512,256
Batch size	1024
learning rate	0.001
r	30%

4 Results

4.1 Datasets

We collected all phage reference genomes released by the National Center for Biotechnology Information (NCBI) prior to October 31, 2022. To assess the effectiveness of our methods in identifying novel phages, we used genomes published before January 1, 2018, as the training set, while the remaining genomes served as the test set. Additionally, we collected the reference genome of *H.sapiens*(version: GRCh38.p14).

To train our model on each complete genome, we allowed the model to learn features from the complete genome by uniformly extracting non-overlapping substrings of 400bp in length. We also extracted an equal number of such substrings from the complete genome of *H.sapiens* to form the training set. The training set contains 382,755 phage sequences and 382,755 *H.sapiens* sequences. The validation set was generated using the read simulation tool WgSim[29], with specified error rates to provide a more accurate estimation of the model's performance on real sequencing data.

We utilized WgSim to generate 10 simulated sequencing datasets with increasing error rates ranging from 1% to 10%, incremented by 1%. Each dataset comprises 10,000 phage reads and 10,000 *H.sapiens* reads, with lengths varying from 100bp to 400bp.These reads were derived from phage reference genomes released after January 2018 and reference genomes of *H.sapiens* that were not used in the training dataset, respectively.

To evaluate the extended capabilities of MetaCleaner, we obtained a real human gut metagenomic dataset from the National Center For Respiratory Medicine (The First

Affiliated Hospital of Guangzhou Medical University), which included 1,976,820 reads from phage, Corynebacterium striatum strain, Stenotrophomonas maltophilia strain, Pseudomonas aeruginosa, Enterococcus faecalis, Candida albicans, *H.sapiens* and influenza A. Among them, 93,924 reads are from phage. The length of reads ranged from 100bp to 400bp.The number of reads for each species is shown in **Table 2**.

Table 2. A real human gut metagenomic dataset

Class	nums
phage	93924
Corynebacterium striatum strain	471706
Stenotrophomonas maltophilia strain	8318
Pseudomonas aeruginosa	4887
Enterococcus faecalis	1973
Human herpesvirus 5 strain Merlin	2607
Influenza A virus	5
Candida albicans	97
H.sapiens	1393303

4.2 Experiment Setting

We choose Recall, F1-score, Balance-accuracy and Accuracy as our evaluation metrics in our experiments. Recall, F1-score, Balance-accuracy and Accuracy are as follows:

$$Recall = \frac{TP}{TP + FN} \tag{26}$$

$$Precision = \frac{TP}{TP + FP} \tag{27}$$

$$F1score = \frac{2 \times Recall \times Precision}{Recall + Precision} \tag{28}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{29}$$

$$Balance\ accuracy = \frac{1}{2} \times \left(\frac{TP}{TP + FN} + \frac{TN}{TN + FP} \right) \tag{30}$$

Here, TP are positive samples correctly labeled as positives; FP are negative samples incorrectly labeled as positives; TN are negative samples correctly labeled as negatives; and FN are positive samples incorrectly labeled as negatives.Because real metagenomic datasets are class imbalanced datasets, the performance of MetaCleaner and all baseline methods on real metagenomic datasets is evaluated using Balanced accuracy, which has been used in Self-GenomeNet[30].

4.3 Main Results

Performance on simulated datasets with increasing error rates. The genome sequencing process is imperfect and can result in reads containing various types of noise, including base substitutions, insertions, and deletions (INDELs). For example, the most expensive Illumina sequencing technology produces "short" reads of hundreds of bases with an average substitution error rate of less than 1%, while the advent of cheaper sequencing technologies comes with noisier reads, For example, the Oxford Nanopore MinION [15] has error rate close to 10%. To eliminate the effect of noise, MetaCleaner sets the denoising module. To verify the effectiveness of the denoising module, we generated 10 simulated datasets with increasing error rates from 1% to 10% with a growth step of 1% using the sequencing simulation tool WigSim. Each dataset contains 10000 phage sequences and 10000 human sequences. The performance of the baseline models, including MetaPhaPred, CHEER, DeepVirFinder, Virtifier and DETIRE, are compared with MetaCleaner on these 10 datasets, respectively.

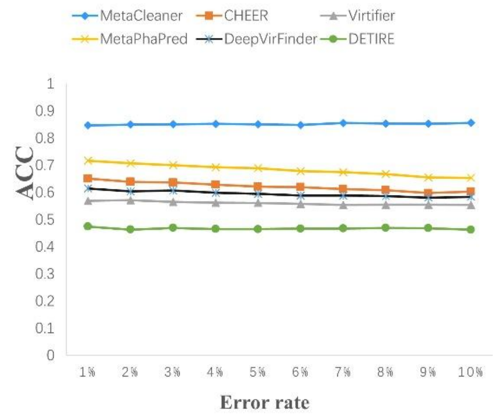

Fig. 3. Accuracy of model on the simulated datasets.The abscissa represents the error rate of the simulated datasets, and the ordinate represents the accuracy of the model on the simulated datasets.

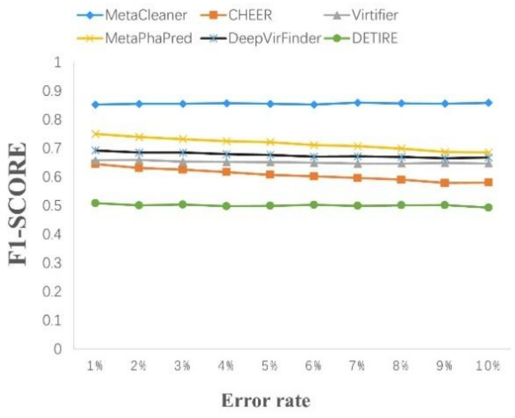

Fig. 4. F1-score of model on the simulated datasets.The abscissa represents the error rate of the simulated datasets, and the ordinate represents the F1-score of the model on the simulated datasets.

Fig. 3 and **Fig. 4** show the accuracy and F1-score of MetaCleaner and other models on the simulated dataset, respectively. We can see that as the error rate increases, the accuracy of the other models continues to decrease. When the error rate increases to 10%, the accuracy of CHEER, Virtifier, MetaPhaPred, DeepVirFinder and DETIRE decreases by 4.84%, 1.51%, 6.34%, 3.06% and 1.22%, respectively. The F1-scores of CHEER, Virtifier, MetaPhaPred, DeepVirFinder and DETIRE decreased by 6.47%, 1.09%, 6.45%, 2.46% and 1.56%, respectively. This proves that sequencing noise affects the performance of deep learning models. However, the accuracy of MetaCleaner does not show a continuous decline. When the error rate increases to 4%, the accuracy of MetaCleaner increases by 0.58%. When the error rate goes up to 6%, MetaCleaner's accuracy drops by 0.43%. When the error rate increases to 10%, MetaCleaner's accuracy increases by 0.78%. It can be seen that the accuracy of MetaCleaner only shows a smaller fluctuation when the error rate of the dataset rises.The same situation occurs for the F1-score of MetaCleaner.

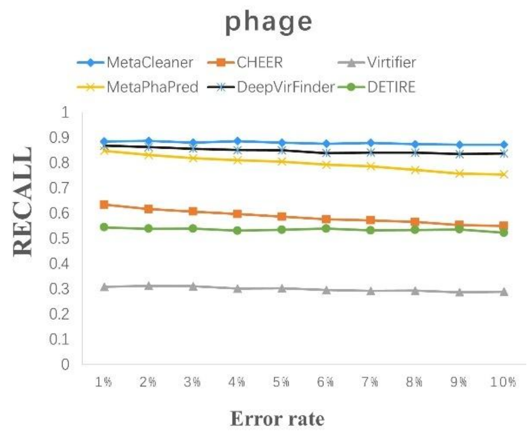

Fig. 5. Recall of model on the phage DNA sequences.The abscissa represents the error rate of the simulated datasets, and the ordinate represents the recall of the model for phage DNA sequences.

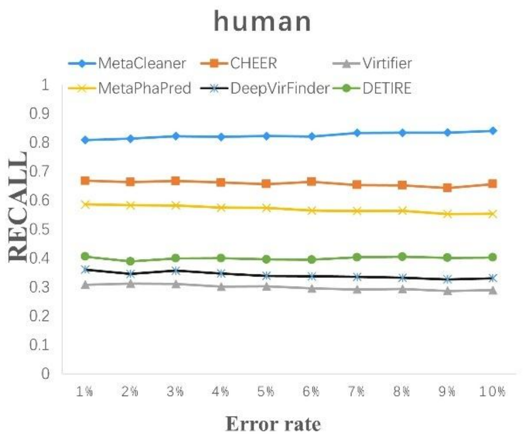

Fig. 6. Recall of model on the human DNA sequences.The abscissa represents the error rate of the simulated datasets, and the ordinate represents the recall of the model for human DNA sequences

Fig. 5 and **Fig. 6** show the Recall of MetaCleaner and other models on the phage DNA sequences and Recall of MetaCleaner and other models on the human DNA sequences, respectively. We can see that as the error rate increases, the recall of the other models for each class continues to decrease. When the error rate grows to 10%, The recall of CHEER, Virtifier, MetaPhaPred, DeepVirFinder and DETIRE for human DNA sequences decreased by 1.2%, 1.9%, 3.28%, 2.94% and 0.32%, respectively. The recall of phage DNA sequences for CHEER, Virtifier, MetaPhaPred, DeepVirFinder and DETIRE decreased by 8.46%, 1.9%, 9.39%, 3.16%and 2.12%, respectively. However, when the error rate increases to 10%, the recall of MetaCleaner pairs and human DNA sequences keeps increasing by 3.14%. The recall of MetaCleaner on bacteriophage DNA sequences decreases by 1.2%. The decrease of MeatCleaner is smaller than that of CHEER,Virtifier, MetaPhaPred, DeepVirFinder and DETIRE.

The above results prove that MetaCleaner is more robust than CHEER, Virtifier, MetaPhaPred, DeepVirFinder and DETIRE. **Fig. 3**, **Fig. 4**,**Fig. 5** and **Fig. 6** also show that MetaCleaner performs better than other models in all metrics, which also proves that MetaCleaner has the ability to distinguish between phage DNA sequences and human DNA sequences.

Performance on real metagenomic dataset. We compare the performance of the baseline models and MetaCleaner on a real metagenomic dataset, and the experimental results are shown in **Fig. 7**, which shows that MetaCleaner achieves higher class-balanced accuracy than all baseline methods on the real metagenomic dataset. The balance accuracy of MetaCleaner is 13.64%, 11%, 14.51%, 14.97% and 21.48% higher than MetaPhaPred, CHEER, DeepVirFinder, Virtifier and DETIRE,respectively.

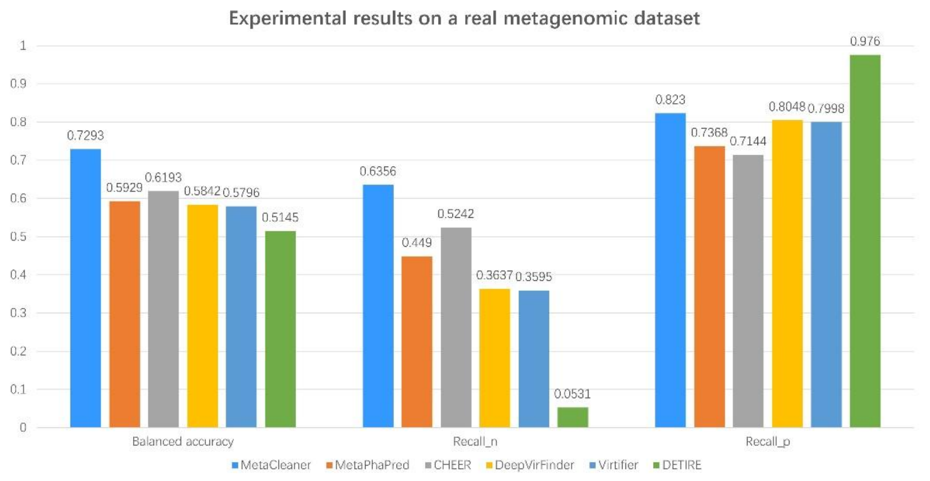

Fig. 7. Experiment results on a real metagenomic dataset. Recalln represents the recall of model on non-phage DNA sequences, Recallp represents the recall of model on phage DNA sequences, and the ordinate represents the value of each metric.

The recall of MetaCleaner to phage DNA sequences was 8.62%,10.86%,1.82% and 2.32% higher than MetaPhaPred, CHEER, DeepVirFinder, Virtifier, respectively.Although DETIRE has the highest recall for phage sequences among all models, its recall for non-phage sequences is only 0.0531%, and its balance accuracy is also the lowest.

The recall of MetaCleaner for non-phage DNA sequences was 18.66%,11.14%,27.19%,27.61% and 58.25%higher than MetaPhaPred, CHEER, DeepVirFinder, Virtifier and DETIRE, respectively.

The above experimental results show that MetaCleaner performs better than MetaPhaPred, CHEER, DeepVirFinder, Virtifier and DETIRE on real metagenomic datasets.

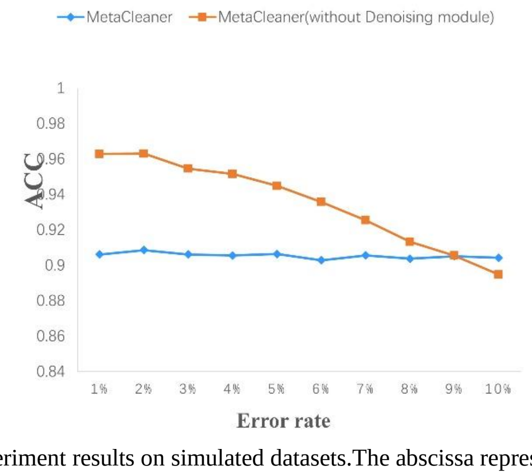

Fig. 8. Ablation experiment results on simulated datasets.The abscissa represents the error rate of the simulated datasets, and the ordinate represents the accuracy of the model on simulated datasets.

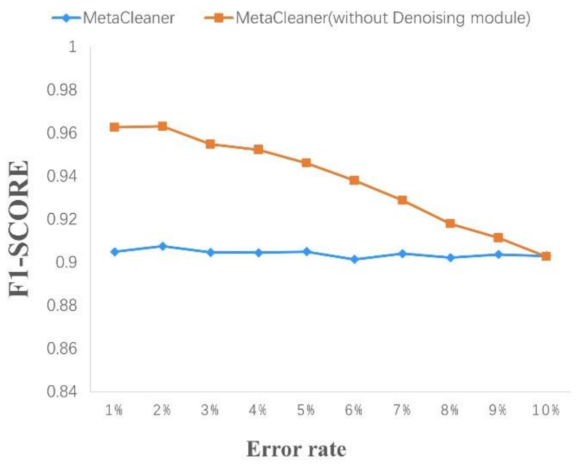

Fig. 9. Ablation experiment results on simulated datasets. The abscissa represents the error rate of the simulated datasets, and the ordinate represents the F1-score of the model on simulated datasets.

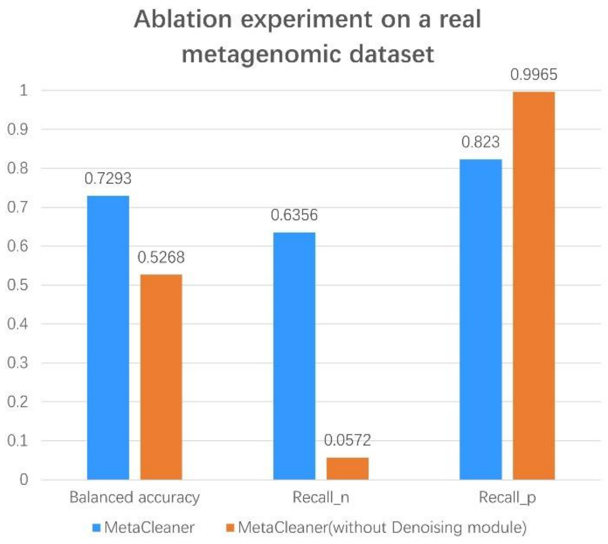

Fig. 10. Ablation experiment results on a real metagenomic dataset. Recalln represents the recall of model on non-phage DNA sequences, Recallp represents the recall of model on phage DNA sequences, and the ordinate represents the value of each metric.

Ablation experiment results on a real metagenomic dataset. To prove the necessity of the denoising module, we removed the denoising module of MetaCleaner and tested its performance on simulated and real metagenomic datasets, and the experimental results are shown in **Fig. 8**,**Fig. 9** and **Fig. 10**, respectively.

It can be seen that the accuracy of MetaCleaner without denoising module continues to decrease as the error rate increases. When the error rate grows to 10%. MetaCleaner without denoising module has 6.81% lower accuracy and 6% lower F1-score. However, MetaCleaner only shows a small range of fluctuations in accuracy and F1-score when using the denoising module. This proves that the denoising module provides robustness to the model.

Although the performance of MetaCleaner without denoising module exceeds that of MetaCleaner with denoising module on simulated datasets, its performance on real metagenomic datasets is much worse than that of MetaCleaner with denoising module. MetaCleaner with denoising module achieves 20.25% higher balanced accuracy on real metagenomic dataset than MetaCleaner without denoising module. This proves that the denoising module contributes to the performance of the model.

Funding. This research was funded by the Guangdong Provincial Science and Technology Project: Key Technologies for Multi-level Knowledge Retrieval Based on Big Data Intelligence [2021B0101420001].

Acknowledgments. The authors would like to express gratitude to the valuable input from the editing reviewers. We also thank to the National Center For Respiratory Medicine (The First Affiliated Hospital of Guangzhou Medical University) for providing datasets for this work.

References

1. M. B. Dion, F. Oechslin, and S. Moineau, "Phage diversity, genomics and phylogeny," Nature Reviews Microbiology, vol. 18, no. 3, pp. 125–138, 2020.
2. A S. Roux, F. Enault, B. L. Hurwitz, and M. B. Sullivan, "Virsorter: mining viral signal from microbial genomic data," PeerJ, vol. 3, p. e985, 2015.
3. V. I. Jurtz, J. Villarroel, O. Lund, M. Voldby Larsen, and M. Nielsen, "Metaphin- der—identifying bacteriophage sequences in metagenomic data sets," PLoS One, vol. 11, no. 9, p. e0163111, 2016.
4. J. Ren, N. A. Ahlgren, Y. Y. Lu, J. A. Fuhrman, and F. Sun, "Virfinder: a novel k-mer based tool for identifying viral sequences from assembled metagenomic data," Microbiome, vol. 5, pp. 1–20, 2017.
5. K. Kieft, Z. Zhou, and K. Anantharaman, "Vibrant: automated recovery, annotation and curation of microbial viruses, and evaluation of viral community function from genomic sequences," Microbiome, vol. 8, pp. 1–23, 2020.
6. J. Guo, B. Bolduc, A. A. Zayed, A. Varsani, G. Dominguez-Huerta, T. O. Del- mont, A. A. Pratama, M. C. Gazitúa, D. Vik, M. B. Sullivan et al., "Virsorter2: a multi-classifier, expert-guided approach to detect diverse dna and rna viruses," Microbiome, vol. 9, pp. 1–13, 2021.
7. J. Ren, K. Song, C. Deng, N. A. Ahlgren, J. A. Fuhrman, Y. Li, X. Xie, R. Poplin, and F. Sun, "Identifying viruses from metagenomic data using deep learning," Quan-titative Biology, vol. 8, no. 1, pp. 64–77, 2020.
8. J. Shang and Y. Sun, "Cheer: Hierarchical taxonomic classification for viral metage-nomic data via deep learning," Methods, vol. 189, pp. 95–103, 2021.
9. Y. Miao, F. Liu, T. Hou, and Y. Liu, "Virtifier: a deep learning-based identifier for viral sequences from metagenomes," Bioinformatics, vol. 38, no. 5, pp. 1216–1222,2022.

10. Y. Miao, J. Bian, G. Dong, and T. Dai, “Detire: a hybrid deep learning model for identifying viral sequences from metagenomes,” Frontiers in Microbiology, vol. 14,p. 1169791, 2023.
11. L. Ma, W. Deng, Y. Bai, Z. Du, M. Xiao, L. Wang, J. Li, and A. K. Nandi, “Identifying phage sequences from metagenomic data using deep neural network with word embedding and attention mechanism,” IEEE/ACM Transactions on Computational Biology and Bioinformatics, 2023.
12. M. Kleiner, L. V. Hooper, and B. A. Duerkop, “Evaluation of methods to purify virus-like particles for metagenomic sequencing of intestinal viromes,” BMC genomics, vol. 16, pp. 1–15, 2015.
13. C. A. Marotz, J. G. Sanders, C. Zuniga, L. S. Zaramela, R. Knight, and K. Zengler,“Improving saliva shotgun metagenomics by chemical host dna depletion,” Microbiome, vol. 6, pp. 1–9, 2018.
14. P. Zhang, Z. Jiang, Y. Wang, and Y. Li, “Clmb: Deep contrastive learning for robust metagenomic binning,” in International Conference on Research in Computational Molecular Biology. Springer, 2022, pp. 326–348.
15. M. Jain, H. E. Olsen, B. Paten, and M. Akeson, “The oxford nanopore minion: delivery of nanopore sequencing to the genomics community,” Genome biology, vol. 17, pp. 1–11, 2016.
16. M. Rojas-Carulla, I. Tolstikhin, G. Luque, N. Youngblut, R. Ley, and B. Schölkopf,“Genet: Deep representations for metagenomics,” arXiv preprint arXiv:1901.11015, 2019.
17. X. Liu, Y. Yu, J. Liu, C. F. Elliott, C. Qian, and J. Liu, “A novel data structure to support ultra-fast taxonomic classification of metagenomic sequences with k-mer signatures,” Bioinformatics, vol. 34, no. 1, pp. 171–178, 2018.
18. P. Ng, “dna2vec: Consistent vector representations of variable-length k-mers,”arXiv preprint arXiv:1701.06279, 2017.
19. Z. Liu, S. S. Venkatesh, and C. C. Maley, “Sequence space coverage, entropy of genomes and the potential to detect non-human dna in human samples,” BMC genomics, vol. 9, pp. 1–17, 2008.
20. P. A. Noble, R. W. Citek, and O. A. Ogunseitan, “Tetranucleotide frequencies in microbial genomes,” Electrophoresis, vol. 19, no. 4, pp. 528–535, 1998.
21. D. D. Kang, J. Froula, R. Egan, and Z. Wang, “Metabat, an efficient tool for accurately reconstructing single genomes from complex microbial communities,” PeerJ, vol. 3, p. e1165, 2015.
22. A. Tampuu, Z. Bzhalava, J. Dillner, and R. Vicente, “Viraminer: Deep learning on raw dna sequences for identifying viral genomes in human samples,” PloS one, vol. 14, no. 9, p. e0222271, 2019.
23. K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, “Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising,” IEEE transactions on image processing, vol. 26, no. 7, pp. 3142–3155, 2017.
24. K. Ma, C. Tang, W. Zhang, B. Cui, K. Ji, Z. Chen, and A. Abraham, “Dc-cnn: Dual-channel convolutional neural networks with attention-pooling for fake news detection,” Applied Intelligence, vol. 53, no. 7, pp. 8354–8369, 2023.
25. Y. Rong, X. Wei, T. Lin, Y. Wang, and E. Kasneci, “Dynstatf: An efficient feature fusion strategy for lidar 3d object detection,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 3238–3247.
26. J. Shen, Y. Chen, Y. Liu, X. Zuo, H. Fan, and W. Yang, “Icafusion: Iterative cross-attention guided feature fusion for multispectral object detection,” Pattern Recognition, vol. 145, p. 109913, 2024.

27. Y. Bussi, R. Kapon, and Z. Reich, “Large-scale k-mer-based analysis of the informational properties of genomes, comparative genomics and taxonomy,” PloS one, vol. 16, no. 10, p. e0258693, 2021.
28. I. Fischer-Hwang, I. Ochoa, T. Weissman, and M. Hernaez, “Denoising of aligned genomic data,” Scientific reports, vol. 9, no. 1, p. 15067, 2019.
29. Li, H. wgsim-Read simulator for next generation sequencing. Github Repository. (2011)
30. Gündüz, H., Binder, M., To, X., Mreches, R., Bischl, B., McHardy, A., Münch, P. & Rezaei, M. A self-supervised deep learning method for data-efficient training in genomics. Communications Biology. 6, 928 (2023)
31. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, Ł. & Polosukhin, I. Attention is all you need. Advances In Neural Information Processing Systems. 30 (2017)