SCIENTIFIC REPORTS

OPEN

Recurrent Neural Network for Predicting Transcription Factor Binding Sites

Received: 27 April 2018
Accepted: 25 September 2018
Published online: 15 October 2018

Zhen Shen, Wenzheng Bao & De-Shuang Huang

It is well known that DNA sequence contains a certain amount of transcription factors (TF) binding sites, and only part of them are identified through biological experiments. However, these experiments are expensive and time-consuming. To overcome these problems, some computational methods, based on k-mer features or convolutional neural networks, have been proposed to identify TF binding sites from DNA sequences. Although these methods have good performance, the context information that relates to TF binding sites is still lacking. Research indicates that standard recurrent neural networks (RNN) and its variants have better performance in time-series data compared with other models. In this study, we propose a model, named KEGRU, to identify TF binding sites by combining Bidirectional Gated Recurrent Unit (GRU) network with k-mer embedding. Firstly, DNA sequences are divided into k-mer sequences with a specified length and stride window. And then, we treat each k-mer as a word and pre-trained word representation model though word2vec algorithm. Thirdly, we construct a deep bidirectional GRU model for feature learning and classification. Experimental results have shown that our method has better performance compared with some state-of-the-art methods. Additional experiments about embedding strategy show that k-mer embedding will be helpful to enhance model performance. The robustness of KEGRU is proved by experiments with different k-mer length, stride window and embedding vector dimension.

Transcription factors (TFs) play a critical role in gene expression. It can control genetic information transmission from DNA to messenger RNA by binding to a specific area in DNA sequence[1–3]. The mutations of TF binding sites and its adjacent have great influence on gene expression, and then increase the risk of complex disease[4–8]. It is no doubt that detailed analysis of the TF binding is significant to the further study of gene expression. Many related researches, like digging TF binding sites and exploring the effect of loci mutation on TF binding, have been done in the lab. However, biological experiments for TF binding are expensive and time-consuming. With the development of high-throughput sequencing technology, more and more biological datasets have been proposed[9–11]. Therefore, the principal mission of researchers is to develop a computational model to infer the underlying binding rules and identify TF binding sites without prior information from these datasets.

At the beginning of this study, many computational models, which were used to describe TF binding preference, are proposed based on position weight matrices (PWMs) or motifs[12–18]. These models don't consider the effect of other sequence features on TF binding, such as low-affinity binding sites, flanking DNA, sequence GC bias, and so on[19–24]. Moreover, ChIP-seq data not only contain TF binding information, but also have the aforementioned sequence features. Therefore, many ChIP-seq-based computational models have been proposed and have better performance than previous models[14,20,25–32]. For example, unlike original kmer-SVM, Mahmoud *et al.*[33] extracted gapped k-mer feature from ChIP-seq and trained a SVM classifier, which would be used to predict functional genomic regulatory elements and tissue-specific enhancers. This change has greatly improved the prediction performance of original model. In addition, some computational models have made a comprehensive application of ChIP-seq and DNase-seq[27,34–41], which ensure the prediction accuracy of these models. What's more, some researchers also proposed related models to identify functional modules from a whole-genome sequence[42,43], like promoter[44], enhancer[45,46], and recombination spots[47]. For instance, a two-layer predictor[45], named 'iEnhancer-2L', was developed to identify enhancers and their strength by pseudo k-tuple nucleotide

Institute of Machine Learning and Systems Biology, School of Electronics and Information Engineering, Tongji University, Shanghai, 201804, P. R. China. Correspondence and requests for materials should be addressed to D.-S.H. (email: dshuang@tongji.edu.cn)

composition. Liu *et al.*[47] proposed an ensemble learning approach to identify recombination spots by using multi-modal feature obtained from genome sequence.

Due to the reason that deep learning[48,49] can learn feature information directly from huge amounts of data, it has developed very rapidly in the past few years and has been widely used in computer vision[50], image analysis[51–55], speech recognition[56], natural language processing (NLP)[57], and others[58,59]. Recently, researchers used deep learning to extract gene regulation information from DNA sequences[38,60–63]. For example, Babak *et al.*[28] proposed a model based on deep convolutional neural networks (CNN), named DeepBind, to predict the sequence specificities of DNA- and RNA- binding protein. This model has achieved better performance than other existing methods. Zeng *et al.*[64] made a systematic exploration of CNN application in DNA-protein binding. The characteristic of this study is that they used a flexible cloud-based framework to achieve the rapid exploration of alternative neural network architectures. These CNN-based models have achieved better performance, but we also note that CNN only focus on the current state and cannot capture the influence of previous state and future state on current state. To address this problem, Quang *et al.*[65] proposed a hybrid convolutional and recurrent neural network framework for predicting the function of short DNA sequence. Since recurrent neural networks (RNN)[66] can effectively extract feature information from time-series data, it has been widely used in the process of sequence data, like text classification, video description. In this paper, we use Bidirectional Gated Recurrent Units (GRU)[67] network to extract feature information from DNA sequence, and then predict TF binding sites by using the feature information.

Neural networks cannot be used for text analysis directly unless we transform text data into the specific format[68–70]. Therefore, word embedding was proposed to solve the defect of one-hot, which can't reflect the distribution characteristic of text data. For word embedding, word corpus, specific language model and feature learning should be finished at first, and then words or phrases from the corpus are mapped to vectors of real numbers[71–74]. In this paper, k-mer is considered as a word in the sentence, so DNA sequences are divided into a k-mer series with a specified length and stride window. These k-mer datasets would be sent into Bidirectional GRU network for feature learning and classification.

Here, we proposed KEGRU, a novel computational method for predicting DNA-protein binding sties. In KEGRU, a DNA sequence is divided into a k-mer sequence with a specified length and stride window at first. And then, the k-mer sequence is mapped into D-dimensional vector space by word2vec. Thirdly, we use BiGRU to learn features from k-mer sequences and give prediction result. To evaluate the performance of our model, we chose four cell line TF binding datasets HESC, A549, HUVEC and MCF7 from the Encyclopedia of DNA Elements (ENCODE)[75] project. Experiment results show that our model has better performance than other competing methods on the task of predicting TF binding sites. We prove that k-mer embedding is helpful for the transformation of k-mer sequence. We verify the influence of different k-mer length, slide window and vector length. We also compare the performance of KEGRU with three baseline methods: gkmSVM, DeepBind and CNN_ZH[64]. We hope that our method could contribute to the study of DNA sequence modeling and DNA regulatory mechanisms.

Results

In this section, we used the Keras platform to implement the KEGRU model. A series of experiments were performed to evaluate the performance of our model. For simplicity, we call a CNN-based model, which was proposed by Zeng *et al.*[64], as CNN_ZH in this paper. We compared our model with gkmSVM, DeepBind, and CNN_ZH. We evaluated the efficacy of k-mer embedding strategy. We also evaluated the robustness of our model with different k-mer length, stride window, and embedding vector dimension.

AUC (the area under the receiver operating characteristic curve) was used in this paper to evaluate the performance of our model. As a common evaluation metric, AUC is widely used in machine learning and motif discovery. AUC represents a probability, which is generated by a classifier that will rank a randomly chosen positive instance higher than a randomly chosen negative one. In addition, we also used average precision score (APS) to evaluate the performance of our model. APS summarizes a precision-recall curve as the weighted mean of precisions achieved at each threshold, with the increase in recall from the previous threshold used as the weight.

Experiment setup. To evaluate the performance of our model, we used 125 TF binding sites ChIP-seq experiments from the ENCODE project, including A549, MCF-7, H1-HESC and HUVEC. For each cell type, the centered 101 bps were chosen as positive samples from each record in peak file. To meet model test requirement, equal numbers of negative samples were generated by matching the size, GC-content and repeat fraction of the positive sample. Each dataset was randomly divided into three groups: training, validation and test sets.

For the training of k-mer embedding model, a k-mer corpus was generated by setting k to 5, and the stride s to 2. We used the python implementation of the word2vec model in Gensim package to obtain the k-mer embedding vectors. All parameters in word2vec were left at their default values.

Hyper-parameter. The hyper-parameter setting in our methods consists of two groups: model-related and data-related. For model-related, it contains 12 parameter settings, which are generated by the combination of different optimizer and GRU number. Details of the model-related hyper-parameter setting are summarized in Table 1. For each ChIP-seq dataset, we execute all hyper-parameters, and record the performance of different parameter settings and get the best parameter setting by comparing the performance of all datasets. The hyper-parameter of three baseline models, gkmSVM, DeepBind, and CNN_ZH, remain unchanged.

The whole process is made up of two steps: model training and data statistical analysis. In model training, for each ChIP-seq dataset, we used all hyper-parameters to train KEGRU on the training set with a mini-batch size of 200 and tested it on the validation set. After training, all trained model should be tested on the corresponding test set and record test results. In data statistical analysis, all test results are classified by cell type, hyper-parameter

Hyper-parameter	Optional
GRU units number	50,80,100
Optimizer	SGD, Adam, Adagrad, RMSprop

Table 1. Hyper-parameter settings.

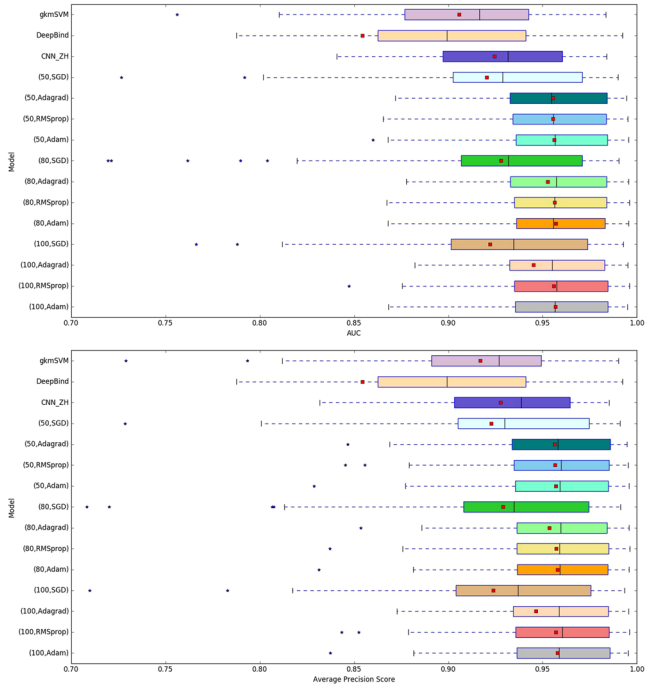

Figure 1. The distribution of AUCs and APSs across 125 experiments in the task of DNA-protein binding prediction.

setting and test indicators. The reason for this is that TF may have different binding properties on different cell type and a hyper-parameter setting may have better performance on one cell type and opposite case on other cell types. We would have the best hyper-parameter on specific cell type by comparing the test results.

Although the word embedding has been widely used in NLP, the effect of different embedding strategy on TF binding site prediction is unknown. In addition, we still don't know the influence of k-mer length, stride window and embedding vector dimension on model performance. Therefore, these settings were also regarded as hyper-parameter and would be discussed later in following section.

Performance comparison with existing methods. To demonstrate the performance of our model KEGRU and compare it with three baseline models gkmSVM, DeepBind and CNN_ZH, we performed a series of experiments with the different hyper-parameter setting. Figure 1 is the concentrated display of the AUC and APS distribution of four models on four cell types with various hyper-parameter settings, highlighting the excellent performance of KEGRU. Table 2 displays the average APS of KEGRU compared with three baseline models. Table 3 displays the average AUCs of KEGRU compared with three baseline models. Given the above information, we chose the best hyper-parameter setting and used scatter plots to display the performance gap of KRGRU and three baseline models. Figure 2(a) shows that KEGRU has higher performance than gkmSVM on four cell line. Figure 2(b,c) show the performance comparison of KEGRU and other two baseline model, respectively. As shown in Fig. 2, our model KEGRU always performs better than three baseline models. In general, with the help of BiGRU and k-mer embedding, our model KEGRU has higher AUC scores than other three baseline models, which means that our model would be helpful for the motif discovery task.

Evaluate the effect of k-mer embedding. Although word embedding has proven its efficacy in NLP, it is rarely used in human genomic research[76]. The role of the embedding layer in our model is to map the k-mer index to k-mer vectors obtained by the pre-trained k-mer model. Previous experiments have shown that when using k-mer embedding and BiGRU, KEGRU has better performance than other three baseline models. But, the question is that we still don't know how much the effect of the embedding strategy on model performance is. To address this question, we designed three embedding ways: no-init, init-no-train, and init-train. No-init means that the weights of the embedding layer are initialized with uniform distribution and the neural network can adjust the weights during model training. Init-no-train means that the weights of the embedding layer are initialized by the pre-trained k-mer vectors and will not be updated during model training. Init-train means that

Cell Line				
hyper-parm	HUVEC	MCF7	A549	H1-HESC
gkmSVM	0.9226	0.9016	0.9134	0.9224
DeepBind	0.8601	0.8857	0.8775	0.8291
CNN_ZH	0.9368	0.9173	0.9351	0.9241
50, SGD	0.9423	0.9454	0.9328	0.9046
50, Adagrad	0.9611	0.9648	0.9582	0.9516
50, RMSprop	0.9612	0.9656	0.9601	0.9502
50, Adam	0.9613	0.9656	0.9599	0.9517
80, SGD	0.9429	0.9516	0.9357	0.9144
80, Adagrad	0.9616	0.9657	0.9466	0.9521
80, RMSprop	0.9618	**0.9659**	0.9600	0.9519
80, Adam	0.9618	0.9653	**0.9608**	**0.9528**
100, SGD	0.9424	0.9530	0.9342	0.9041
100, Adagrad	**0.9620**	0.9350	0.9455	0.9469
100, RMSprop	0.9607	0.9655	0.9607	0.9510
100, Adam	0.9619	0.9657	0.9606	0.9525

Table 2. Average APS scores across 125 ChIP-Seq datasets.

Cell Line				
hyper-parm	HUVEC	MCF7	A549	H1-HESC
gkmSVM	0.9119	0.8830	0.9031	0.9129
DeepBind	0.8613	0.8861	0.8784	0.8296
CNN_ZH	0.9335	0.9147	0.9316	0.9210
50, SGD	0.9393	0.9426	0.9285	0.9039
50, Adagrad	0.9603	0.9637	0.9565	0.9509
50, RMSprop	0.9604	0.9646	0.9583	0.9497
50, Adam	0.9605	0.9644	0.9580	0.9513
80, SGD	0.9409	0.9494	0.9320	0.9153
80, Adagrad	0.9610	0.9647	0.9453	0.9514
80, RMSprop	0.9607	**0.9649**	0.9584	0.9512
80, Adam	0.9608	0.9643	**0.9593**	**0.9524**
100, SGD	0.9392	0.9505	0.9301	0.9042
100, Adagrad	**0.9612**	0.9332	0.9431	0.9461
100, RMSprop	0.9599	0.9645	0.9591	0.9502
100, Adam	0.9611	0.9646	0.9588	0.9520

Table 3. Average AUC scores across 125 ChIP-Seq datasets.

the weights of the embedding layer are initialized by the pre-trained k-mer vectors and will be fine-tuned during training. This strategy is adopted in our model KEGRU.

Figure 3 displays the average APS of the above three embedding strategies on four datasets. The contrast examination shows that when using pre-trained k-mer vectors, the impact of fine-tuning is minor to the performance of our model. The k-mer embedding strategy has got the desired improvements in the model performance. In conclusion, the pre-trained k-mer vectors could reflect the distribution characteristics of k-mer in DNA sequence well, and help improve model performance.

Sensitivity analysis. In this section, we used the HUVEC dataset to analyze the effect of k-mer length k, stride window s and embedding dimension d. With the increase of k, the capacity of k-mer vocabulary would be explosive growth. Besides, small k can't reflect the characteristics of TF binding. In this paper, we reconstructed the k-mer corpus with different k (from 4 to 6) and obtained the k-mer embedding vectors by re-training k-mer model. Figure 4(a) shows that our model has got close prediction performance with different k.

The number of k-mer N is determined together by DNA sequence length L, k-mer length k and stride window s.

$$N = \lfloor (L - k)/s \rfloor + 1 \tag{1}$$

Figure 4(b) shows that when using four different stride window $s = 2, 3, 4$ and 5, the performance of our model is decreasing. As shown in Equation (1), the size of the k-mer corpus will be decreased by a larger s, and can lead to the lack of useful information. This may have negative impact on the embedding representation. We don't specify the stride window as 1 because it may sharply increase the size of the k-mer corpus and make a k-mer largely

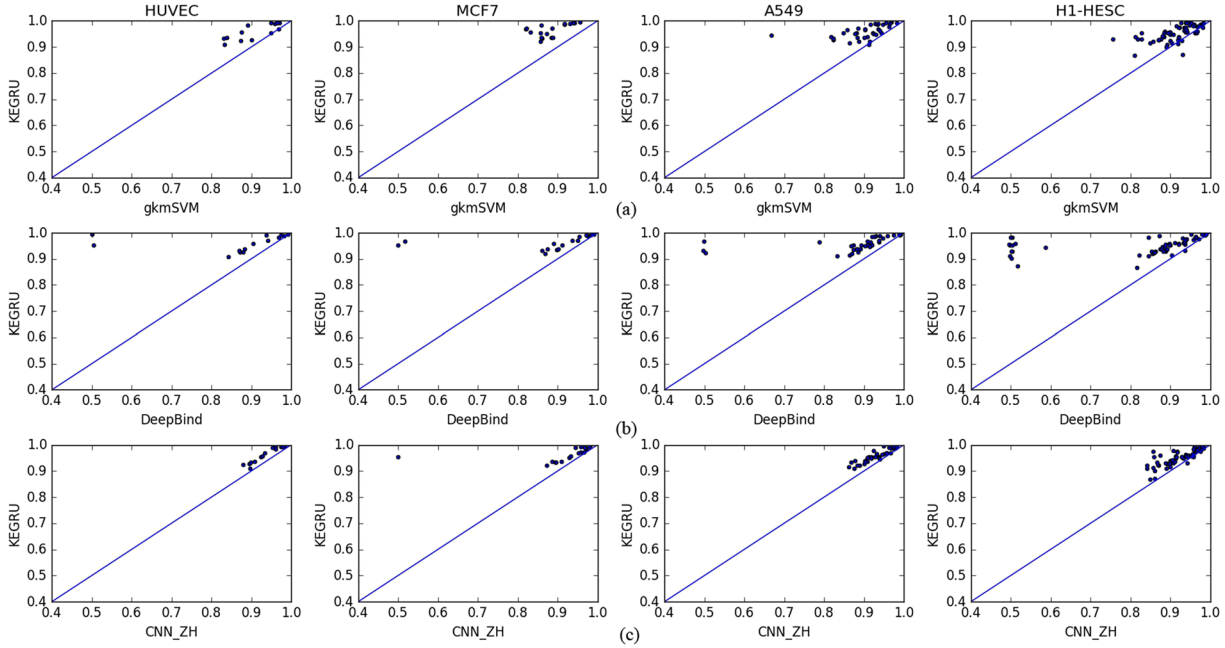

Figure 2. Performance comparison between KEGRU and three baseline models on four cell lines.

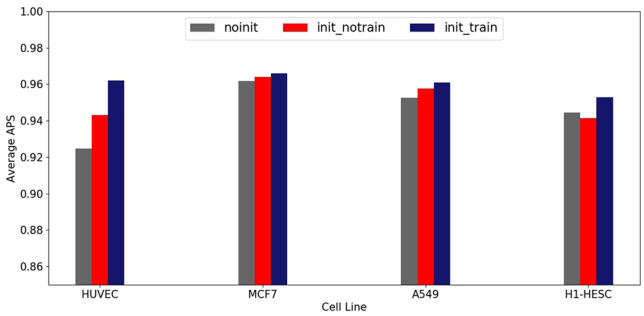

Figure 3. Model performance for different embedding strategies.

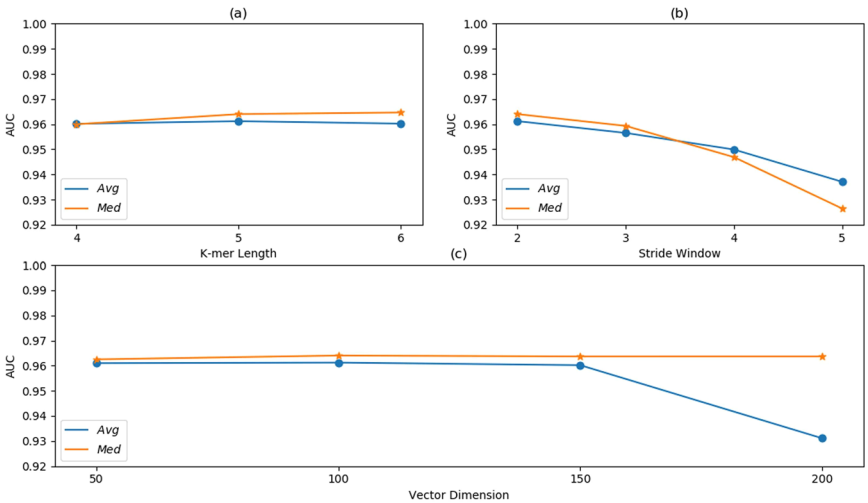

Figure 4. Sensitivity analysis of different k-mer length, stride window and embedding vector dimension, performed on the HUVEC dataset. The average and median AUC scores on HUVEC are reported.

overlap with its neighbor. This also has great influence on the embedding representation. In general, to make the embedding layer work well, a proper stride window s 2 is recommended here.

In addition, we also analyzed the effect of different embedding dimension d, including 50, 100, 150 and 200. The model complexity will be increased by more weight parameters, which is caused by a larger d and needs to be learned in embedding layer. As shown in Fig. 4(c), although changes in embedding dimension d may lead to overfitting, the prediction accuracy of our model remains fairly constant throughout the experiment.

Discussion

In this paper, we propose a bidirectional gated recurrent unit neural network with k-mer embedding to identify TF binding sites from DNA sequence. The characteristic of our model is summarized as follows. Firstly, word embedding has been introduced in our model and applied to k-mer sequence representation using the unsupervised learning algorithm word2vec. To avoid dimension curse caused by one-hot encoding, the k-mer embedding vectors are used for feature representation. This measurement is beneficial to following feature learning and classification task. Secondly, our model KEGRU is suitable to processing variable-length input sequences. The shortcoming of CNNs is that the length of input sequence must be fixed. The application of BiGRU network not only improves the compatibility of variable length input sequence, but also is able to capture complex context information from the k-mer sequence. In addition, we prove that our model has better prediction performance than other baseline methods. We also prove that k-mer embedding is an effective method to improve the performance of our model. What's more, we show the robustness of our model through hyper-parameter experiments. We also show the role of NLP and RNN in the DNA sequence analysis.

There are still some works to be done in the future to improve the performance of our model. First, in the double-stranded DNA sequences, domain-specific modifications may appear identically on one strand or its reverse complement. Therefore, GRU with reverse complement mechanism is helpful to feature learning and classification task. Second, the attention mechanism is also an excellent choice. Attention mechanism has been used in document classification and sentiment classification and achieved better performance. The basic idea of attention mechanism is that it can focus on the key parts of the whole sequence. This characteristic may be used in our model to help explore the important TF binding sites on DNA sequences. In NLP, there are some existing methods to embed sentence or documents into a vector directly by sentence2vec and paragraph2vec. Therefore, we can design a new embedding algorithm for the representation of the variable-length k-mer sequence. Finally, we hope that our method would contribute to the study of gene regulation mechanism.

Methods

In this section, we describe the basic structure of KEGRU at first. Then, we discuss the detail information of word embedding in our model, which is used to represent a k-mer as a low-dimensional vector. At last, we used Bidirectional GRUs to capture long range dependencies and form fixed-length feature representation of arbitrary-length DNA sequences.

Model architecture. Given the k-mer length k and stride widow s, a DNA sequence with L_D base pairs will be split into a k-mer sequence KS with length $L_k = \lfloor (L_D - k)/s \rfloor + 1$. Each k-mer in KS is indexed by positive integers in $K = [1, 2, \cdots, N]$. Then, we will explore an appropriate approach to learn the co-occurrence information of k-mer in KS, which will help map k-mer sequence into a vector space V.

In KEGRU, each DNA sequence is given a binary label, which represents whether the short DNA sequence is a TF binding region or not. Suppose that we have M labeled instances $\{x_i, y_i\}_{i=1}^{N}$, where $x_i \in K^{L_k}$, $y_i \in (0, 1)$. Our task is to build the prediction model that is used to predict the label for each instance. Figures 5 shows the basic structure of KEGRU. We use formula (2) to represent the entire flow:

$$Y = f_{pred}(f_{GRU}(f_{embed}(x))) \tag{2}$$

where x denotes a k-mer sequence.

During k-mer embedding, a k-mer will be mapped into a vector by learning the co-occurrence statistics of each k-mer in k-mer sequence. We use a Bidirectional GRU network to capture the long-term dependencies, and then generate a fixed-length feature vector. In the prediction stage, we will obtain a prediction by performing a logistic regression on the feature representations. Given k-mer sequence x_i and model parameters Φ, the conditional likelihood of predicting label y_i is computed by[76]:

$$\log p(y_i|x_i, \Phi) = y_i \log \delta(\beta^T h_i) + (1 - y_i \log(1 - \delta(\beta^T h_i))) \tag{3}$$

where β denotes the prediction parameter. h_i represents the learned fixed-length feature. δ denotes the logistic sigmoid function. Our model is trained by minimizing the loss function:

$$\varphi = -\sum_{i=1}^{N} \log p(y_i|x_i, \Phi) \tag{4}$$

K-mer embedding with Word2vec. If we use one-hot[77] to encode each word in a large number of text data, the word vector may be a high-dimensional vector. We have to consume more and more computer resources to store and process these vectors. To address this problem, researchers proposed the concept of distributed representation[78–81], including word embedding. Briefly, the core idea behind word embedding is to model and analyze semantic similarities between words based on their distributional properties in large samples of document

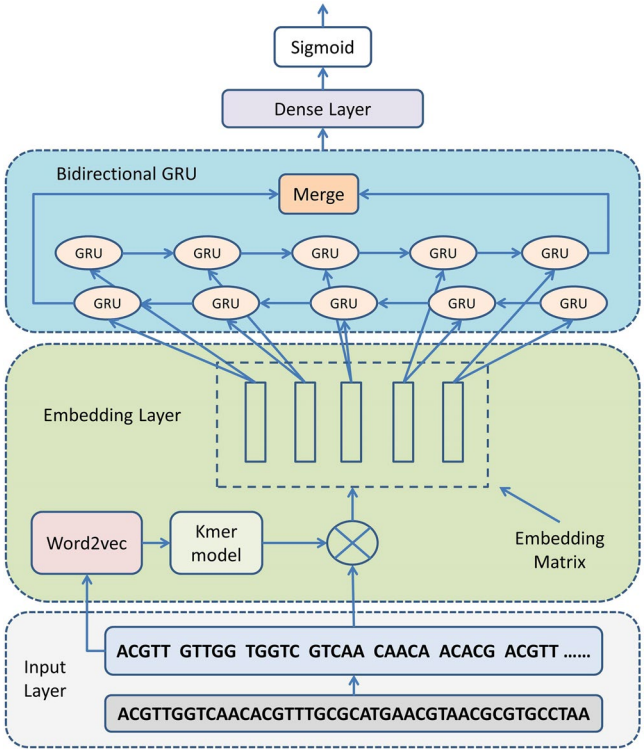

Figure 5. The basic architectural structure of our model KEGRU. (1) We first built the k-mer corpus, which consists of a number of k-mer sequence built by splitting DNA sequence. (2) Based on the k-mer corpus built at first step, we use the pre-trained model word2vec to learn the k-mer embedding vectors. All k-mer vectors are stacked into the embedding matrix that will be used to initialize the embedding layer. (3) We use bidirectional GRU network to solve long-range dependencies problem and to learn feature information from input k-mer sequence. (4) The prediction results were generated by the dense layer and the sigmoid layer, and then we use a loss function to compare the prediction results with the true target labels.

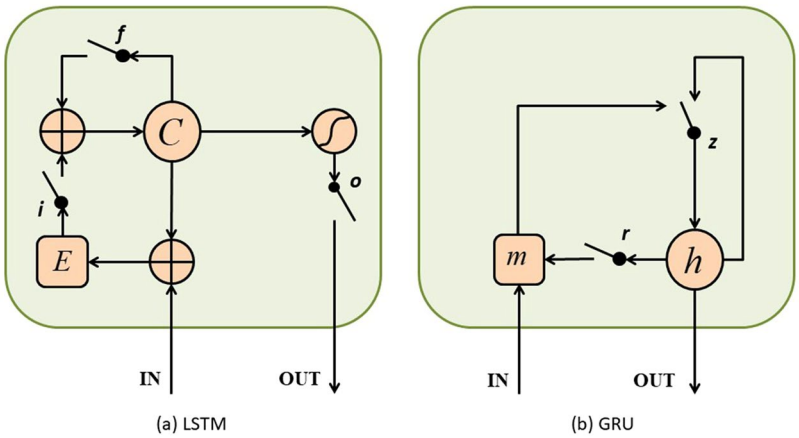

Figure 6. Structural comparison between (**a**) LSTM and (**b**) GRU[67]. (**a**) *i*, *f* and *o* denote the input, forget and output gates, respectively. *C* and *E* denote the current memory cell state and the new memory cell state. (**b**) *r* and *z* represent the reset and update gates. *h* and *m* are the current unit state and the candidate unit state.

data. After word embedding, words from the corpus would be mapped to vectors of real numbers, which can be processed by a neural network model.

In practical use, all the word vectors will be deposited into a matrix $WE \in R^{d \times N}$, where N denotes the size of the corpus and d denotes the word vector dimension. We call this matrix as the embedding layer or the lookup table layer. The embedding layer can be initialized through a pre-trained algorithm, and some algorithms have been proposed based on neural networks[70], dimensionality reduction on the word co-occurrence matrix[72],

probabilistic models[82], and explicit representation in terms of the context in which words appear[83]. For example, word2vec is a set of related models based on continuous bag-of-words and skip-gram. These models are sample neural networks that are trained to generate the context information of the word. As an unsupervised learning algorithm, Glove was used to learn word feature information from a corpus. Word vector is obtained through global word-word co-occurrence statistics.

In our model, each k-mer in k-mer sequence is considered as a word in the sentence. Therefore, we can use word embedding to represent a k-mer sequence at the word level. Given a k-mer sequence KS consisting of N k-mers, it can be represented as $KS = \{k_1, k_2, k_3, \cdots, k_N\}$. We first train a k-mer model KM through word2vec. Then, the vector of k-mer k_t is obtained by KM:

$$kv_t = KM(k_t) \tag{5}$$

Then, the k-mer sequence KS can be represented as $KS_e = \{kv_1, kv_2, kv_3, \cdots, kv_N\}$.

Bidirectional GRU. In order to deal with several shortcomings about the standard RNN model, a list of efforts, including Long short-term memory (LSTM)[84,85] and other similar approaches, have been proposed in this field. The GRU was proposed by Cho *et al.*[86]. Figure 6 shows the internal structure of LSTM and GRU. In a gated recurrent neural network, each unit can control the flow of information through resetting gate and updating gate, and all memory contents are fully exposed at each time step. Besides, the output of GRU is to achieve a balance between the previous memory state and the new candidate memory state.

The update gate z_t is computed by[67]

$$z_t = sigmoid(W_z x_t + U_z h_{t-1} + b_z) \tag{6}$$

where x_t is the input vector of the GRU. h_{t-1} is the previous output of the GRU. W_z, U_z and b_z are forward matrices, recurrent matrices and biases for update gate, respectively.

Similarly to the update gate, the reset gate is computed by[67]

$$r_t = sigmoid(W_r x_t + U_r h_{t-1} + b_r) \tag{7}$$

where the parameters are as above.

And then, the candidate memory state m_t is computed by[67]

$$m_t = \tanh(W_h x_t + U_h(r_t * h_{t-1}) + b_h) \tag{8}$$

where σ_h is the hyperbolic tangent function. * is an element-wise multiplication.

Finally, the memory state h_t of the GRU is computed by[67]

$$h_t = (1 - z_t)h_{t-1} + z_t m_t \tag{9}$$

To make our model have a flexible input data format and can reach future input information from the current state, we used Bidirectional RNN (BiRNN). The basic idea of BiRNN is that all neurons in regular RNN are split into forward layer and backward layer, which represent the positive time direction and negative time direction, respectively. By using this structure, it is easy to capture the effect of input information from the past and future on current state. The output of BiRNN is computed by merging forward layer out and backward layer out with specific mode, like concatenate, sum, average and multiplication.

In our model, standard RNN unit is replaced by GRU. The output of BiGRU is calculated by

$$output = merge(fout, bout) \tag{10}$$

where *fout* is the output of forward layer. *bout* is the output of backward layer.

Data Availability Statement

The datasets generated during and analyzed during the current study are available from the corresponding author on reasonable request.

References

1. Latchman, D. S. Transcription factors: an overview. *The international journal of biochemistry & cell biology* **29**, 1305–1312 (1997).
2. Karin, M. Too many transcription factors: positive and negative interactions. *The New Biologist* **2**, 126–131 (1990).
3. Pan, Y., Tsai, C.-J., Ma, B. & Nussinov, R. Mechanisms of transcription factor selectivity. *Trends in Genetics* **26**, 75–83 (2010).
4. Mathelier, A., Shi, W. & Wasserman, W. W. Identification of altered cis-regulatory elements in human disease. *Trends in Genetics* **31**, 67–76 (2015).
5. Weinhold, N., Jacobsen, A., Schultz, N., Sander, C. & Lee, W. Genome-wide analysis of noncoding regulatory mutations in cancer. *Nature genetics* **46**, 1160–1165 (2014).
6. Friedensohn, S. & Sawarkar, R. Cis-regulatory variation: significance in biomedicine and evolution. *Cell and tissue research* **356**, 495–505 (2014).
7. Deplancke, B., Alpern, D. & Gardeux, V. The genetics of transcription factor DNA binding variation. *Cell* **166**, 538–554 (2016).
8. Yu, H.-J. & Huang, D.-S. Normalized feature vectors: a novel alignment-free sequence comparison method based on the numbers of adjacent amino acids. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)* **10**, 457–467 (2013).
9. Johnson, D. S., Mortazavi, A., Myers, R. M. & Wold, B. Genome-wide mapping of *in vivo* protein-DNA interactions. *Science* **316**, 1497–1502 (2007).
10. Consortium, E. P. An integrated encyclopedia of DNA elements in the human genome. *Nature* **489**, 57–74 (2012).
11. Deng, S.-P. & Huang, D.-S. SFAPS: an R package for structure/function analysis of protein sequences based on informational spectrum method. *Methods* **69**, 207–212 (2014).

12. Warner, J. B. *et al*. Systematic identification of mammalian regulatory motifs' target genes and functions. *Nature methods* **5**, 347–353 (2008).
13. Badis, G. *et al*. A library of yeast transcription factor motifs reveals a widespread function for Rsc3 in targeting nucleosome exclusion at promoters. *Molecular cell* **32**, 878–887 (2008).
14. Weirauch, M. T. *et al*. Evaluation of methods for modeling transcription factor sequence specificity. *Nature biotechnology* **31**, 126–134 (2013).
15. Wang, J. *et al*. Factorbook. org: a Wiki-based database for transcription factor-binding data generated by the ENCODE consortium. *Nucleic acids research* **41**, D171–D176 (2012).
16. Mathelier, A. *et al*. JASPAR 2014: an extensively expanded and updated open-access database of transcription factor binding profiles. *Nucleic acids research* **42**, D142–D147 (2013).
17. Deng, S.-P., Zhu, L. & Huang, D.-S. Predicting hub genes associated with cervical cancer through gene co-expression networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)* **13**, 27–35 (2016).
18. Bao, W., Huang, Z., Yuan, C.-A. & Huang, D.-S. Pupylation sites prediction with ensemble classification model. *International Journal of Data Mining and Bioinformatics* **18**, 91–104 (2017).
19. von Hippel, P. H. Increased subtlety of transcription factor binding increases complexity of genome regulation. *Proceedings of the National Academy of Sciences* **111**, 17344–17345 (2014).
20. Siggers, T. & Gordân, R. Protein–DNA binding: complexities and multi-protein codes. *Nucleic acids research* **42**, 2099–2111 (2013).
21. Afek, A., Schipper, J. L., Horton, J., Gordân, R. & Lukatsky, D. B. Protein− DNA binding in the absence of specific base-pair recognition. *Proceedings of the National Academy of Sciences* **111**, 17140–17145 (2014).
22. Hoff, B. & Kück, U. Use of bimolecular fluorescence complementation to demonstrate transcription factor interaction in nuclei of living cells from the filamentous fungus Acremonium chrysogenum. *Current genetics* **47**, 132–138 (2005).
23. Ross, M. G. *et al*. Characterizing and measuring bias in sequence data. *Genome biology* **14**, R51 (2013).
24. Deng, S.-P., Zhu, L. & Huang, D.-S. Mining the bladder cancer-associated genes by an integrated strategy for the construction and analysis of differential co-expression networks. *BMC genomics* **16**, S4 (2015).
25. Ghandi, M. *et al*. gkmSVM: an R package for gapped-kmer SVM. *Bioinformatics* **32**, 2205–2207 (2016).
26. Lee, D., Karchin, R. & Beer, M. A. Discriminative prediction of mammalian enhancers from DNA sequence. *Genome research* **21**, 2167–2180 (2011).
27. Zeng, H., Hashimoto, T., Kang, D. D. & Gifford, D. K. GERV: a statistical method for generative evaluation of regulatory variants for transcription factor binding. *Bioinformatics* **32**, 490–496 (2015).
28. Alipanahi, B., Delong, A., Weirauch, M. T. & Frey, B. J. Predicting the sequence specificities of DNA-and RNA-binding proteins by deep learning. *Nature biotechnology* **33**, 831–838 (2015).
29. Xie, D. *et al*. Dynamic trans-acting factor colocalization in human cells. *Cell* **155**, 713–724 (2013).
30. Zheng, C.-H., Huang, D.-S., Zhang, L. & Kong, X.-Z. Tumor clustering using nonnegative matrix factorization with gene selection. *IEEE Transactions on Information Technology in Biomedicine* **13**, 599–607 (2009).
31. Zhu, L., Deng, S.-P. & Huang, D.-S. A two-stage geometric method for pruning unreliable links in protein-protein networks. *IEEE transactions on nanobioscience* **14**, 528–534 (2015).
32. Huang, D.-S. Radial basis probabilistic neural networks: Model and application. *International Journal of Pattern Recognition and Artificial Intelligence* **13**, 1083–1101 (1999).
33. Ghandi, M., Lee, D., Mohammad-Noori, M. & Beer, M. A. Enhanced regulatory sequence prediction using gapped k-mer features. *PLoS computational biology* **10**, e1003711 (2014).
34. Pique-Regi, R. *et al*. Accurate inference of transcription factor binding from DNA sequence and chromatin accessibility data. *Genome research* **21**, 447–455 (2011).
35. Sherwood, R. I. *et al*. Discovery of directional and nondirectional pioneer transcription factors by modeling DNase profile magnitude and shape. *Nature biotechnology* **32**, 171–178 (2014).
36. Wang, P. *et al*. Methylation-mediated silencing of the miR-124 genes facilitates pancreatic cancer progression and metastasis by targeting Rac1. *Oncogene* **33**, 514–524 (2014).
37. Lee, D. *et al*. A method to predict the impact of regulatory variants from DNA sequence. *Nature genetics* **47**, 955–961 (2015).
38. Qin, Q. & Feng, J. Imputation for transcription factor binding predictions based on deep learning. *PLoS computational biology* **13**, e1005403 (2017).
39. Zheng, C.-H., Zhang, L., Ng, V. T.-Y., Shiu, C. K. & Huang, D.-S. Molecular pattern discovery based on penalized matrix decomposition. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)* **8**, 1592–1603 (2011).
40. Huang, D.-S. & Du, J.-X. A constructive hybrid structure optimization methodology for radial basis probabilistic neural networks. *IEEE Transactions on Neural Networks* **19**, 2099–2115 (2008).
41. Bao, W., Jiang, Z. & Huang, D.-S. Novel human microbe-disease association prediction using network consistency projection. *BMC bioinformatics* **18**, 543 (2017).
42. Liu, B. BioSeq-Analysis: A platform for DNA, RNA and protein sequence analysis based on machine learning approaches. *Briefings in bioinformatics* (2017).
43. Liu, B. *et al*. Pse-in-One: a web server for generating various modes of pseudo components of DNA, RNA, and protein sequences. *Nucleic acids research* **43**, W65–W71 (2015).
44. Liu, B., Yang, F., Huang, D.-S. & Chou, K.-C. iPromoter-2L: a two-layer predictor for identifying promoters and their types by multi-window-based PseKNC. *Bioinformatics* **34**, 33–40 (2017).
45. Liu, B., Fang, L., Long, R., Lan, X. & Chou, K.-C. iEnhancer-2L: a two-layer predictor for identifying enhancers and their strength by pseudo k-tuple nucleotide composition. *Bioinformatics* **32**, 362–369 (2015).
46. Liu, B., Li, K., Huang, D. -S. & Chou, K. -C. iEnhancer-EL: Identifying enhancers and their strength with ensemble learning approach. *Bioinformatics* (2018).
47. Liu, B., Wang, S., Long, R. & Chou, K.-C. iRSpot-EL: identify recombination spots with an ensemble learning approach. *Bioinformatics* **33**, 35–41 (2016).
48. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436–444 (2015).
49. Schmidhuber, J. Deep learning in neural networks: An overview. *Neural networks* **61**, 85–117 (2015).
50. Hoermann, S., Bach, M. & Dietmayer, K. Dynamic Occupancy Grid Prediction for Urban Autonomous Driving: A Deep Learning Approach with Fully Automatic Labeling. *arXiv preprint arXiv:1705.08781* (2017).
51. Zhu, L., You, Z.-H., Huang, D.-S. & Wang, B. t-LSE: a novel robust geometric approach for modeling protein-protein interaction networks. *PloS one* **8**, e58368 (2013).
52. Shen, Z. *et al*. miRNA-Disease Association Prediction with Collaborative Matrix Factorization. *Complexity* **2017** (2017).
53. Bao, W., Wang, D. & Chen, Y. Classification of Protein Structure Classes on Flexible Neutral Tree. *IEEE/ACM transactions on computational biology and bioinformatics* (2016).
54. Litjens, G. *et al*. A survey on deep learning in medical image analysis. *Medical image analysis* **42**, 60–88 (2017).
55. Sallab, A. E., Abdou, M., Perot, E. & Yogamani, S. Deep reinforcement learning framework for autonomous driving. *Electronic Imaging* **2017**, 70–76 (2017).
56. Hinton, G. *et al*. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine* **29**, 82–97 (2012).

57. Collobert, R. *et al*. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* **12**, 2493–2537 (2011).
58. Huang, D.-S. & Zheng, C.-H. Independent component analysis-based penalized discriminant method for tumor classification using gene expression data. *Bioinformatics* **22**, 1855–1862 (2006).
59. Chuai, G. *et al*. DeepCRISPR: optimized CRISPR guide RNA design by deep learning. *Genome biology* **19**, 80 (2018).
60. Min, S., Lee, B. & Yoon, S. Deep learning in bioinformatics. *Briefings in bioinformatics* **18**, 851–869 (2017).
61. Gusmao, E. G., Allhoff, M., Zenke, M. & Costa, I. G. Analysis of computational footprinting methods for DNase sequencing experiments. *Nature methods* **13**, 303 (2016).
62. Shrikumar, A., Greenside, P. & Kundaje, A. Reverse-complement parameter sharing improves deep learning models for genomics. *bioRxiv*, 103663 (2017).
63. Zhu, L., Zhang, H.-B. & Huang, D.-S. Direct AUC optimization of regulatory motifs. *Bioinformatics* **33**, i243–i251 (2017).
64. Zeng, H., Edwards, M. D., Liu, G. & Gifford, D. K. Convolutional neural network architectures for predicting DNA–protein binding. *Bioinformatics* **32**, i121–i127 (2016).
65. Quang, D. & Xie, X. DanQ: a hybrid convolutional and recurrent deep neural network for quantifying the function of DNA sequences. *Nucleic acids research* **44**, e107–e107 (2016).
66. Bullinaria, J. A. Recurrent neural networks. *Neural Computation: Lecture* **12** (2013).
67. Chung, J., Gulcehre, C., Cho, K. & Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).
68. Kim, Y. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* (2014).
69. Huang, D. -S. *Systematic Theory of Neural Networks for Pattern Recognition (in Chinese)* (May 1996).
70. Mikolov, T., Sutskever, I., Chen, K., Corrado, G. & Dean, J. Distributed Representations of Words and Phrases and their Compositionality. *arXiv preprint arXiv:1310.4546* (2013).
71. Serban, I. V., Sordoni, A., Bengio, Y., Courville, A. & Pineau, J. Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models. *arXiv preprint arXiv:1507.04808* (2015).
72. Goldberg, Y. & Levy, O. word2vec Explained: deriving Mikolov *et al*.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722* (2014).
73. Asgari, E. & Mofrad, M. R. Continuous distributed representation of biological sequences for deep proteomics and genomics. *PloS one* **10**, e0141287 (2015).
74. Zhu, L., Guo, W.-L., Deng, S.-P. & Huang, D.-S. ChIP-PIT: enhancing the analysis of ChIP-Seq data using convex-relaxed pair-wise interaction tensor decomposition. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* **13**, 55–63 (2016).
75. Consortium, E. P. The ENCODE (ENCyclopedia of DNA elements) project. *Science* **306**, 636–640 (2004).
76. Min, X., Zeng, W., Chen, N., Chen, T. & Jiang, R. Chromatin accessibility prediction via convolutional long short-term memory networks with k-mer embedding. *Bioinformatics* **33**, i92–i101 (2017).
77. Harris, D. & Harris, S. *Digital design and computer architecture* (Morgan Kaufmann, 2010).
78. Bengio, Y., Ducharme, R., Vincent, P. & Jauvin, C. A neural probabilistic language model. *Journal of machine learning research* **3**, 1137–1155 (2003).
79. Huang, D.-S. *et al*. Prediction of protein-protein interactions based on protein-protein correlation using least squares regression. *Current Protein and Peptide Science* **15**, 553–560 (2014).
80. Huang, D.-S. & Jiang, W. A general CPL-AdS methodology for fixing dynamic parameters in dual environments. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* **42**, 1489–1500 (2012).
81. Paccanaro, A. & Hinton, G. E. Learning distributed representations of concepts using linear relational embedding. *IEEE Transactions on Knowledge and Data Engineering* **13**, 232–244 (2001).
82. Globerson, A., Chechik, G., Pereira, F. & Tishby, N. Euclidean embedding of co-occurrence data. *Journal of Machine Learning Research* **8**, 2265–2295 (2007).
83. Cho, K. *et al*. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).
84. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural computation* **9**, 1735–1780 (1997).
85. Gers, F. A., Schmidhuber, J. & Cummins, F. Learning to forget: Continual prediction with LSTM (1999).
86. Cho, K., Van Merriënboer, B., Bahdanau, D. & Bengio, Y. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* (2014).

Acknowledgements

This work is partly supported by National Natural Science Foundation of China (Grant nos 61520106006, 31571364, 61732012, 61532008, U1611265, 61672382, 61772357, 61472173, 61572447, and 61672203) and China Postdoctoral Science Foundation (Grant nos 2016M601646 and 2017M611619) and supported by "BAGUI Scholar" Program of Guangxi Province of China.

Author Contributions

D.S.H. supervised the project. Z.S. and W.Z.B. designed the experiments. Z.S. conducted the experiments. Z.S. and W.Z.B. analyzed the results. Z.S. wrote the manuscript. All authors reviewed the manuscript.

Additional Information

Competing Interests: The authors declare no competing interests.

Publisher's note: Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.