Research Article

Gene Prediction in Metagenomic Fragments with Deep Learning

Shao-Wu Zhang, Xiang-Yang Jin, and Teng Zhang

Key Laboratory of Information Fusion Technology of Ministry of Education, School of Automation,
Northwestern Polytechnical University, Xi'an 710072, China

Correspondence should be addressed to Shao-Wu Zhang; zhangsw@nwpu.edu.cn

Next generation sequencing technologies used in metagenomics yield numerous sequencing fragments which come from thousands of different species. Accurately identifying genes from metagenomics fragments is one of the most fundamental issues in metagenomics. In this article, by fusing multifeatures (i.e., monocodon usage, monoamino acid usage, ORF length coverage, and Z-curve features) and using deep stacking networks learning model, we present a novel method (called Meta-MFDL) to predict the metagenomic genes. The results with 10 CV and independent tests show that Meta-MFDL is a powerful tool for identifying genes from metagenomic fragments.

1. Introduction

Metagenomics bypasses the needs for isolation and lab cultivation of individual species, and the uncultured microbes are sampled directly from their environment [1–3]. High-throughput sequencing technology used in metagenomics can yield millions of short DNA/RNA fragments (or reads) in a single run for the environmental samples and helps us uncover the microbial diversity and better understand how these unknown microbes live and coexist with unprecedented resolution. However, it is in many cases impossible to reliably assemble these short reads into longer contigs because the metagenomic sequencing reads come from thousands of highly uneven different species, and it cannot provide a high sequencing coverage of single species. Several de novo metagenomic short read assemblers such as Meta-IDBA [4], IDBA-UD [5], Ray Meta [6], MetaVelvet-SL [7], Omega [8], and metaSPAdes [9] have shown promising results and effectiveness in assembling metagenomic short reads, but they do not work well in cases where there are more species presenting in the environmental samples [10]. Thus, one way to analyze the metagenomics data is to bypass assembly and go directly finding the genes from these short reads. Gene recognition is a necessary step to fully understand the functions, activities, and roles of genes in cellular processes.

Accurate gene prediction in metagenomes is more complicated than in isolated genomes [11–13]. One reason is that most fragments (reads) from the high-throughput sequencing technologies are very short [11, 14]. Lots of genes are incomplete with one or two ends exceed the fragments, and a single fragment usually contains only one or two genes. Another reason is that the source genomes of the fragments are always unknown or totally new, which brings challenge on statistical model construction and feature selection [11].

Until now, several methods have been developed to predict genes from metagenomics DNA fragments. These prediction approaches can be categorized into homology-based methods, model-based methods, and machine learning-based methods. The homology-based methods such as CRITICA [15] and Orpheus [16] often used the BLAST package to compare the input fragments against known protein databases for analyzing the short sequences. However, these methods just are used to find the genes with previously known homologous proteins and cannot predict novel genes. The model-based methods, such as MetaGeneAnnotator [17], MetaGene [18], MetaGeneMark [19], FragGeneScan [14], and Glimmer-MG [13], used either the higher-order Markov chain models or the hidden Markov chain models to identify genes in metagenomics. However, the main limitation of such Markov chain models is that thousands of parameters are needed in practical use. The machine learning-based

methods such as Orphelia [20, 21], MGC [22], and MetaGUN [11] often formulated the metagenomic fragments with an effective mathematical expression that can truly reflect their intrinsic correlation with the target to be predicted and then designed a powerful classifier with machine learning to operate the gene prediction. Orphelia [20, 21] integrates monocodon and dicodon usage, sequence patterns around translation initiation sites (TISs), open reading frame (ORF) length, and GC-content to represent the ORF fragments, then inputting into an artificial neural network to estimate the probability that a given ORF encodes a protein. As an improvement over the Orphelia algorithm, MGC [22] learns separate models for several predefined GC ranges as opposed to the single model used in Orphelia and applies the appropriate model to each fragment based on its GC-content. That is, MGC uses a two-stage machine learning approach to predict genes by firstly computing six features (i.e., monocodon score, dicodon score, monoamino acid score, diamino acid score, TIS coverage, and TIS probability) from ORF with the corresponding linear discriminant, then estimating the gene probability of a given ORF with the neural network model from the corresponding GC range, which takes nine features of the monocodon score, dicodon score, monoamino acid score, diamino acid score, TIS coverage and TIS probability, length complete, length incomplete, and GC-content as input. MetaGUN [11] implements a three-stage strategy to predict genes by firstly classifying input sequence into different phylogenetic groups, then identifying genes for each group independently with support vector machine (SVM) classifier that integrate entropy density profile of codon usage, translation initiation site scores, and ORF length as the input features, and finally adjusting TISs by employing a modified version of MetaTISA. Although the existing metagenomics gene predictors or methods can effectively identify genes, these predictors employ the shallow architectures such as hidden Markov models (HMMs), SVMs, and multilayer perceptron (MLP) with a single hidden layer. A common property of these shallow learning models is the simple architecture that consists of only one layer responsible for transforming the raw input features into a problem-specific feature space [23], which has been shown effective in solving many simple or well-constrained problems, but its limited modelling and representational power can cause difficulties when dealing with more complicated real-world applications [24].

To further enhance metagenomic gene prediction accuracy, in this study, we developed a new powerful predictor (named as Meta-MFDL) by fusing multiple features of the ORF length coverage, monocodon usage, monoamino acid usage, and Z-curve features and employing the deep learning classification algorithm. Deep learning is a new area of machine learning research, which attempt to model high-level abstractions in data by using model architectures composed of multiple nonlinear transformations [24, 25]. Now, some deep learning architectures such as convolutional deep neural networks (CNNs), deep belief networks (DBNs), deep neural networks (DNNs), and deep stacking networks (DSNs) have been applied to improve the performance in image and speech recognition [26, 27], in natural language processing [28, 29], and most recently in computational

biology [30, 31], such as protein structure and subcellular localization prediction [32–35], gene expression regulation regarding splice junctions or RNA binding proteins [36–42], lncRNA prediction [24], and metagenomic classification [43]. In comparison with the existing predictors, Meta-MFDL showed better performance on the training datasets (i.e., Set700 and Set120) derived from 120 complete genomes in 10-fold cross validation (10 CV) and independent testing datasets (i.e., ^{Tes}Data700 and ^{Tes}Data120) derived from 13 genomes.

2. Materials and Methods

2.1. Datasets. Genomic sequences and annotation information of 120 complete genomes (i.e., 109 bacteria and 11 archaea) and 13 complete genomes (i.e., 10 bacteria and 3 archaea) are obtained from NCBI RefSeq database for separately constructing the training and testing datasets. The genomes of the 13 species used in previous methods [20–22] are not included in the training dataset. All genomes information in the training and testing datasets is listed in Tables S1 and S2 in Supplementary Material available online at https://doi.org/10.1155/2017/4740354. In order to simulate DNA sequences from different sequencing technologies, the genome sequences in training dataset are randomly split into 700 bp and 120 bp fragments with 1-fold genome coverage for each genome, and the Metasim [44] is used to generate 700 bp and 120 bp fragments with the 3-fold genome coverage for each genome in the testing dataset. Now that the purpose of metagenomics gene prediction is to discriminate the coding ORFs from the noncoding ORFs; thus we extracted all ORFs from these fragments and divided these ORFs into coding and noncoding ORFs based on the annotation of the genome. ORFs obtained can be categorized into complete and incomplete ORFs. The complete ORFs have both the start codon and the stop codon. The incomplete ORFs lack the upstream end, the downstream end, or both ends in which case the ORF holds the whole fragments without starting codons or stopping codons [21, 22]. Only ORFs with a minimal length of 60 bp are kept in both training and testing datasets, for the ORFs less than 60 bp is too short to provide useful information [11, 17]. After strictly following the above procedures, we finally obtained two training datasets of ^{Tra}Data700 and ^{Tra}Data120 and two testing datasets of ^{Tes}Data700 and ^{Tes}Data120. ^{Tra}Data700 dataset consists of 1,654,069 coding ORF fragments and 2,267,896 noncoding ORF fragments. ^{Tes}Data700 dataset consists of 303,664 coding ORF fragments and 423,078 noncoding ORF fragments. ^{Tra}Data120 dataset consists of 5,204,861 coding ORF fragments and 5,812,025 noncoding ORF fragments. ^{Tes}Data120 dataset consists of 548,357 coding ORF fragments and 612,049 noncoding ORF fragments. The ^{Tra}Data700, ^{Tes}Data700, ^{Tra}Data120, and ^{Tes}Data120 datasets can be downloaded from http://180.208.58.19/Meta-MFDL/.

In general, establishing a highly useful predictor involves the following five steps [24, 45, 46]: (1) constructing a valid benchmark dataset to train and test the predictor; (2) using effective mathematical expression to convert the nucleotide (or protein) alphabetic sequences into feature vectors that

truly reflect their intrinsic correlation with the attribute to be predicted; (3) developing/choosing a powerful algorithm to operate the prediction; (4) properly selecting the cross-validation tests to objectively evaluate the performance of the predictor; and (5) establishing a software tool. The Meta-MFDL predictor can be divided into three steps: feature extraction, feature fusion, and pattern classification. For feature extraction which is one of the most critical steps in designing a classifier, the query ORF fragments are converted into a series of vectors with the ORF coverage (ORFC), monocodon usage (MCU), monoamino acid usage (MAU), and Z-curve descriptors. For feature fusion, the four kinds of features of ORFC, MCU, MAU, and Z-curve are integrated to represent the candidate ORF fragments. For pattern classification, the vectors are classified by using the deep stacking networks, one of the deep learning architectures.

2.2. Feature Descriptors

2.2.1. ORF Coverage Descriptor. Previous study shows that the length of coding ORFs is considerably longer than that of noncoding ORFs [47]. Thus, we can use the following feature vector X_{ORFC} to represent metagenomic fragment by computing the length proportion of the ORF to the fragment:

$$X_{\mathrm{ORFC}} = \frac{l}{L}, \tag{1}$$

where l is the ORF length and L is the length of fragment.

2.2.2. Monocodon Usage Descriptor. Codon usage is a useful feature in discriminating coding and noncoding ORFs [21]. Here, we use the following vector X_{MC} to represent one fragment by counting the monocodon frequency of ORF (including the complete ORF fragment and incomplete ORF fragment):

$$X_{\mathrm{MC}} = \left[f_1^c, \ldots, f_i^c, \ldots, f_{64}^c \right]$$
$$f_i^c = \frac{n_i}{N}, \tag{2}$$

where f_i^c is the frequency of ith monocodon (e.g., AAA, AAC, ..., UUG, UUU, and three stop codons of UGA, UAA, and UAG) occurring in ORF, n_i is the number of ith codon in ORF, and N is the number of all codons in ORF.

2.2.3. Monoamino Acid Usage Descriptor. The monoamino acid usage is the 21 amino acid (20 amino acids plus one "STOP" codon) frequencies occurring in the DNA sequences, some of which are linearly related to the GC-content of the genome [48] and are important in discriminating coding and noncoding DNA sequences [22]. Here, we use the following vector X_{MA} to represent one fragment by counting the monoamino acid frequency of ORF (including the complete ORF fragment and incomplete ORF fragment).

$$X_{\mathrm{MA}} = \left[f_1^{\mathrm{aa}}, \ldots, f_j^{\mathrm{aa}}, \ldots, f_{21}^{\mathrm{aa}} \right], \tag{3}$$

where f_j^{aa} is the frequency of jth amino acid (e.g., A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y, and one "STOP" codon) occurring in ORF.

2.2.4. Z-Curve Parameter Descriptor. Z-curve parameter features are calculated for the frequencies of frame-dependent k-mers by using the Z-transform of DNA sequences [49] and have been successfully applied to gene prediction [50, 51]. Let $a_1, c_1, g_1, t_1; a_2, c_2, g_2, t_2; a_3, c_3, g_3$, and t_3 represent the frequencies of bases A, C, G, and T occurring at positions 1, 4, 7, 10, ...; 2, 5, 8, 11, ...; 3, 6, 9, 12, ... and in an ORF fragment, respectively. The $p_{12}(\mathrm{AA}), p_{12}(\mathrm{AC}), \ldots, p_{12}(\mathrm{TT})$; $p_{23}(\mathrm{AA}), p_{23}(\mathrm{AC}), \ldots$, and $p_{23}(\mathrm{TT})$ denote the frequencies of the 16 dinucleotides AA, AC, ..., and TT occurring at the codon positions 1-2 and 2-3 of an ORF fragment, respectively. By using the Z-transform of DNA sequence [50], we can use the following vectors $X_{\mathrm{ZCPS}}, X_{\mathrm{ZCPD}}$ to represent the ORF fragment by counting the frequencies of codon-position-dependent single nucleotides and the frequencies of phase-specific dinucleotides, respectively.

$$X_{\mathrm{ZCPS}} = \left[x_1, y_1, z_1, x_2, y_2, z_2, x_3, y_3, z_3 \right]$$
$$x_i = (a_i + g_i) - (c_i + t_i)$$
$$y_i = (a_i + c_i) - (g_i + t_i)$$
$$z_i = (a_i + t_i) - (g_i + c_i)$$
$$x_i, y_i, z_i \in [-1, 1], \quad i = 1, 2, 3$$
$$X_{\mathrm{ZCPD}} = \left[x_{12}^{\mathrm{A}}, y_{12}^{\mathrm{A}}, z_{12}^{\mathrm{A}}, \ldots, x_{12}^{\mathrm{T}}, y_{12}^{\mathrm{T}}, z_{12}^{\mathrm{T}}, x_{23}^{\mathrm{A}}, y_{23}^{\mathrm{A}}, z_{23}^{\mathrm{A}}, \ldots, \right.$$
$$\left. x_{23}^{\mathrm{T}}, y_{23}^{\mathrm{T}}, z_{23}^{\mathrm{T}} \right] \tag{4}$$
$$x_k^M = ((p_k(\mathrm{MA}) + p_k(\mathrm{MG}))$$
$$- (p_k(\mathrm{MC}) + p_k(\mathrm{MT})))$$
$$y_k^M = ((p_k(\mathrm{MA}) + p_k(\mathrm{MC}))$$
$$- (p_k(\mathrm{MG}) + p_k(\mathrm{MT})))$$
$$z_k^M = ((p_k(\mathrm{MA}) + p_k(\mathrm{MT})) - (p_k(\mathrm{MC}) + p_k(\mathrm{MG})))$$
$$M = \mathrm{A, C, G, T}; \quad k = 12, 23.$$

2.3. Feature Fusion. Feature fusion can derive the most discriminatory information from original multifeature sets and eliminate the redundant information from the correlation between distinct feature sets, which benefits the final decision. Here, four kinds of feature set of ORF coverage, monocodon usage, monoamino acid usage, and Z-curve parameters are concatenated into one set of feature vectors to represent the ORF fragments, which can be formulized as follows:

$$X = \left[X_{\mathrm{ORFC}}, X_{\mathrm{MC}}, X_{\mathrm{MA}}, X_{\mathrm{ZCPS}}, X_{\mathrm{ZCPD}} \right]. \tag{5}$$

2.4. Deep Stacking Network. As one of the deep neural networks, deep stacking network (DSN) has been successfully applied in speech classification [52], information retrieval [53], and lncRNA identification [24]. A DSN is stacked by a series of modules with the same or similar structures. Each DSN module takes a simplified form of the shallow multilayer

perceptron, consisting of a linear input layer, a nonlinear hidden layer and a linear output layer. "Stacking" is accomplished by concatenating the output of all previous modules with the raw input vector to form the new "input" vector as the input of the next module. The DSN weight parameters W (input weight matrices) and U (output network weight matrices) in each module are learned efficiently from the training data by using the basic learning algorithm and the fine tuning algorithm.

2.4.1. Basic Learning Algorithm. Let $\mathbf{X} = [\mathbf{x_1}, \ldots, \mathbf{x_i}, \ldots, \mathbf{x_N}]$ represent the training vectors, in which each vector is denoted by $\mathbf{x}_i = [x_{1i}, \ldots, x_{ui}, \ldots, x_{Di}]^T$; $\mathbf{T} = [\mathbf{t_1}, \ldots, \mathbf{t_i}, \ldots, \mathbf{t_N}]$ represents the target vectors, in which each vector is denoted by $\mathbf{t}_i = [t_{1i}, \ldots, t_{vi}, \ldots, t_{Ci}]^T$, where N is total number of training samples, D is the dimension of the input vector, and C is dimension of the output vector. Let L denote the number of hidden units. Then, the output of a DSN module is $\mathbf{y_i} = \mathbf{U^T h_i}$, where $\mathbf{h_i} = \sigma(\mathbf{W^T x_i})$ is the ith hidden layer output, $\mathbf{U}$ is a $L \times C$ weight matrix at the upper layer, $\mathbf{W}$ is a $D \times L$ weight matrix at the lower layer, and $\sigma(\cdot)$ is the sigmoid function. For a given $\mathbf{W}$, the parameter $\mathbf{U}$ is learned to minimize the average of the total square error.

$$E = \|\mathbf{Y} - \mathbf{T}\|^2 = \mathrm{Tr}\left[(\mathbf{Y} - \mathbf{T})(\mathbf{Y} - \mathbf{T})^{\mathbf{T}}\right], \qquad (6)$$

where $\mathbf{Y} = [\mathbf{y_1}, \ldots, \mathbf{y_i}, \ldots, \mathbf{y_N}]$. If the lower layer weight matrix $\mathbf{W}$ is fixed, the hidden layer output values $\mathbf{H}$ can also been calculated. Consequently, the upper-layer weight matrix $\mathbf{U}$ in each module can be determined by setting the gradient $\partial E/\partial \mathbf{U} = 2\mathbf{H}(\mathbf{U}^T\mathbf{H} - \mathbf{T})^T$ to zero, then leading to the closed-form solution.

$$\mathbf{U} = \left(\mathbf{HH}^T\right)^{-1} \mathbf{HT}^T. \qquad (7)$$

In general, there are two ways to set $\mathbf{W}$: firstly, by using various distributions to generate the random numbers to set $\mathbf{W}$; secondly, by applying contrastive divergence to separately train the restricted Boltzmann machines (RBM), then using the trained RBM weights to set $\mathbf{W}$. In this paper, we used the trained RBM weights to set $\mathbf{W}$ for the bottom module.

2.4.2. Module-Bound Fine Tuning. The weight matrices $\mathbf{W}$ of DSN in each module can be further learned using the batch-mode gradient descent [54]. That is,

$$\frac{\partial E}{\partial \mathbf{W}} = \frac{\partial \mathrm{Tr}\left[\left(\mathbf{U}^T\mathbf{H} - \mathbf{T}\right)\left(\mathbf{U}^T\mathbf{H} - \mathbf{T}\right)^T\right]}{\partial \mathbf{W}} = 2\mathbf{X}\left[\mathbf{H}^T\right.$$
$$\left. \circ (1 - \mathbf{H})^T \circ \left[\mathbf{H}^\dagger\left(\mathbf{HH}^T\right)\left(\mathbf{TH}^\dagger\right) - \mathbf{T}^T\left(\mathbf{TH}^\dagger\right)\right]\right] \qquad (8)$$
$$\mathbf{W}^{(j+1)} = \mathbf{W}^{(j)} + \eta \times \frac{\partial E^{(j)}}{\partial \mathbf{W}^{(j)}},$$

where $\mathbf{H}^\dagger = \mathbf{H}^T(\mathbf{HH}^T)^{-1}$ is pseudoinverse of $\mathbf{H}$, the symbol $\circ$ represents the element-wise matrix multiplication, and η is the learning rate of updating the weight matrices $\mathbf{W}$.

2.5. The Performance Measures of the Prediction System. The performance measures of the sensitivity or true positive rate (TPR), precision or positive predictive value (PPV), accuracy (ACC), and $F1$ score were used to evaluate the performance of the prediction system. They are defined as follows:

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$
$$\mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$
$$F1 = \frac{2 \times \mathrm{PPV} \times \mathrm{TPR}}{\mathrm{PPV} + \mathrm{TPR}} \qquad (9)$$
$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}},$$

where TP is the number of correctly predicted coding genes and FP and FN are the number of incorrectly predicted coding genes and noncoding gene, respectively. $F1$ score is the harmonic mean of precision and sensitivity, which provides a composite of precision and sensitivity.

3. Results and Discussion

3.1. Performance of Meta-MFDL. In statistical prediction, the following three cross-validation methods are often used to examine a predictor for its effectiveness in practical application: independent dataset test, K-fold (e.g., 5-fold, 10-fold) crossover or subsampling test, and jackknife test. Of the three test methods, the jackknife test is deemed the least arbitrary that can always yield a unique result for a given benchmark dataset [45]. However, for large scale database, the jackknife test needs to spend lots of time to generate the prediction results. To reduce the computational time and evaluate the generalization performance of a predictor, in this study, we adopted the 10-fold cross-validation (10 CV) test and independent dataset test as done by most investigators [24, 46, 55–59].

Generally, more ORF fragments are used to train the deep learning model, higher gene prediction accuracy can be obtained, but it will spend more computational time. By making a trade-off between the prediction accuracy and computational time, we randomly select 50,000 coding ORF fragments and 50,000 noncoding ORF fragments from $^{\mathrm{Tra}}$Data700 and $^{\mathrm{Tra}}$Data120, respectively, to form Set700 and Set120 datasets to train the Meta-MFDL predictor. To demonstrate the superiority of Meta-MFDL, in 10 CV test, we compared it with other two state-of-the-art predictors, Orphelia [21, 22] and FragGeneScan [14]. Orphelia [21, 22] firstly extracts the features of monocodon usage, dicodon usage, and translation initiation sites by building linear discriminants and then combines these features with ORF length and GC-content by using an artificial neural network to estimate the probability of an ORF encoding a protein. FragGeneScan [14] combines codon usage, sequence patterns for start/stop codons, and sequencing error models in a hidden Markov model to improve the prediction of protein-coding region in short reads. In order to show the outstanding performance of

TABLE 1: The performance of Orphelia, FragGeneScan, Meta-MFSVM, and Meta-MFDL on the Set700 and Set120 training datasets in 10 CV test.

Predictors	Set700			Set120		
	TPR (%)	PPV (%)	F1	TPR (%)	PPV (%)	F1
Orphelia	88.61 ± 1.89	89.05 ± 1.51	0.888 ± 0.016	83.19 ± 0.98	82.44 ± 1.20	0.847 ± 0.985
Meta-MFSVM	89.59 ± 2.39	90.23 ± 1.34	0.908 ± 0.018	84.70 ± 0.66	85.07 ± 1.26	0.849 ± 0.543
FragGeneScan	90.38 ± 2.23	91.89 ± 1.98	0.918 ± 0.013	86.56 ± 0.71	86.78 ± 0.54	0.868 ± 0.013
Meta-MGPDL	91.47 ± 1.37	93.26 ± 1.97	0.923 ± 0.008	89.28 ± 0.63	90.58 ± 0.61	0.899 ± 0.006

TABLE 2: The overall accuracy (%) of Orphelia, FragGeneScan, MGC, MetaGUN, Meta-MFSVM, and Meta-MFDL on the [Tes]Data700 independent testing dataset.

Species	Orphelia	Meta-MFSVM	MGC	FragGeneScan	MetaGUN	Meta-MFDL
A. fulgidus	89.49	92.25	92.47	93.14	94.30	94.70
N. pharaonis	84.73	89.87	89.90	91.67	94.40	94.12
M. jannaschii	86.13	91.68	91.69	95.91	95.03	94.65
B. aphidicola	94.16	93.43	93.75	95.60	96.24	96.05
B. pseudomallei	88.19	90.71	90.76	91.53	94.13	94.15
B. subtilis	91.56	92.55	91.78	92.49	93.86	93.89
C. jeikeium	86.03	90.15	90.85	91.32	91.31	91.62
C. tepidum	83.59	88.48	88.43	88.07	90.68	90.66
E. coli	91.50	91.98	92.06	93.45	93.58	93.66
H. pylori	92.88	93.30	93.48	94.98	93.81	94.49
P. aeruginosa	85.05	92.42	92.47	94.42	94.03	93.89
P. marinus	89.16	91.24	91.36	90.82	93.73	93.88
W. endosymbiont	78.66	84.70	84.78	75.67	83.69	87.34
Average	87.78	90.98	91.06	91.47	92.98	93.31

deep learning algorithm, we also design a Meta-MFSVM predictor. Meta-MFSVM uses the same features as Meta-MFDL, inputting the support vector machine (SVM) to distinguish between coding and noncoding ORF fragments. The results of four predictors on the same training datasets (i.e., Set700 and Set120) in 10 CV test are shown on Table 1. It can be seen that, for 700 bp dataset, the TPR of Meta-MFDL is 91.47%, which is 2.86, 1.88, and 1.09% higher than that of Orphelia, Meta-MFSVM, and FragGeneScan, respectively; the PPV of Meta-MFDL is 93.26%, which is 4.21, 3.03, and 1.37% higher than that of Orphelia, Meta-MFSVM, and FragGeneScan, respectively; F1 score of Meta-MFDL is 0.923, which is 0.035, 0.015, and 0.005 higher than that of Orphelia, Meta-MFSVM, and FragGeneScan, respectively; most deviations of Meta-MFDL are also lower than that of Orphelia, Meta-MFSVM, and FragGeneScan. For 120 bp dataset, the TPR of Meta-MFDL is 89.28%, which is 6.09, 4.58, and 2.72% higher than that of Orphelia, Meta-MFSVM, and FragGeneScan, respectively; the PPV of Meta-MFDL is 90.58%, which is 8.14, 5.51, and 3.8% higher than that of Orphelia, Meta-MFSVM, and FragGeneScan, respectively; F1 score of Meta-MFDL is 0.899, which is 0.052, 0.05, and 0.031 higher than that of Orphelia, Meta-MFSVM, and FragGeneScan, respectively; all deviations of Meta-MFDL are lower than that of Orphelia, Meta-MFSVM, and FragGeneScan. The results of 700 bp and 120 bp datasets show that Meta-MFDL can effectively recognize the short coding fragments in metagenomics than Orphelia and FragGeneScan predictors.

In addition, the comparing results of Meta-MFDL and Meta-MFSVM indicate that the classifying ability of deep learning is superior to SVM. These results show that the Meta-MFDL predictor has the powerful performance for distinguishing coding and noncoding ORF fragments.

In order to further evaluate the generalized performance of Meta-MFDL predictor, we also implemented Meta-MFDL on [Tes]Data700 and [Tes]Data120 testing datasets derived from 13 species (i.e., 10 bacteria and 3 archaea) and compared with Orphelia [21, 22], FragGeneScan [14], MGC [23], MetaGUN [11], and Meta-MFSVM predictors. The overall accuracies of Meta-MFDL, Meta-MFSVM, Orphelia, FragGeneScan, MGC, and MetaGUN on the two independent testing datasets (i.e., [Tes]Data700 and [Tes]Data120) are shown in Tables 2 and 3, respectively. The TPR, PPV, and F1 score of these six methods are listed in Tables S3 and S4 available in Supplementary Material. It can be seen that both Meta-MFDL and MetaGUN achieve better performance than Orphelia, Meta-MFSVM, MGC, and FragGeneScan on most species. The performance of Meta-MFDL is slightly better than that of MetaGUN; for example, for the [Tes]Data700 independent testing dataset, the average accuracy of Meta-MFDL is 93.31%, which is 0.33% higher than that of MetaGUN; the average F1 score is 0.9184, which is 0.0029 higher than that of MetaGUN.

Table 3: The overall accuracy (%) of Orphelia, FragGeneScan, MGC, MetaGUN, Meta-MFSVM, and Meta-MFDL on the [Tes]Data120 independent testing dataset.

Species	Orphelia	Meta-MFSVM	MGC	FragGeneScan	MetaGUN	Meta-MFDL
A. fulgidus	84.05	85.22	85.67	85.92	86.49	87.38
N. pharaonis	82.43	82.79	82.96	82.94	83.12	83.03
M. jannaschii	82.11	84.25	84.69	84.80	85.23	85.59
B. aphidicola	85.91	87.50	88.21	88.96	89.96	90.06
B. pseudomallei	84.28	84.73	84.93	85.10	85.37	85.60
B. subtilis	88.13	88.36	88.46	88.51	88.60	87.85
C. jeikeium	80.95	83.19	84.64	85.96	87.38	91.04
C. tepidum	79.17	79.67	80.02	80.16	80.52	79.22
E. coli	85.98	86.31	86.56	86.70	86.89	86.91
H. pylori	88.09	89.37	90.66	91.42	92.51	92.81
P. aeruginosa	83.71	84.28	84.57	84.84	85.15	84.67
P. marinus	88.26	88.76	88.98	89.11	89.71	88.73
W. endosymbiont	74.75	76.50	77.72	78.51	79.23	79.55
Average	83.68	84.69	85.24	85.61	86.17	86.34

These results suggest that the Meta-MFDL predictor has a better generalized performance. In addition, MetaGUN used 261 complete genomes (229 bacteria and 32 archaea) to train the predictor, while our Meta-MFDL just used 120 complete genomes (109 bacteria and 11 archaea) to train the predictor. If we use the same complete genomes as MetaGUN to train Meta-MFDL, Meta-MFDL can achieve better generalized performance than that of MetaGUN.

3.2. Effects of Training Data Size and Randomly Sampling.
In general, if we use more ORF fragments to train the deep learning model, we can obtain higher prediction accuracy, but it will spend more computational time. In order to investigate the effects of the training data size, we randomly sample different number fragments (e.g., 5000, 10,000, 50,000, 80,000, 100,000, 150,000, and 200,000) from [Tra]Data700 to build the training dataset in which the number of coding ORF fragments is the same as the noncoding fragments and randomly sample the same number fragments as the training dataset to test the predictors. The results of different training data size are listed in Table S5 available in Supplementary Material. From Table S5, we can see that the metrics of TPR, PPV, ACC, and $F1$ score gradually increase with the increase of size of training data, while the running time significantly increases. For example, the accuracy and running time of Meta-MFDL on the 100,000 and 150,000 fragment training datasets are 92.6%, 92.7%, 1.027 h, and 1.45 h, respectively. The accuracy on the 100,000 dataset just increases 0.1% than 150,000 dataset, while the running time increases 0.423 h. By making a trade-off between the predict metrics and running time, we randomly select 100,000 fragments (i.e., 50,000 coding ORF fragments and 50,000 noncoding ORF fragments) to train our Meta-MFDL predictor in this paper.

In order to investigate the effects of randomly sampling strategy, we randomly sample 50,000 coding ORF fragments

Table 4: The effects of randomly sampling strategy to the Meta-MFDL predictor in 10 CV test.

Sampling times	TPR (%)	PPV (%)	$F1$
(1)	91.27 ± 1.00	91.79 ± 1.10	0.915 ± 0.009
(2)	91.38 ± 0.92	91.38 ± 0.92	0.916 ± 0.007
(3)	91.64 ± 0.66	91.74 ± 0.88	0.916 ± 0.007
(4)	92.09 ± 0.95	91.73 ± 0.92	0.919 ± 0.008
(5)	92.16 ± 0.77	91.93 ± 0.51	0.920 ± 0.006
(6)	92.22 ± 0.63	91.98 ± 0.53	0.922 ± 0.004
(7)	92.14 ± 0.82	92.57 ± 0.91	0.924 ± 0.006
(8)	92.40 ± 1.26	92.64 ± 0.87	0.925 ± 0.007
(9)	91.92 ± 0.67	93.28 ± 1.50	0.926 ± 0.087
(10)	91.47 ± 1.37	93.26 ± 1.97	0.923 ± 0.008
Average	91.87 ± 0.90	92.23 ± 1.01	0.921 ± 0.015

and 50,000 noncoding ORF fragments from [Tra]Data700 ten times and use the 10 CV test to assess the performance of Meta-MFDL. Table 4 gives the results of ten randomly samplings, in which we can see that randomly sampling strategy has little effect to Meta-MFDL predictor. Thus, the strategy of randomly sampling 50,000 coding ORF fragments and 50,000 noncoding ORF fragments to train the Meta-MFDL model is reasonable and feasible, which can reduce computational time and gives better prediction results.

3.3. Comparison with Individual Feature Classifier.
To further verify the effectiveness of Meta-MFDL predictor, we compared it with four other individual feature deep learning classifiers based on the ORF coverage (ORFC), monocodon usage (MC), monoamino acid usage (MA), and Z-curve parameter (ZC) feature descriptors, respectively. The results

TABLE 5: Results of Meta-MFDL and four individual feature deep learning classifiers on Set700 dataset in 10 CV test.

Features	TPR (%)	PPV (%)	$F1$
ORFC-DL	87.64 ± 1.24	92.28 ± 1.35	0.899 ± 0.003
MC-DL	87.97 ± 0.75	92.37 ± 1.05	0.899 ± 0.003
MA-DL	88.53 ± 0.49	92.68 ± 0.86	0.905 ± 0.004
ZC-DL	88.95 ± 0.36	93.85 ± 0.75	0.913 ± 0.002
Meta-MFDL	91.47 ± 1.37	93.26 ± 1.97	0.923 ± 0.008

on the Set700 dataset in 10 CV test are shown in Table 5, from which we can see that the TPR of Meta-MFDL is 3.83, 3.5, 2.94, and 2.52% higher than that of ORFC-DL, MC-DL, MA-DL, and ZC-DL classifiers, respectively; the PPV is little higher than that of ORFC-DL, MC-DL, and MA-DL, and slightly lower than that of ZC-DL; $F1$ score of Meta-MFDL is also bigger than that of other four individual feature classifiers, suggesting that our feature fusion strategy is effective for identifying the gene fragments. The MA and ZC feature descriptors are more powerful than ORFC and MC feature descriptors, meaning that MA and ZC features contribute the most to the overall performance of Meta-MFDL predictor. These results show that Meta-MFDL predictor is effective and robust for predicting metagenomic gene compared with the individual feature DL classifier.

4. Conclusions

Identification of genes directly from metagenomic fragments is an important task in annotating metagenomes. However, due to the incomplete and fragmented nature of the metagenomic sequencing data, it is more complicated in metagenomes than in isolated genomes, and the assembly of metagenomes is often not available. Therefore, it is important to develop the computational methods for identifying the coding ORFs from the metagenomics short reads. In this study, based on the DNA sequences, we introduced four kinds of feature descriptors (i.e., ORFC, MC, MA, and ZC) and fused them forming a vector to represent the ORF fragments. Instead of the shallow learning models (e.g., SVM, HMM), we used DSN deep learning architecture model to design the Meta-MFDL predictor for identifying the metagenomic gene fragments. Comparing with the existing Orphelia, FragGeneScan, MGC, and MetaGUN predictors for identifying metagenomics genes on the training datasets and other 13 species independent testing datasets, Meta-MFDL predictor shows strong robust and powerful ability for identifying metagenomic gene fragments, and it represents an intriguing and promising avenue for predicting metagenomic genes. Meta-MFDL software package is available at http://180.208.58.19/Meta-MFDL/.

Conflicts of Interest

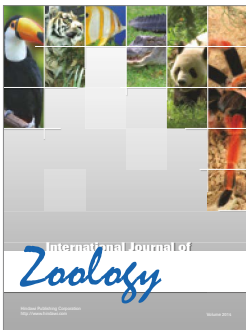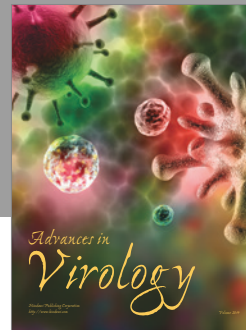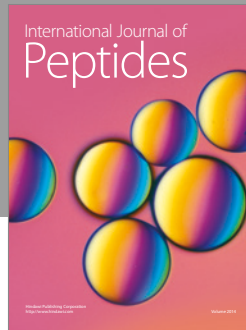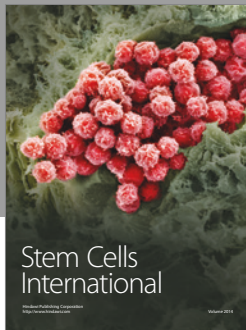The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] C. S. Riesenfeld, P. D. Schloss, and J. Handelsman, "Metagenomics: genomic analysis of microbial communities," *Annual Review of Genetics*, vol. 38, no. 1, pp. 525–552, 2004.

[2] J. C. Venter, K. Remington, J. F. Heidelberg et al., "Environmental genome shotgun sequencing of the sargasso sea," *Science*, vol. 304, no. 5667, pp. 66–74, 2004.

[3] S. G. Tringe, C. Von Mering, A. Kobayashi et al., "Comparative metagenomics of microbial communities," *Science*, vol. 308, no. 5721, pp. 554–557, 2005.

[4] Y. Peng, H. C. M. Leung, S. M. Yiu, and F. Y. L. Chin, "Meta-IDBA: a de Novo assembler for metagenomic data," *Bioinformatics*, vol. 27, no. 13, pp. i94–i101, 2011.

[5] Y. Peng, H. C. M. Leung, S. M. Yiu, and F. Y. L. Chin, "IDBA-UD: a de novo assembler for single-cell and metagenomic sequencing data with highly uneven depth," *Bioinformatics*, vol. 28, no. 11, Article ID bts174, pp. 1420–1428, 2012.

[6] S. Boisvert, F. Raymond, E. Godzaridis, F. Laviolette, and J. Corbeil, "Ray Meta: scalable *de novo* metagenome assembly and profiling," *Genome Biology*, vol. 13, article R122, 2012.

[7] Afiahayati, K. Sato, and Y. Sakakibara, "MetaVelvet-SL: An extension of the Velvet assembler to a de novo metagenomic assembler utilizing supervised learning," *DNA Research*, vol. 22, no. 1, pp. 69–77, 2015.

[8] B. Haider, T.-H. Ahn, B. Bushnell, J. Chai, A. Copeland, and C. Pan, "Omega: An Overlap-graph de novo Assembler for Metagenomics," *Bioinformatics*, vol. 30, no. 19, pp. 2717–2722, 2014.

[9] S. Nurk, D. Meleshko, A. Korobeynikov, and P. A. Pevzner, "metaSPAdes: a new versatile metagenomic assembler," *Genome Research*, vol. 27, no. 5, pp. 824–834, 2017.

[10] J. Vollmers, S. Wiegand, and A.-K. Kaster, "Comparing and evaluating metagenome assembly tools from a microbiologist's perspective - Not only size matters!," *PLoS ONE*, vol. 12, no. 1, Article ID e0169662, 2017.

[11] Y. Liu, J. Guo, G. Hu, and H. Zhu, "Gene prediction in metagenomic fragments based on the SVM algorithm," *BMC Bioinformatics*, vol. 14, no. 5, article no. S12, 2013.

[12] D. Hyatt, P. F. Locascio, L. J. Hauser, and E. C. Uberbacher, "Gene and translation initiation site prediction in metagenomic sequences," *Bioinformatics*, vol. 28, no. 17, Article ID bts429, pp. 2223–2230, 2012.

[13] D. R. Kelley, B. Liu, A. L. Delcher, M. Pop, and S. L. Salzberg, "Gene prediction with Glimmer for metagenomic sequences augmented by classification and clustering," *Nucleic Acids Research*, vol. 40, no. 1, p. e9, 2012.

[14] M. Rho, H. Tang, and Y. Ye, "FragGeneScan: predicting genes in short and error-prone reads," *Nucleic Acids Research*, vol. 38, no. 20, p. e191, 2010.

[15] J. H. Badger and G. J. Olsen, "CRITICA: Coding region identification tool invoking comparative analysis," *Molecular Biology and Evolution*, vol. 16, no. 4, pp. 512–524, 1999.

[16] D. Frishman, A. Mironov, H.-W. Mewes, and M. Gelfand, "Combining diverse evidence for gene recognition in completely sequenced bacterial genomes," *Nucleic Acids Research*, vol. 26, no. 12, pp. 2941–2947, 1998.

[17] H. Noguchi, T. Taniguchi, and T. Itoh, "MetaGeneAnnotator: detecting species specific patterns of ribosomal binding site for precise gene prediction in anonymous prokaryotic and phage genomes," *DNA Research*, vol. 15, no. 6, pp. 387–396, 2008.

[18] H. Noguchi, J. Park, and T. Takagi, "MetaGene: prokaryotic gene finding from environmental genome shotgun sequences," *Nucleic Acids Research*, vol. 34, no. 19, pp. 5623–5630, 2006.

[19] W. Zhu, A. Lomsadze, and M. Borodovsky, "*Ab initio* gene identification in metagenomic sequences," *Nucleic Acids Research*, vol. 38, article e132, Article ID gkq275, 2010.

[20] K. J. Hoff, T. Lingner, P. Meinicke, and M. Tech, "Orphelia: Predicting genes in metagenomic sequencing reads," *Nucleic Acids Research*, vol. 37, no. 2, pp. W101–W105, 2009.

[21] K. J. Hoff, M. Tech, T. Lingner, R. Daniel, B. Morgenstern, and P. Meinicke, "Gene prediction in metagenomic fragments: A large scale machine learning approach," *BMC Bioinformatics*, vol. 9, article no. 217, 2008.

[22] A. El Allali and J. R. Rose, "MGC: A metagenomic gene caller," *BMC Bioinformatics*, vol. 14, no. 9, article no. S6, 2013.

[23] D. Yu, L. Deng, I. Jang, P. Kudumakis, M. Sandler, and K. Kang, "Deep learning and its applications to signal and information processing," *IEEE Signal Processing Magazine*, vol. 28, no. 1, pp. 145–154, 2011.

[24] X.-N. Fan and S.-W. Zhang, "LncRNA-MFDL: identification of human long non-coding RNAs by fusing multiple features and using deep learning," *Molecular BioSystems*, vol. 11, no. 3, pp. 892–897, 2015.

[25] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[26] J. Tompson, A. Jain, Y. LeCun, and C. Bregler, "Joint training of a convolutional network and a graphical model for human pose estimation," in *Proceedings of the 28th Annual Conference on Neural Information Processing Systems 2014, NIPS 2014*, pp. 1799–1807, can, December 2014.

[27] G. Hinton, L. Deng, D. Yu et al., "Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[28] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, pp. 2493–2537, 2011.

[29] C. Xiong, S. Merity, and R. Socher, "Dynamic memory networks for visual and textual question answering," in *Proceedings of the 33rd International Conference on Machine Learning, ICML 2016*, pp. 3574–3583, usa, June 2016.

[30] P. Mamoshina, A. Vieira, E. Putin, and A. Zhavoronkov, "Applications of Deep Learning in Biomedicine," *Molecular Pharmaceutics*, vol. 13, no. 5, pp. 1445–1454, 2016.

[31] C. Angermueller, T. Pärnamaa, L. Parts, and O. Stegle, "Deep learning for computational biology," *Molecular Systems Biology*, vol. 12, no. 7, article no. 878, 2016.

[32] J. Lyons, A. Dehzangi, R. Heffernan et al., "Predicting backbone Cα angles and dihedrals from protein sequences by stacked sparse auto-encoder deep neural network," *Journal of Computational Chemistry*, vol. 35, no. 28, pp. 2040–2046, 2014.

[33] R. Heffernan, K. Paliwal, J. Lyons et al., "Improving prediction of secondary structure, local backbone angles, and solvent accessible surface area of proteins by iterative deep learning," *Scientific Reports*, vol. 5, Article ID 11476, 2015.

[34] M. Spencer, J. Eickholt, and J. Cheng, "A deep learning network approach to," *IEEE Transactions on Computational Biology and Bioinformatics*, vol. 12, no. 1, pp. 103–112, 2015.

[35] S. r. Sønderby, C. K. Sønderby, H. Nielsen, and O. Winther, "Convolutional LSTM networks for subcellular localization of proteins," in *Algorithms for computational biology*, vol. 9199 of *Lecture Notes in Comput. Sci.*, pp. 68–80, Springer, Cham, 2015.

[36] S. Cheng, M. Guo, C. Wang, X. Liu, Y. Liu, and X. Wu, "MiRTDL: A deep learning approach for miRNA target prediction," *IEEE Transactions on Computational Biology and Bioinformatics*, vol. PP, no. 99, 2015.

[37] Y. Chen, Y. Li, R. Narayan, A. Subramanian, and X. Xie, "Gene expression inference with deep learning," *Bioinformatics*, vol. 32, no. 12, pp. 1832–1839, 2016.

[38] R. Singh, J. Lanchantin, G. Robins, and Y. Qi, "DeepChrome: Deep-learning for predicting gene expression from histone modifications," *Bioinformatics*, vol. 32, no. 17, pp. i639–i648, 2016.

[39] B. Alipanahi, A. Delong, M. T. Weirauch, and B. J. Frey, "Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning," *Nature Biotechnology*, vol. 33, no. 8, pp. 831–838, 2015.

[40] J. Zhou and O. G. Troyanskaya, "Predicting effects of noncoding variants with deep learning-based sequence model," *Nature Methods*, vol. 12, no. 10, pp. 931–934, 2015.

[41] D. Quang and X. Xie, "DanQ: A hybrid convolutional and recurrent deep neural network for quantifying the function of DNA sequences," *Nucleic Acids Research*, vol. 44, no. 11, article no. e107, 2016.

[42] D. R. Kelley, J. Snoek, and J. L. Rinn, "Basset: Learning the regulatory code of the accessible genome with deep convolutional neural networks," *Genome Research*, vol. 26, no. 7, pp. 990–999, 2016.

[43] G. Ditzler, R. Polikar, and G. Rosen, "Multi-Layer and Recursive Neural Networks for Metagenomic Classification," *IEEE Transactions on NanoBioscience*, vol. 14, no. 6, pp. 608–616, 2015.

[44] D. C. Richter, F. Ott, A. F. Auch, R. Schmid, and D. H. Huson, "MetaSim—a sequencing simulator for genomics and metagenomics," *PLoS ONE*, vol. 3, no. 10, Article ID e3373, 2008.

[45] K.-C. Chou, "Some remarks on protein attribute prediction and pseudo amino acid composition," *Journal of Theoretical Biology*, vol. 273, pp. 236–247, 2011.

[46] S.-W. Zhang, Y.-F. Liu, Y. Yu, T.-H. Zhang, and X.-N. Fan, "MSLoc-DT: A new method for predicting the protein subcellular location of multispecies based on decision templates," *Analytical Biochemistry*, vol. 449, no. 1, pp. 164–171, 2014.

[47] M. Pohl, G. Theien, and S. Schuster, "GC content dependency of open reading frame prediction via stop codon frequencies," *Gene*, vol. 511, no. 2, pp. 441–446, 2012.

[48] J. Besemer and M. Borodovsky, "Heuristic approach to deriving models for gene finding," *Nucleic Acids Research*, vol. 27, no. 19, pp. 3911–3920, 1999.

[49] C.-T. Zhang and R. Zhang, "Analysis of distribution of bases in the coding sequences by a digrammatic technique," *Nucleic Acids Research*, vol. 19, no. 22, pp. 6313–6317, 1991.

[50] F.-B. Guo, H.-Y. Ou, and C.-T. Zhang, "ZCURVE: A new system for recognizing protein-coding genes in bacterial and archaeal genomes," *Nucleic Acids Research*, vol. 31, no. 6, pp. 1780–1789, 2003.

[51] S. Chen, C.-Y. Zhang, and K. Song, "Recognizing short coding sequences of prokaryotic genome using a novel iteratively

adaptive sparse partial least squares algorithm," *Biology Direct*, vol. 8, no. 1, article no. 23, 2013.

[52] L. Deng, D. Yu, and J. Platt, "Scalable stacking and learning for building deep architectures," in *Proceedings of the 2012 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '12)*, pp. 2133–2136, Kyoto, Japan, March 2012.

[53] L. Deng, X. He, and J. Gao, "Deep stacking networks for information retrieval," in *Proceedings of the 2013 38th IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2013*, pp. 3153–3157, can, May 2013.

[54] D. Yu and L. Deng, "Accelerated parallelizable neural network learning algorithm for speech recognition," in *Proceedings of the 12th Annual Conference of the International Speech Communication Association, INTERSPEECH 2011*, pp. 2281–2284, ita, August 2011.

[55] S.-W. Zhang and Z.-G. Wei, "Some remarks on prediction of protein-protein interaction with machine learning," *Medicinal Chemistry*, vol. 11, no. 3, pp. 254–264, 2015.

[56] S.-W. Zhang, L.-Y. Hao, and T.-H. Zhang, "Prediction of protein-protein interaction with pairwise kernel support vector machine," *International Journal of Molecular Sciences*, vol. 15, no. 2, pp. 3220–3233, 2014.

[57] S.-W. Zhang, Y.-L. Zhang, H.-F. Yang, C.-H. Zhao, and Q. Pan, "Using the concept of Chou's pseudo amino acid composition to predict protein subcellular localization: an approach by incorporating evolutionary information and von Neumann entropies," *Amino Acids*, vol. 34, no. 4, pp. 565–572, 2008.

[58] W.-R. Qiu, B.-Q. Sun, X. Xiao, Z.-C. Xu, and K.-C. Chou, "iPTM-mLys: Identifying multiple lysine PTM sites and their different types," *Bioinformatics*, vol. 32, no. 20, pp. 3116–3123, 2016.

[59] J. Jia, Z. Liu, X. Xiao, B. Liu, and K.-C. Chou, "iSuc-PseOpt: identifying lysine succinylation sites in proteins by incorporating sequence-coupling effects into pseudo components and optimizing imbalanced training dataset," *Analytical Biochemistry*, vol. 497, pp. 48–56, 2016.

BioMed
Research International

Stem Cells
International

International Journal of
Peptides

Advances in
Virology

International Journal of
Genomics

International Journal of
Zoology

Journal of
Nucleic Acids

Journal of
Signal Transduction

Hindawi

Submit your manuscripts at
https://www.hindawi.com

The Scientific
World Journal

Genetics
Research International

Anatomy
Research International

International Journal of
Microbiology

Biochemistry
Research International

Advances in
Bioinformatics

Archaea

Enzyme
Research

International Journal of
Evolutionary Biology

Molecular Biology
International

Journal of
Marine Biology