BMC Bioinformatics

RESEARCH

Open Access

MultiScale-CNN-4mCPred: a multi-scale CNN and adaptive embedding-based method for mouse genome DNA N4-methylcytosine prediction

Peijie Zheng[1], Guiyang Zhang[1], Yuewu Liu[2] and Guohua Huang[1*]

*Correspondence: guohuahhn@163.com

[1] School of Information Engineering, Shaoyang University, Shaoyang 42200, China
[2] College of Information and Intelligence, Hunan Agricultural University, Changsha 410128, China

Abstract

N4-methylcytosine (4mC) is an important epigenetic mechanism, which regulates many cellular processes such as cell differentiation and gene expression. The knowledge about the 4mC sites is a key foundation to exploring its roles. Due to the limitation of techniques, precise detection of 4mC is still a challenging task. In this paper, we presented a multi-scale convolution neural network (CNN) and adaptive embedding-based computational method for predicting 4mC sites in mouse genome, which was referred to as MultiScale-CNN-4mCPred. The MultiScale-CNN-4mCPred used adaptive embedding to encode nucleotides, and then utilized multi-scale CNNs as well as long short-term memory to extract more in-depth local properties and contextual semantics in the sequences. The MultiScale-CNN-4mCPred is an end-to-end learning method, which requires no sophisticated feature design. The MultiScale-CNN-4mCPred reached an accuracy of 81.66% in the 10-fold cross-validation, and an accuracy of 84.69% in the independent test, outperforming state-of-the-art methods. We implemented the proposed method into a user-friendly web application which is freely available at: http://www.biolscience.cn/MultiScale-CNN-4mCPred/.

Keywords: Deep learning, Convolutional neural network, Long short-term memory, Embedding, N4-Methylcytosine

Background

DNA modifications play a crucial role in some biological processes and diseases, including development [1], aging [2], cancer [3], and regulation of gene expression [4]. Over 17 types of chemical modifications have been identified in DNA so far [5], including 5-methylcytosine (5mC), 5-hydroxymethylcytosine (5hmC), 5-formylcytosine (5fC), 5-carboxylcytosine (5caC), and N6-methyladenine (6mA) [6, 7]. DNA methylation is an important epigenetic modification, which is catalyzed by a family of DNA methyltransferases. DNA methylation is involved in heavy metal modification and regulation of chromatin as well as gene expression [8]. Methylation has widely been found in

prokaryotic and eukaryotic genomes, including N4-methylcytosine (4mC) [9], 5-methylcytosine (5mC) [10], and N6-methyladenine (6mA) [11, 12]. The 5mC is formed by transferring a methyl group from S-adenyl methionine to the fifth carbon of a cytosine residue [13] and has been widely demonstrated to play an important role in the biological progression associated with diabetes, cancer, and some neurological diseases [14–16]. The well-known DNA 6mA is a process of adding a methyl group to the 6-th position of an adenine ring catalyzed by DNA methyltransferases [17]. The 6mA is an essential epigenetic modification. The 4mC formation is catalyzed by N4-cytosine-specific DNA methyltransferases, which methylate the amino group at the fourth position of cytosine [18]. The 4mC is involved in such biological functions as host defense, transcription regulation, gene expression, and DNA replication [19]. In comparison with the 5mC and the 6mA, little was known about the 4mC modification and functions.

Detecting 4mC sites is critical to exploring 4mC functions. Many experimental methods have been developed to detect 4mC sites over the past decade, including single-molecule real-time sequencing (SMRT) and bisulfite sequencing [20]. These experimental methods generally are labor-intensive and time-consuming. The emergence of bioinformatics makes it possible to identify 4mC sites on a large scale. Hundreds of computational tools have been developed over the past thirty years for predicting post-translational modification [17, 21–23], RNA modification [24–26], single-cell analysis [27], protein functions [28, 29], as well as protein structure [30], gene selection [31], cancer diagnosis [32], and even food recommendation [33]. With the help of computers of powerful computational ability, the computational methods made great progress in the protein structure recognition, which is thought as one of best challenging tasks. AlphaFold [30] is approaching accuracy of the experimentally determining protein structure. The vast achievement attributed to design of machine learning algorithm, which refined high level representation of protein structure.

Following these successes, many computational tools have been developed for predicting 4mC sites [21, 26, 34, 35]. iDNA4mC [9], Meta-4mCpred [34], and 4mCPred [26] were designed to predict 4mC sites for six species: Arabidopsis thaliana (A. thaliana), Caenorhabditis elegans (C. elegans), Drosophila melanogaster (D. melanogaster), Escherichia coli (E. coli), Geobacter pickeringii (G. pickeringii) and Geoalkalibacter subterraneus (G. subterraneus). These methods are simple to use, but their capacity of detecting 4mC sites cross species need promoting. No less than five computational methods [36–41] were intended to predict 4mC sites in mouse genomes. Both 4mCPred-EL [36] and i4mC-Mouse [37] were feature engineering-based methods. 4mCPred-EL [36] integrated probabilities outputted by 28 classifiers which were generated by combining four common machine learning algorithms with seven types of features. The i4mC-Mouse [37] also used multiple representations and selected informative representations for detecting 4mC sites. The 4mCPred-CNN [38] and the Mouse4mC-BGRU [39] were deep learning-based schemes, which were quite different from the two methods above. The 4mCPred-CNN [38] used one-hot encoding and convolutional neural networks (CNNs), while the Mouse4mC-BGRU [39] utilized the adaptive embedding and bidirectional gated recurrent units (GRU). The deep learning-based methods exhibited a superiority over the 4mCPred-EL [36] and the i4mC-Mouse [37] in the empirical experiments. In addition, the 4mCPred-CNN [38] and the Mouse4mC-BGRU [39]

were an end-to-end learning method that takes primary sequences as input, and outputs directly predictive results, requiring no feature extraction.

Although vast efforts were paid to promote the predictive accuracy of identifying 4mC sites, there are some drawbacks to overcome. Representations of sequence are critical in the feature engineering-based methods. Most representations are degenerative mapping. That's to say, during the course of converting sequences into representation, there would be information lost. Ensemble of more representations would make the model improve predictive accuracy, but added the computational costs. Most representations are based on character statistics, disregarding the relationship between characters, namely contextual semantics. The deep learning-based methods are very promising to promote significantly predictive accuracy of 4mC recognitions. CNNs are good at characterizing local property, while the GRU and the long short-term memory (LSTM) do well in capturing contextual semantics. Different scale CNN reflects different scale information of sequences. The 4mCPred-CNN [38] failed to exploit multi-scale information.

To overcome the drawbacks above we proposed a deep neural network-based computational method named MultiScale-CNN-4mCPred for mouse 4mC site prediction. The MultiScale-CNN-4mCPred consisted of adaptive embedding, bidirectional LSTM (Bi-LSTM) followed by three scale CNNs, and fully-connected network. DNA sequences resemble sentences in the natural language, and thus there would be semantics hidden in it. The LSTM was intended to extract the semantics of 4mC DNA sequences, and the multi-scale CNN was intended to extract sequence representation at a different scale. The highlight of the MultiScale-CNN-4mCPred was summarized as follows. Firstly, the MultiScale-CNN-4mCPred combined the multi-scale CNNs and Bi-LSTM, and used them to capture scale representations and semantics respectively, which promoted predictive accuracy. The 4mCPred-CNN [38] didn't use LSTM or GRU to capture semantics, while the Mouse4mC-BGRU [39] didn't exploit CNNs to characterize local property. Secondly, the MultiScale-CNN-4mCPred used the adaptive embedding which is dynamic to represent sequence with comparison to one-hot encoding.

Results

Optimizing combination of different scale CNNs

Multiscale refers to signal sampling at different scales, and different characteristics can be observed at different scales. Larger scale characterizes global information, while the smaller scale reflects local characteristics. Therefore, the CNNs with different scales are of capacity to capture different information. We investigated the effect of the combination of different scale CNNs on the performances. We randomly selected 80% samples of the training dataset as the training samples and the remaining 20% as the validation set. Table 1 showed the performances of the combination of different scale CNNs. It was obviously observed that the combination of CNNs with the kernel sizes 3, 5, and 7 substantially outperformed others, indicating that the combination of these three scales can characterize well global, medium, and local structure. Therefore, we chose the above three scales for later experiments.

Table 1 The performance of different scale combinations

	Sn	Sp	Acc	MCC
1,3,5	0.7651	0.7867	0.7759	0.5519
3,5,7	**0.8188**	0.7867	**0.8027**	**0.6057**
5,7,9	0.7852	0.7800	0.7826	0.5652
7,9,11	0.7517	**0.8267**	0.7893	0.5801
11,13,15	0.7114	0.8333	0.7726	0.5490
13,15,17	0.7517	0.8200	0.7860	0.5731

The bold highlighted the best values in the column

Table 2 Comparison between different embedding

	Sn	Sp	Acc	MCC
One-hot	0.7584	**0.8400**	0.7993	0.6005
Word2Vec(1)	0.7315	0.8133	0.7726	0.5468
Word2Vec(2)	0.8188	0.7600	0.7893	0.5797
Word2Vec(3)	0.7248	0.8200	0.7726	0.5474
Adaptive Embedding	**0.8456**	0.7800	**0.8127**	**0.6269**

The bold highlighted the best values in the column

Comparison with different embedding

For text sequences, there are many frequently used embedding to represent it such as one-hot encoding and Word2Vec [42, 43]. One-hot encoding is well-known encoding in the natural language processing, where a word corresponds a 0/1 binary vector. One-hot encoding is unable to capture textual semantic relationship between words. Word-2vec translated each word into a contiguous vector which contains intrinsic semantics in the sequence. We compared the adaptive embedding with two types of embedding. In the Word2Vec, we considered 1, 2, and 3 contiguous nucleotides as a word respectively. Except for the embedding, the other parts of model are identical. As shown in Table 2, the overall performance of the adaptive embedding over the validation set was the best, with three indices (Sn, Acc, and MCC) to be in the lead. This is a reason to choose the adaptive embedding, not One-hot or Word2Vec.

Discussion

Comparison with state-of-the-art methods

As mentioned previously, many computational methods have been developed to predict DNA N4-methylcytosine in mouse genome over the recent ten years, such as Mouse4mC-BGRU [39], i4mC-Mouse [37], and 4mCPred-EL [36]. We performed the 10-fold cross-validation and the independent test to compare them. In the 10-fold cross-validation, the training set was randomly divided into ten parts in equal or approximate size. Nine parts were used for training the model, and the remaining one part for testing the trained model. This process was repeated ten times. In the independent test, the whole training set was used for training the model, and the testing set was used for testing the trained model. As shown in Table 3, Our MultiScale-CNN-4mCPred reached competitive performances with these state-of-the-art methods over the 10-fold cross-validation. For example, the MultiScale-CNN-4mCPred obtained the best Acc, the

Table 3 The performance over the 10-fold cross-validation

	Sn	Sp	Acc	MCC
4mCpred-EL*	0.8040	0.7870	0.7950	0.5910
i4mC-Mouse*	0.6831	0.9020	0.7930	0.6510
Mouse4mC-BGRU*	0.7940	0.8400	0.8100	0.6200
MultiScale-CNN-4mCPred	0.8008	0.8294	0.8166	0.6335

*Result came from the references [36, 37, 39]

Table 4 The performance over the independent test

	Sn	Sp	Acc	MCC
4mCpred-EL*	0.7572	0.8251	0.7910	0.5840
i4mC-Mouse*	0.8071	0.8252	0.8161	0.6330
Mouse4mC-BGRU*	0.8000	0.8500	0.8250	0.6510
MultiScale-CNN-4mCPred	0.8563	0.8375	0.8469	0.6939

*Result came from the references [36, 37, 39]

second-best MCC as well as Sn, and the third-best Sp. In addition, our method was more tradeoff between Sn and Sp than other methods, i.e., both Sn and Sp exceeded 0.8. Table 4 listed the performances over the independent test. Except for Sp, all indices of the MultiScale-CNN-4mCPred were best. The MultiScale-CNN-4mCPred increased Sn by 0.0563 over the Mouse4mC-BGRU [39], by 0.0492 over the i4mC-Mouse [37], and by 0.0991 over the 4mCpred-EL [36]. The MultiScale-CNN-4mCPred increased Acc by 0.0219 over the Mouse4mC-BGRU [39], by 0.0308 over the i4mC-Mouse [37], and by 0.0559 over the 4mCpred-EL [36]. The MultiScale-CNN-4mCPred outperformed Mouse4mC-BGRU [39] by 0.0429 MCC, the i4mC-Mouse [37] by 0.0609 MCC, and the 4mCpred-EL [36] by 0.1099 MCC. Therefore, according to the performances over both cross-validation and the independent test, the MultiScale-CNN-4mCPred is superior to these three methods. We compared MultiScale-CNN-4mCPred with the newly developed method 4mCPred-CNN [38] which is implemented into a user-friendly webserver: http://nsclbio.jbnu.ac.kr/tools/4mCPred-CNN/. We uploaded the independent set to the webserver of the 4mCPred-CNN [38] for conducting prediction. We counted the predicted results under various thresholds, as listed in Table 5. As a comparison, we also listed performance of MultiScale-CNN-4mCPred under various thresholds. The MultiScale-CNN-4mCPred was superior to the 4mCPred-CNN [38] in value of Acc and MCC. These results indicated that MultiScale-CNN-4mCPred is a competitive and advanced computational method for predicting DNA 4mC sites.

Contributions of CNNs to 4mC prediction

We also investigated the contributions of CNNs with different scales to 4mC prediction. Table 6 showed the performance of removing a CNN from the MultiScale-CNN-4mCPred each time. Exclusion of a scale CNN caused the performance to decrease. For example, excluding the CNN with the size 3 made Sn decrease, excluding the CNN with the size 5 made Sp decrease, and excluding the CNN with the size 7 made both Sn and

Table 5 Comparison with 4mCPred-CNN over the independent test for various thresholds

Method	4mCPred-CNN				MultiScale-CNN-4mCPred			
	Sn	Sp	Acc	MCC	Sn	Sp	Acc	MCC
0.1	0.9250	0.3438	0.6344	0.3302	0.9563	0.4875	0.7219	0.5024
0.2	0.8500	0.6063	0.7281	0.4704	0.9125	0.6063	0.7594	0.5449
0.3	0.7875	0.7375	0.7625	0.5257	0.9000	0.6938	0.7969	0.6068
0.4	0.6813	0.8000	0.7406	0.4847	0.8750	0.7625	0.8188	0.6416
0.5	0.6125	0.8625	0.7375	0.4906	0.8563	0.8375	0.8469	0.6939
0.6	0.4938	0.9125	0.7031	0.4474	0.7625	0.8688	0.8156	0.6348
0.7	0.3813	0.9438	0.6625	0.3931	0.6875	0.8875	0.7875	0.5869
0.8	0.2688	0.9750	0.6219	0.3443	0.6375	0.9313	0.7844	0.5950
0.9	0.1563	0.9875	0.5719	0.2586	0.4625	0.9625	0.7125	0.4907

Table 6 The performance on excluding a scale CNN

Removing scale	Sn	Sp	Acc	MCC
3	0.8000	0.8562	0.8281	0.6572
5	0.8500	0.8187	0.8343	0.6690
7	0.8375	0.8250	0.8312	0.6625

Table 7 The performance at single-scale CNN

Used scale	Sn	Sp	Acc	MCC
3	0.8750	0.7688	0.8219	0.6474
5	0.7875	0.9000	0.8437	0.6918
7	0.7875	0.8750	0.8312	0.6651

Sp decrease. Table 7 showed the performance of the MultiScale-CNN-4mCPred with the single-scale CNN. Different scale CNNs contributes differently. The CNN with the size 3 contributed mainly to Sn, the size 5 CNN to Sp, and the size 7 CNN to Sp and then to Sn. Therefore, the multi-scale CNNs contributed jointly to 4mC prediction, which is a reason to use it.

Generalized ability

We also tested the proposed method for the ability to predict 4mC sites in other species. We downloaded 6 training and 6 independent datasets respectively from six species: A. thaliana, C. elegans, D. melanogaster, E. coli, G. pickeringii, and G. subterraneus at http://thegleelab.org/Meta-4mCpred/4mCPredData.html [34]. The numbers of samples in the training and independent datasets were 3956 and 2500 in A. thaliana, 3108 and 1500 in C. elegans, 3538 and 2000 in D. melanogaster, 776 and 268 in E. coli, 1138 and 400 in G. pickeringii, and 1812 and 700 in G. subterraneus. We used the training dataset to train MultiScale-CNN-4mCPred, and used the independent dataset from the same species to test the trained model. Table 8 listed performances over independent datasets. In terms of MCC, the MultiScale-CNN-4mCPred performed better than both DeepTorrent [44] and 4mCPred [26] over three species: A. thaliana, C. elegans and

Table 8 Comparison with DeepTorrent and 4mCPred over six species

Method	Species	Sn	Sp	Acc	MCC
MultiScale-CNN-4mCPred	*A. thaliana*	0.7792	**0.8440**	**0.8116**	**0.6245**
	C. elegans	0.8467	**0.8987**	**0.8727**	**0.7463**
	D. melanogaster	**0.9060**	**0.8850**	**0.8955**	**0.7912**
	E. coli	0.8358	**0.8731**	0.8545	0.7094
	G. pickeringii	0.7150	0.8000	0.7575	0.5169
	G. subterraneus	0.7457	0.8400	0.7929	0.5883
DeepTorrent*	*A. thaliana*	**0.8182**	0.7803	0.7992	0.5989
	C. elegans	**0.9038**	0.8077	0.8558	0.7149
	D. melanogaster	0.8898	0.8220	0.8559	0.7135
	E. coli	**0.9231**	0.8462	**0.8846**	**0.7715**
	G. pickeringii	**0.8684**	**0.9474**	**0.9079**	**0.8183**
	G. subterraneus	0.8333	**0.9167**	**0.8750**	**0.7526**
4mCPred*	*A. thaliana*	0.7652	0.7652	0.7652	0.5300
	C. elegans	0.8558	0.7885	0.8221	0.6500
	D. melanogaster	0.8390	0.8136	0.8263	0.6500
	E. coli	0.8462	0.8077	0.8269	0.6500
	G. pickeringii	**0.8684**	0.6842	0.7763	0.5600
	G. subterraneus	**0.9167**	0.7500	0.8333	0.6800

*Result came from the reference [26, 44], and the bold highlighted the best values in the corresponding species

D. melanogaster, performed worse than DeepTorrent [44] over three species: E. coli, G. pickeringii, and G. subterraneus. The first three species and mouse are eukaryote, while the last three species are prokaryote. The results indicated that MultiScale-CNN-4mCPred would be more suitable to predict 4mC sites eukaryote than in prokaryote.

Conclusions

The 4mC is one of the epigenetic mechanisms. Precise and large-scale detecting 4mC sites is still challenging. We presented a multi-scale CNNs and adaptive embedding-based method called MultiScale-CNN-4mCPred for predicting 4mC sites in the mouse genomes. The MultiScale-CNN-4mCPred reached the state-of-the-art performance in the mouse genome, and showed curtain ability to predict DNA 4mC across species. This method is end-to-end without manual feature extraction, and is simple to realize. Different scales reflect different characterizations of 4mC. The MultiScale-CNN-4mCPred failed to discover the biological meaning of the different scale characterization, and textual semantic of single nucleotide residue. We also developed a user-friendly web application which is convenient to predict 4mC sites. The proposed method and the developed tool are beneficial to epigenetic research. In the future, we shall employ the Transformer [45] or the BERT [46] to improve the semantical representation of sequences, and exploit attention mechanism to promote interpretability of models.

Methods

As shown in Fig. 1, the presented MultiScale-CNN-4mCPred is a deep neural network-based method, consisting mainly of adaptive embedding [39], bi-directional long short-term memory (Bi-LSTM) [47], multi-scale CNNs [48, 49], and three fully-connected layers. The DNA letter sequences were first transformed into sequences of integers. Then, the

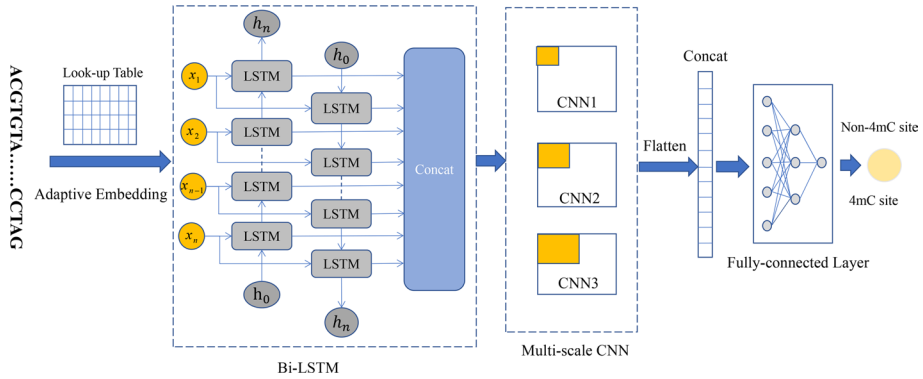

Fig. 1 Workflow of the MultiScale-CNN-4mCPred

Table 9 Number of parameters and shape of output in the proposed

Layers	Parameters	Output
Adaptive embedding	32	[None, 41, 8]
Bi-directional LSTM	720	[None, 41,12]
Multi-Scale CNN_3	333	[None, 39, 9]
Multi-Scale CNN_5	549	[None, 37, 9]
Multi-Scale CNN_7	765	[None, 35, 9]
Dropout_3	0	[None, 39, 9]
Dropout_5	0	[None, 37, 9]
Dropout_7	0	[None, 35, 9]
Concat	0	[None,111, 9]
Flatten	0	[None, 999]
Dense(27)	27,000	[None, 27]
Dropout	0	[None, 27]
Dense(9)	252	[None, 9]
Dense(1)	10	[None, 1]

adaptive embedding layer was used to map sequences of integers into continuous vectors. The Bi-LSTM layer was used to extract contextual semantics from DNA sequences. The multi-scale CNN layer was used to extract local features through multiple kernels of different scales. To reduce or avoid overfitting, the dropout was attached at the end of the multi-scale CNN layer and the first fully-connected layer, respectively. The multi-scale representations were concatenated to be fed into three fully-connected layers. The final output represented probabilities of predicting inputs as 4mC. The MultiScale-CNN-4mCPred is an end-to-end learning model. That's to say, the DNA sequence were fed into the MultiScale-CNN-4mCPred and the predictive results were directly returned without any manual operations. The parameters in each layer of the MultiScale-CNN-4mCPred were listed in Table 9. The total number of trainable parameters was 29661.

Character encoding

The DNA sequence consisted of four characters: A, T, G, and C. The deep neural network cannot directly process the character sequences. Therefore, the character must be converted into numerical sequences. Here, we defined a map as follows:

$$f(c) = \begin{cases} 0 \ if \ c \ was \ A \\ 1 \ if \ c \ was \ G \\ 2 \ if \ c \ was \ C \\ 3 \ if \ c \ was \ T \end{cases} \quad (1)$$

Adaptive embedding

Embedding is a way to transform discrete variables into continuous representations [50]. In neural networks, the embedding technique not only reduces the spatial dimension of discrete variables, but also meaningfully characterizes the variable. For example, the word2vec [42, 43] is a frequently used embedding algorithm in the field of natural language processing, where an interesting and famous example is that vector (king) − vector (man) + vector (women) ≈ vector (queen). On the contrary, the traditional one-hot encoding method translated each word into a binary vector, which has two limitations. Firstly, when the vocabulary was large, the vector generated by one-hot might be very sparse. Secondly, the one-hot encoding assumed that words were independent of each other. Thus, similarities between words could not be measured. We used adaptive embedding [39] as the first layer of the MultiScale-CNN-4mCPred.

CNN

CNN is a well-known neural network architecture [48]. CNN first appeared in the LeNet-5 which was a multi-layer neural network [51, 52] and was used to distinguish handwritten digits. The LeNet-5 used the backpropagation algorithm for training. The CNN attracted considerable attention for the AlexNet [53], a deep CNN neural network architecture that reduced significantly the error rate of image classification in comparison with other state-of-the-art methods. Since then, many variants of deep CNN architecture have been proposed, such as ZFNet [54], VGGNet [55], GoogleNet [56], and ResNet [57]. It has widely been demonstrated that these variants are of effectiveness and efficiency in most cases. For example, the ResNet [57] won the champion of ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2015. CNN is becoming one of the most popular and practically used components in the field of deep learning.

The CNN contained three basic components: convolution, pooling, and activations [58]. The aim of the convolution is to compute a feature map. Each neuron in the feature map is linked to a region of neighboring neurons in the previous layer which is generally called the receptive field. The kernel (also called filters) convolves with receptive fields to generate a feature map. The receptive fields slide at a certain stride in horizontal or vertical directions to generate various convolved results. The kernel is learnable and shared by all receptive fields. Various kernels yielded various feature maps. In most cases, multiple kernels are used to characterize the studied object from multiple perspectives. The convolved results were non-linearly transformed by the activation function. The

commonly used activation functions include Sigmoid [59], Tanh [60], and ReLU [61], The pooling is to reduce the dimension of the feature map, and thus reduced the complexity of training deep neural networks. The pooling included max pooling, min pooling, and mean pooling.

Bi-directional long short-term memory

Recurrent neural network (RNN) is a different architecture of neural network from the CNN. At the time step t in the RNN, the state of the hidden node $h^{(t)}$ was not only associated with the current input $x^{(t)}$, but also with the hidden state $h^{(t-1)}$ at the previous time step t-1, which was computed by

$$h^t = \sigma\left(W^{hx}x^{(t)} + W^{hh}h^{(t-1)} + b_h\right) \tag{2}$$

where W^{hx} denoted the linking weight between the input and the hidden state, and W^{hh} was the linking weight between the hidden states. The output at the time step t was computed by

$$\widehat{y}^{(t)} = softmax\left(W^{yh}h^{(t)} + b_y\right) \tag{3}$$

where W^{yh} denoted linking weight between output and hidden state, b_y denotes bias. The marked trait of the RNN was that all the nodes shared linking weights. The RNN is especially suitable for sequence analysis. Although RNN was capable of solving the order problem of neural network in time-based sequence, the RNN still had some limitations [62, 63]. The major limitation was to vanish and to explode for the gradient. The LSTM [64] solved well this limitation of the early RNN by replacing the traditional nodes of the hidden layer with a memory cell which was a unit of computation [65].

The common long short-term memory (LSTM) cell contained input, the previous hidden state, the previous cell state, the current hidden state, the current cell state, a candidate cell state, and three gates: forget gate, input gate, and output gate. The cell state was equivalent to the path of information transmission, which was the horizontal line with only some minor linear interactions, running straight down the entire chain, so that the previous information could flow along the sequence unchanged. The cell state was viewed as the "memory" of the neural network [62]. In theory, the cell state could transmit the relevant information forever. The addition and the removal of information in the sequence were controlled by the gates. The forget gate determined what information was preserved from the cell state. The forget gate outputted a number between 0 and 1, 0 representing completely getting rid of the information and 1 completely keeping it [62]. The input gate decided which value was updated. The new candidate cell state was created by the tanh function. The cell state was updated by multiplying the previous cell state by the forget gate, and by multiplying the candidate cell state by the input gate. The hidden state was updated by multiplying the output gate by tanh of the cell state. All the computations in each LSTM cell were listed as follows:

$$g^{(t)} = \emptyset\left(W^{gx}x^{(t)} + W^{gh}h^{(t-1)} + b_g\right) \tag{4}$$

$$i^{(t)} = \sigma\left(W^{ix}x^{(t)} + W^{ih}h^{(t-1)} + b_i\right) \tag{5}$$

$$f^{(t)} = \sigma\left(W^{fx}x^{(t)} + W^{fh}h^{(t-1)} + b_f\right) \tag{6}$$

$$o^{(t)} = \sigma\left(W^{ox}x^{(t)} + W^{oh}h^{(t-1)} + b_0\right) \tag{7}$$

$$s^{(t)} = g^{(t)} \odot i^{(t)} + s^{(t-1)} \odot f^{(t)} \tag{8}$$

$$h^{(t)} = \emptyset\left(s^{(t)}\right) \odot o^{(t)} \tag{9}$$

In equations above, W^{fx} and W^{fh} denoted the linking weight between the input and the forget gate as well as the linking weight between the forget gate and the hidden state. W^{ix}, W^{ih}, W^{gx} and W^{gh} denoted the linking weights between the input and the input gate, between the input gate and the hidden state, between the candidate cell and the input, as well as between the candidate cell and the hidden state, respectively. Besides, W^{ox} and W^{oh} denoted the linking weight between the input and the output gate, as well as between output gate and hidden state, respectively. The symbols σ and $\varnothing$ represented the activation function sigmoid and tanh, respectively, and the symbol $\odot$ represented the point by point multiplication of vectors.

The LSTM allowed the information to flow from the past to the future. In some cases, the output was not only related to the previous input, but also might be related to the future input. A Bi-LSTM proposed by Schuster et al. [47] addressed well the issue.

Dropout and flatten layer

The dropout was pioneered by Hinton et al. [66]. The dropout is that in the training stage, a certain proportion of the neurons were dropped out and parameters of only preserved neurons were updated. The aim of designing dropout have two-fold: avoiding overfitting and reducing the complexity of computation. Due to its effectiveness and efficiency, the dropout is increasingly becoming a frequently used trick in the deep learning area [67–69].

The flatten layer is to play the role of a bridge, which links the fully-connected layer to the non-fully connected layer. The flatten layer transformed the output of non-fully connected layer into one dimension, so that it could link to the subsequent fully-connected layer. The flatten layer has no trainable parameters. The fully-connected layer was similar to the hidden layer in the multilayer perceptron, where each neuron was connected to all the neurons in the preceding layer.

Datasets

For a fair comparison with the state-of-the-art methods, we used the same dataset as Mouse4mC-BGRU [39], 4mCPred-CNN [38], i4mC-Mouse [37], and 4mCPred-EL [36]. The dataset originated from the MethSMRT [70], which was the first database to deposit DNA 6mA and 4mC sites. The process of MethSMRT [70] collecting the dataset was

briefly described as follows. The MethSMRT [70] firstly retrieved the SMRT sequencing data from the NCBI Gene Expression Omnibus (http://www.ncbi.nlm.nih.gov/geo/) [71] and Sequence Read Archive (http://www.ncbi.nlm.nih.gov/sra) [72] and then employed PacBio SMRT analysis platform which is available at http://www.pacb.com/products-and-services/analytical-software/smrt-analysis/analysis-applications/epigenetics/ to detect 6mA and 4mC site. Reads with less than 50 nucleotides (nt), or a low-quality region (read score < 0.75 by default) were filtered out by using SFilter. The reads left were aligned to the reference genome by pbalign. The sites with less than 25-fold coverage per strand or with less than 20 modification scores were removed. The MethSMRT [70] hosted methylations of 156 species, including 7 eukaryotes and 149 prokaryotes. The DNA sequences were divided into windows of 41 base pairs with the cytosine at the center of it. The windows containing the 4mC sites were considered as positive samples and others were as negative ones. To reduce or remove the influence of sequence homology on the methods, the CD-HIT programming [73] was used to cluster the original sequences. The sequence identity was set to 0.7. Therefore, among the preserved sequences, the sequence identity between any two was less than 0.7. The same number of negative samples as the positive samples were randomly selected to a keep balance between them. All the samples were divided into the training set and the testing set. The training set consisted of 746 positive and 746 negative samples, while the testing set consisted of 160 positive and 160 negative samples.

Evaluation metrics

We adopted Sensitivity (Sn), Specificity (Sp), Accuracy (Acc), and Matthews correlation coefficient (MCC) for evaluating the performances, which were respectively defined as follows:

$$Sn = \frac{TP}{TP + FN} \tag{10}$$

$$Sp = \frac{TN}{TN + FP} \tag{11}$$

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{12}$$

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{13}$$

where TP represented the number of the true 4mC samples that were correctly predicted to be 4mC, TN the number of non-4mC samples that were correctly predicted to be non-4mC, FP the number of non-4mC samples that were incorrectly predicted to be 4mC, and FN the number of 4mC samples that were incorrectly predicted to be non-4mC.

The MultiScale-CNN-4mCPred was implemented by Tensorflow 2.9.1 framework, and all scripting programs were written via Python 3.8.12. The entire project was run on a Windows 10 system configured with 2.42 GHz CPU, 2 GB GPU, and 16 GB RAM. The

learnable parameters in MultiScale-CNN-4mCPred were randomly initiated. 20% of the training set is randomly sampled as the validation set.

Web server

For the purpose of conveniently using the MultiScale-CNN-4mCPred for 4mC site prediction, we implemented the proposed method into a user-friendly web server, which is freely accessible at http://www.biolscience.cn/MultiScale-CNN-4mCPred/. The web server is very easy to use. Users paste or upload DNA sequences in the FASTA format, and then click the "submit" button. The web server will perform predictions and return the predicted results to users.

Acknowledgements
Not applicable.

Author contributions
PZ: data curation, methodology, software, investigation, writing. GZ: methodology. YL: conceptualization, writing. GH: conceptualization, funding acquisition, supervision, writing—review and editing. All authors read and approved the final manuscript.

Funding
This work is supported by National Natural Science Foundation of China (No. 62272310), by Hunan Provincial Natural Science Foundation of China (No. 2022JJ50177, No. 2020JJ4034), by Scientific Research Fund of Hunan Provincial Education Department (No. 21A0466, No. 19A215), and by Shaoyang University Innovation Foundation for Postgraduate (CX2021SY052).

Availability of data and materials
The data is available at: http://www.biolscience.cn/MultiScale-CNN-4mCPred/, and the source code is available at: https://github.com/paomian97/MultiScale_CNN_4mCPred.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that no completing interest exist.

Received: 15 September 2022 Accepted: 4 January 2023
Published online: 18 January 2023

References

1. Greenberg MVC, Bourc'his D. The diverse roles of DNA methylation in mammalian development and disease. Nat Rev Mol Cell Biol. 2019;20:590–607. https://doi.org/10.1038/s41580-019-0159-6.
2. Unnikrishnan A, Freeman WM, Jackson J, Wren JD, Porter H, Richardson A. The role of DNA methylation in epigenetics of aging. Pharmacol Ther. 2019;195:172–85. https://doi.org/10.1016/j.pharmthera.2018.11.001.
3. Koch A, Joosten SC, Feng Z, de Ruijter TC, Draht MX, Melotte V, Smits KM, Veeck J, Herman JG, Van Neste L, et al. Analysis of DNA methylation in cancer: location revisited. Nat Rev Clin Oncol. 2018;15:459–66. https://doi.org/10.1038/s41571-018-0004-4.
4. Baylin S. DNA methylation and gene silencing in cancer. Nat Clin Pract Oncol. 2005;2:S4–11. https://doi.org/10.1038/ncponc0354.
5. Zhao LY, Song J, Liu Y, Song CX, Yi C. Mapping the epigenetic modifications of DNA and RNA. Protein Cell. 2020;11:792–808. https://doi.org/10.1007/s13238-020-00733-7.
6. Ramsawhook AH, Lewis LC, Eleftheriou M, Abakir A, Durczak P, Markus R, Rajani S, Hannan NRF, Coyle B, Ruzov A. Immunostaining for DNA modifications: computational analysis of confocal images. J Vis Exp. 2017. https://doi.org/10.3791/56318.
7. Yang S, Wang Y, Chen Y, Dai Q. MASQC: next generation sequencing assists third generation sequencing for quality control in N6-methyladenine DNA identification. Front Genet. 2020;11:269. https://doi.org/10.3389/fgene.2020.00269.
8. Costello JF, Plass C. Methylation matters. J Med Genet. 2001;38:285–303. https://doi.org/10.1136/jmg.38.5.285.
9. Chen W, Yang H, Feng P, Ding H, Lin H. iDNA4mC: identifying DNA N4-methylcytosine sites based on nucleotide chemical properties. Bioinformatics. 2017;33:3518–23. https://doi.org/10.1093/bioinformatics/btx479.

10. Ehrlich M, Wang RY. 5-Methylcytosine in eukaryotic DNA. Science. 1981;212:1350–7. https://doi.org/10.1126/science.6262918.
11. Davis BM, Chao MC, Waldor MK. Entering the era of bacterial epigenomics with single molecule real time DNA sequencing. Curr Opin Microbiol. 2013;16:192–8. https://doi.org/10.1016/j.mib.2013.01.011.
12. Pataillot-Meakin T, Pillay N, Beck S. 3-methylcytosine in cancer: an underappreciated methyl lesion? Epigenomics. 2016;8:451–4. https://doi.org/10.2217/epi.15.121.
13. Moore LD, Le T, Fan G. DNA methylation and its basic function. Neuropsychopharmacology. 2013;38:23–38. https://doi.org/10.1038/npp.2012.112.
14. Jones PA. Functions of DNA methylation: islands, start sites, gene bodies and beyond. Nat Rev Genet. 2012;13:484–92. https://doi.org/10.1038/nrg3230.
15. Ling C, Groop L. Epigenetics: a molecular link between environmental factors and type 2 diabetes. Diabetes. 2009;58:2718–25. https://doi.org/10.2337/db09-1003.
16. Yao B, Jin P. Cytosine modifications in neurodevelopment and diseases. Cell Mol Life Sci. 2014;71:405–18. https://doi.org/10.1007/s00018-013-1433-y.
17. Hou R, Wu J, Xu L, Zou Q, Wu Y-J. Computational prediction of protein arginine methylation based on composition–transition–distribution features. ACS Omega. 2020;5:27470–9. https://doi.org/10.1021/acsomega.0c03972.
18. Manavalan B, Hasan MM, Basith S, Gosu V, Shin T-H, Lee G. Empirical comparison and analysis of web-based DNA N4-methylcytosine site prediction tools. Mol Ther Nucl Acids. 2020;22:406–20. https://doi.org/10.1016/j.omtn.2020.09.010.
19. Khanal J, Tayara H, Zou Q, Chong KT. Identifying DNA N4-methylcytosine sites in the rosaceae genome with a deep learning model relying on distributed feature representation. Comput Struct Biotechnol J. 2021;19:1612–9. https://doi.org/10.1016/j.csbj.2021.03.015.
20. Yu M, Ji L, Neumann DA, Chung D-H, Groom J, Westpheling J, He C, Schmitz RJ. Base-resolution detection of N 4-methylcytosine in genomic DNA using 4mC-Tet-assisted-bisulfite-sequencing. Nucl Acids Res. 2015. https://doi.org/10.1093/nar/gkv738.
21. Huang G, Shen Q, Zhang G, Wang P, Yu ZG. LSTMCNNsucc: a bidirectional LSTM and CNN-based deep learning method for predicting lysine succinylation sites. Biomed Res Int. 2021;2021:9923112. https://doi.org/10.1155/2021/9923112.
22. Huang G, Zheng Y, Wu YQ, Han GS, Yu ZG. An information entropy-based approach for computationally identifying histone lysine butyrylation. Front Genet. 2019;10:1325. https://doi.org/10.3389/fgene.2019.01325.
23. Huang G, Zeng W. A discrete hidden Markov model for detecting histone crotonyllysine sites. MATCH Commun Math Comput Chem. 2016;75:717–30.
24. Lv Z, Zhang J, Ding H, Zou Q. RF-PseU: a random forest predictor for RNA pseudouridine sites. Front Bioeng Biotechnol. 2020;8:134. https://doi.org/10.3389/fbioe.2020.00134.
25. Chen W, Xing P, Zou Q. Detecting N6-methyladenosine sites from RNA transcriptomes using ensemble Support Vector Machines. Sci Rep. 2017;7:1–8. https://doi.org/10.1038/srep40242.
26. He W, Jia C, Zou Q. 4mCPred: machine learning methods for DNA N4-methylcytosine sites prediction. Bioinformatics. 2019;35:593–601. https://doi.org/10.1093/bioinformatics/bty668.
27. Dai Q, Bao C, Hai Y, Ma S, Zhou T, Wang C, Wang Y, Huo W, Liu X, Yao Y, et al. MTGIpick allows robust identification of genomic islands from a single genome. Brief Bioinform. 2018;19:361–73. https://doi.org/10.1093/bib/bbw118.
28. Kulmanov M, Hoehndorf R. DeepGOPlus: improved protein function prediction from sequence. Bioinformatics. 2020;36:422–9. https://doi.org/10.1093/bioinformatics/btz595.
29. Yu G, Zhao Y, Lu C, Wang J. HashGO: hashing gene ontology for protein function prediction. Comput Biol Chem. 2017;71:264–73. https://doi.org/10.1016/j.compbiolchem.2017.09.010.
30. Callaway E. "It will change everything": DeepMind's AI makes gigantic leap in solving protein structures. Nature. 2020;588:203–5. https://doi.org/10.1038/d41586-020-03348-4.
31. Saberi-Movahed F, Rostami M, Berahmand K, Karami S, Tiwari P, Oussalah M, Band SS. Dual regularized unsupervised feature selection based on matrix factorization and minimum redundancy with application in gene selection. Knowl Based Syst. 2022;256:109884. https://doi.org/10.1016/j.knosys.2022.109884.
32. Azadifar S, Rostami M, Berahmand K, Moradi P, Oussalah M. Graph-based relevancy-redundancy gene selection method for cancer diagnosis. Comput Biol Med. 2022;147:105766. https://doi.org/10.1016/j.compbiomed.2022.105766.
33. Rostami M, Oussalah M, Farrahi V. A novel time-aware food recommender-system based on deep learning and graph clustering. IEEE Access. 2022;10:52508–24.
34. Manavalan B, Basith S, Shin TH, Wei L, Lee G. Meta-4mCpred: a sequence-based meta-predictor for accurate DNA 4mC site prediction using effective feature representation. Mol Therapy-Nucl Acids. 2019;16:733–44. https://doi.org/10.1016/j.omtn.2019.04.019.
35. Wei L, Su R, Luan S, Liao Z, Manavalan B, Zou Q, Shi X. Iterative feature representations improve N4-methylcytosine site prediction. Bioinformatics. 2019;35:4930–7. https://doi.org/10.1093/bioinformatics/btz408.
36. Manavalan B, Basith S, Shin TH, Lee DY, Wei L, Lee G. 4mCpred-EL: an ensemble learning framework for identification of DNA N(4)-methylcytosine sites in the mouse genome. Cells. 2019;8:1332. https://doi.org/10.3390/cells8111332.
37. Hasan MM, Manavalan B, Shoombuatong W, Khatun MS, Kurata H. i4mC-Mouse: Improved identification of DNA N4-methylcytosine sites in the mouse genome using multiple encoding schemes. Comput Struct Biotechnol J. 2020;18:906–12. https://doi.org/10.1016/j.csbj.2020.04.001.
38. Abbas Z, Tayara H, Chong KT. 4mCPred-CNN—prediction of DNA N4-methylcytosine in the mouse genome using a convolutional neural network. Genes. 2021;12:296. https://doi.org/10.3390/genes12020296.
39. Jin J, Yu Y, Wei L. Mouse4mC-BGRU: Deep learning for predicting DNA N4-methylcytosine sites in mouse genome. Methods. 2022;204:258–62. https://doi.org/10.1016/j.ymeth.2022.01.009.
40. Zulfiqar H, Khan RS, Hassan F, Hippe K, Hunt C, Ding H, Song X-M, Cao R. Computational identification of N4-methylcytosine sites in the mouse genome with machine-learning method. MBE. 2021;18:3348–63. https://doi.org/10.3934/mbe.2021167.

41. Li Y, Zhao Z, Teng Z, Scribante A. i4mC-EL: identifying DNA N4-methylcytosine sites in the mouse genome using ensemble learning. Biomed Res Int. 2021;2021:1–11. https://doi.org/10.1155/2021/5515342.
42. Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J. Distributed representations of words and phrases and their compositionality. Adv Neural Inf Process Syst. 2013;26:3111–9.
43. Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. arXiv preprint arXiv:13013781. 2013. https://doi.org/10.48550/arXiv.1301.3781.
44. Liu Q, Chen J, Wang Y, Li S, Jia C, Song J, Li F. DeepTorrent: a deep learning-based approach for predicting DNA N4-methylcytosine sites. Brief Bioinform. 2020. https://doi.org/10.1093/bib/bbaa124.
45. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I. Attention is all you need. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. Long Beach, California, USA; 2017. p. 6000-10
46. Devlin J, Chang M-W, Lee K, Toutanova K. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:181004805. 2018. https://doi.org/10.48550/arXiv.1810.04805.
47. Schuster M, Paliwal KK. Bidirectional recurrent neural networks. IEEE Trans Signal Process. 1997;45:2673–81. https://doi.org/10.1109/78.650093.
48. Gu J, Wang Z, Kuen J, Ma L, Shahroudy A, Shuai B, Liu T, Wang X, Wang G, Cai J, et al. Recent advances in convolutional neural networks. Pattern Recogn. 2018;77:354–77. https://doi.org/10.1016/j.patcog.2017.10.013.
49. Shin HC, Roth HR, Gao M, Lu L, Xu Z, Nogues I, Yao J, Mollura D, Summers RM. Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning. IEEE Trans Med Imaging. 2016;35:1285–98. https://doi.org/10.1109/TMI.2016.2528162.
50. Inglesfield J. A method of embedding. J Phys C: Solid State Phys. 1981;14:3795.
51. LeCun Y, Boser B, Denker JS, Henderson D, Howard RE, Hubbard W, Jackel LD. Backpropagation applied to handwritten zip code recognition. Neural Comput. 1989;1:541–51. https://doi.org/10.1162/neco.1989.1.4.541.
52. LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proc IEEE. 1998;86:2278–324. https://doi.org/10.1109/5.726791.
53. Deng J, Dong W, Socher R, Li L-J, Li K, Fei-Fei L. Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition: 2009. IEEE: 248–255.
54. Zeiler MD, Fergus R. Visualizing and understanding convolutional networks. In: Fleet D, Pajdla T, Schiele B, Tuytelaars T, editors. European conference on computer vision. Springer; 2014. p. 818–33.
55. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:14091556. 2014. https://doi.org/10.48550/arXiv.1409.1556.
56. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A. Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition: 2015. 1–9.
57. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition: 2016. 770–778.
58. Albawi S, Mohammed TA, Al-Zawi S. Understanding of a convolutional neural network. In: 2017 international conference on engineering and technology (ICET): 2017. IEEE: 1–6.
59. Yin X, Goudriaan J, Lantinga EA, Vos J, Spiertz HJ. A flexible sigmoid function of determinate growth. Ann Bot. 2003;91:361–71. https://doi.org/10.1093/aob/mcg029.
60. Fan E. Extended tanh-function method and its applications to nonlinear equations. Phys Lett A. 2000;277:212–8. https://doi.org/10.1016/S0375-9601(00)00725-8.
61. Agarap AF. Deep learning using rectified linear units (relu). arXiv preprint arXiv:180308375. 2018. https://doi.org/10.48550/arXiv.1803.08375.
62. Olah C. Understanding lstm networks. 2015.
63. Bengio Y. Deep learning of representations for unsupervised and transfer learning. In: Proceedings of ICML workshop on unsupervised and transfer learning: 2012. JMLR Workshop and Conference Proceedings: 17–36.
64. Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput. 1997;9:1735–80. https://doi.org/10.1162/neco.1997.9.8.1735.
65. Lipton ZC, Berkowitz J, Elkan C. A critical review of recurrent neural networks for sequence learning. arXiv preprint arXiv:150600019. 2015. https://doi.org/10.48550/arXiv.1506.00019.
66. Hinton GE, Srivastava N, Krizhevsky A, Sutskever I, Salakhutdinov RR. Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:12070580. 2012. https://doi.org/10.48550/arXiv.1207.0580.
67. Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. Adv Neural Inf Process Syst. 2012;60:84–90. https://doi.org/10.1145/3065386.
68. Bouthillier X, Konda K, Vincent P, Memisevic R. Dropout as data augmentation. arXiv preprint arXiv:150608700. 2015. https://doi.org/10.48550/arXiv.1506.08700.
69. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res. 2014;15:1929–58.
70. Ye P, Luan Y, Chen K, Liu Y, Xiao C, Xie Z. MethSMRT: an integrative database for DNA N6-methyladenine and N4-methylcytosine generated by single-molecular real-time sequencing. Nucleic Acids Res. 2017;45:D85–9. https://doi.org/10.1093/nar/gkw950.
71. Clough E, Barrett T. The gene expression omnibus database. In: Mathé E, Davis S, editors. Statistical genomics. Springer; 2016. p. 93–110.
72. Leinonen R, Sugawara H, Shumway M. The sequence read archive. Nucleic Acids Res. 2010;39:D19–21. https://doi.org/10.1093/nar/gkq1019.
73. Li W, Godzik A. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. Bioinformatics. 2006;22:1658–9. https://doi.org/10.1093/bioinformatics/btl158.

Publisher's Note