Bioinformatics, 2022, 1–9
https://doi.org/10.1093/bioinformatics/btac454
Advance Access Publication Date: 7 July 2022
Original Paper

Genome analysis

DeepGenGrep: a general deep learning-based predictor for multiple genomic signals and regions

Quanzhong Liu [1], Honglin Fang[1], Xiao Wang[1], Miao Wang[1], Shuqin Li[1], Lachlan J. M. Coin[2], Fuyi Li [1,2,*] and Jiangning Song [3,4,*]

[1]Department of Software Engineering, College of Information Engineering, Northwest A&F University, Yangling 712100, China, [2]Department of Microbiology and Immunology, The Peter Doherty Institute for Infection and Immunity, The University of Melbourne, Melbourne, VIC 3000, Australia, [3]Biomedicine Discovery Institute and Department of Biochemistry and Molecular Biology, Monash University, Melbourne, VIC 3800, Australia and [4]Monash Data Futures Institute, Monash University, Melbourne, VIC 3800, Australia

*To whom correspondence should be addressed.

Associate Editor: Alfonso Valencia

Received on January 12, 2022; revised on April 11, 2022; editorial decision on July 2, 2022; accepted on July 6, 2022

Abstract

Motivation: Accurate annotation of different genomic signals and regions (GSRs) from DNA sequences is fundamentally important for understanding gene structure, regulation and function. Numerous efforts have been made to develop machine learning-based predictors for *in silico* identification of GSRs. However, it remains a great challenge to identify GSRs as the performance of most existing approaches is unsatisfactory. As such, it is highly desirable to develop more accurate computational methods for GSRs prediction.

Results: In this study, we propose a general deep learning framework termed DeepGenGrep, a general predictor for the systematic identification of multiple different GSRs from genomic DNA sequences. DeepGenGrep leverages the power of hybrid neural networks comprising a three-layer convolutional neural network and a two-layer long short-term memory to effectively learn useful feature representations from sequences. Benchmarking experiments demonstrate that DeepGenGrep outperforms several state-of-the-art approaches on identifying polyadenylation signals, translation initiation sites and splice sites across four eukaryotic species including *Homo sapiens*, *Mus musculus*, *Bos taurus* and *Drosophila melanogaster*. Overall, DeepGenGrep represents a useful tool for the high-throughput and cost-effective identification of potential GSRs in eukaryotic genomes.

Availability and implementation: The webserver and source code are freely available at http://bigdata.biocie.cn/deepgengrep/home and Github (https://github.com/wx-cie/DeepGenGrep/).

Contact: jiangning.song@monash.edu or fuyi.li@unimelb.edu.au

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Eukaryotic organisms have very complex gene structures with multiple genomic signals and regions (GSRs), such as promoter, transcription start site (TSS), polyadenylation signal (PAS), splice site, translational initiation site (TIS), etc. GSRs play vital roles in regulating gene expression in eukaryotic cells (Hartwell *et al.*, 2011). In this context, accurate identification of these GSRs can enable a better understanding of the gene structures and high-quality genome annotation (Kalkatawi *et al.*, 2019), which is approached by combining experimental techniques and computational methods (Bajic *et al.*, 2002). Generally, computational methods can help narrow down the number of target sequences and guide wet-lab design and hypothesis-driven analyses (Bajic *et al.*, 2002). The genomic sequences generated by next-generation sequencing (NGS) techniques are often annotated incompletely and as a result, there remain uncharacterized genomic 'dark matter' (Sterck *et al.*, 2012). Accordingly, computational approaches that can accurately identify GSRs based on genome sequence information are valuable and can provide important biological insights into the genome structure, regulation and function.

A variety of machine learning-based approaches have been developed to identify specific types of GSRs over the last decade, including promoters (Li *et al.*, 2021a; Liu *et al.*, 2019; Long Vo *et al.*, 2020; Rahman *et al.*, 2019; Zhang *et al.*, 2019, 2022), TSS (Down and Hubbard, 2002; Georgakilas *et al.*, 2014, 2020; Morton *et al.*, 2015), splice site (Degroeve *et al.*, 2005; Mort *et al.*, 2014; Zhang *et al.*, 2003), PAS (Ji *et al.*, 2018; Magana-Mora *et al.*, 2017; Xie *et al.*, 2013) and TIS (Chen *et al.*, 2014; Gao *et al.*, 2010). These methods typically rely on the handcrafted features extracted from

the DNA sequences, such as nucleic acid compositions and a wide range of physicochemical properties of the nucleic acids (Chen *et al.*, 2020, 2021). Subsequently, different types of machine learning algorithms are used to train the prediction models. More recently, deep learning techniques have been increasingly developed to address the GSR prediction task, such as DeeReCT-PromID (Umarov *et al.*, 2019), ReFeaFi (Umarov *et al.*, 2021) and Depicter (Zhu *et al.*, 2021) for the promoter prediction; Splice2Deep (Albaradei *et al.*, 2020), SpliceFinder (Wang *et al.*, 2019) and SpliceRover (Zuallaert *et al.*, 2018) for splice site; DeepPASTA (Arefeen *et al.*, 2019), SANPolyA (Yu and Dai, 2020), DeeReCT-PolyA (Xia *et al.*, 2019) and DeeReCT-APA (Li *et al.*, 2021b) for PAS; and TITER for TIS (Zhang *et al.*, 2017; Zuallaert *et al.*, 2017). These methods employ the one-hot encoding to represent and encode the input nucleotides sequence and apply deep learning to automatically learn high-level feature representations to better recognize specific GSRs.

In the aforementioned studies, both machine learning and deep learning-based approaches have been developed for predicting specific type(s) of GSR. These methods typically formulate the GSR identification task as a binary classification problem. The models are trained on a well-curated dataset with two classes, e.g. the positive samples (true signals) and the negative samples (false signals). Despite the progress made, these approaches have some drawbacks or weaknesses: (i) first, the performance of machine learning-based methods primarily relies on the handcrafted features (Eraslan *et al.*, 2019; Li *et al.*, 2020). However, such handcrafted features can contain noisy, irrelevant and/or redundant features, negatively impacting the training and performance evaluation of the models. Therefore, it is often a trial-and-error process to select relevant and contribute features to model training. Moreover, it remains a difficult task to accurately describe the characteristics of GSRs all around; (ii) second, most approaches are designed and evaluated for a specific type of GSR in a single species (Kalkatawi *et al.*, 2019). However, GSRs interact with each other (Sosa *et al.*, 2020), and therefore predictors that consider multiple GSRs simultaneously might provide more accurate genome annotations. In this regard, only one method, DeepGSR (Kalkatawi *et al.*, 2019), can identify various GSRs simultaneously, which was evaluated on PAS and TIS signals in four species and shown to outperform previous specific GSR predictors; (iii) the predictive performance of existing approaches needs to be improved. For example, although DeepGSR achieved better performance than other tools, there is further room for performance improvement. In addition, DeepGSR was only designed for two signals, and other GSRs need to be considered to develop more robust and accurate models.

To address these shortcomings, here we propose a novel generalized deep learning framework, referred to as DeepGenGrep (Deep learning-based model for genomic signals and regions prediction), for predicting multiple GSRs in the genome sequence. DeepGenGrep employs hybrid deep neural networks comprising a three-layer convolutional neural network and a two-layer long short-term memory to efficiently learn feature representations of the three types of GSRs from genomic sequences. Ablation studies show that DeepGenGrep is capable of identifying multiple GSRs, and the feature representations learned by DeepGenGrep are effective for classifying GSRs. Its predictive performance is validated using the data of three types of GSRs, including PAS and TIS of *Homo sapiens*, *Mus musculus*, *Bos taurus* and *Drosophila melanogaster*, and splice site signals of *H.sapiens*. Benchmarking experiments demonstrate that DeepGenGrep outperforms DeepGSR and other existing start-of-the-art GSR-specific predictors. Moreover, DeepGenGrep also has substantially improved transfer learning capability compared with DeepGSR.

2 Materials and methods

2.1 Datasets

In this study, three types of GSRs, including PAS, TIS and splice sites, were used for model training and evaluation. The PAS and TIS datasets of four species (*H.sapiens*, *M.musculus*, *B.taurus* and *D.melanogaster*) were extracted from DeepGSR's datasets originally prepared by Kalkatawi et al. (Kalkatawi *et al.*, 2019). There were 16 PAS variants, including 'AATAAA', 'ATTAAA', 'TATAAA', 'AGTAAA', 'AAGAAA', 'AATATA', 'AATACA', 'CATAAA', 'GATAAA', 'AATGAA', 'ACTAAA', 'AATAGA', 'TTTAAA', 'AAAAAG', 'AAAACA' and 'GGGGCT'. Only the canonical ATG variant was considered as the TIS signal. Each sequence consists of 300 upstream nucleotides, the considered signal variant and 300 downstream nucleotides. As a result, each PAS sequence has 606 nucleotides, while each TIS sequence has 603 nucleotides. In total, there are 20 933, 18 693, 12 082 and 27 203 PAS sequences, and 28 244, 25 205, 17 558 and 30 283 TIS sequences for *H.sapiens*, *M.musculus*, *B.taurus* and *D.melanogaster*, respectively. Besides, the false PAS and false TIS samples with the same numbers of PAS and TIS samples for each species were extracted from the chromosomes (21, 13, 28, X for *H.sapiens*, *M.musculus*, *B.taurus* and *D.melanogaster*, respectively) with the closest average GC-content compared with the whole genome. In this way, PAS and TIS sequences were regarded as the positive samples, while the sequences with false PAS and TIS (i.e. pseudo sequences containing the canonical PAS and TIS motifs, respectively, but were not related to the polyadenylation translation process) were used as the negative samples. Datasets with the two classes for each species were constructed. A detailed summary of the PAS and TIS datasets is provided in Supplementary Tables S1 and S2, respectively.

The splice site dataset was collected from the Genome Wide datasets for Human (GWH) (Lee and Yoon, 2015). GWH contains two types of splice site datasets, GWH donor and GWH acceptor. Each dataset includes 24 sub-datasets, and each sub-dataset contains sequences from one of the 24 human chromosomes (22 autosomes and two sex chromosomes). The length of all sequences in each sub-dataset is 398-nt. Canonical dimer GT is located in nucleotide positions 200 and 201 of each sequence in the GWH-donor dataset, while canonical dimer AG is located in positions 198 and 199 of each sequence in the GWH-acceptor dataset. These canonical dimers denote true splice sites for the positive samples, but do not indicate splice sites for the negative samples. The ratio of the positive and negative samples on each sub-dataset is 1:5. A detailed statistical summary of the datasets is provided in Supplementary Table S3.

2.2 The framework of DeepGenGrep

An overview of the architecture of DeepGenGrep is illustrated in Figure 1. DeepGenGrep consists of an input layer, three layers of convolutional neural network (CNNs), two layers of long short-term memory (LSTMs), a fully connected layer and an output layer.

For a sequence s, DeepGenGrep predicts whether s contains specific GSRs based on the following major steps.

$$f(s) = \mathrm{den}(\mathrm{lstm}(\mathrm{conv}_3(\mathrm{mpool}(\mathrm{conv}_2(\mathrm{mpool}(\mathrm{conv}_1(s))))))) \quad (1)$$

where s is first encoded into $l \times 4$ matrix and l is the length of s. Then, the encoded matrix is fed into three layers of CNN blocks (i.e. conv1, conv2 and conv3) with varying scale kernels to learn local features. To prevent overfitting, the max polling (mpool) is used to downsample the feature maps of the three CNN blocks. Subsequently, two LSTMs layers (lstm) are used to focus on the order-dependent input sequence. Then, the dense layers (den) with one unit and the sigmoid activation are used to output the probability at which s contains specific GSRs. A detailed description of each stage is provided as follows.

DeepGenGrep employs the one-hot encoding scheme (Chen *et al.*, 2021; Li *et al.*, 2020; Liu *et al.*, 2021; Pedregosa *et al.*, 2011) to represent the nucleotide sequences in the input layer. The one-hot encoding scheme encodes the mononucleotides into a 4D vector. Specifically, 'A', 'T', 'C' and 'G' are encoded into (1, 0, 0, 0), (0, 1, 0, 0), (0, 0, 1, 0), and (0, 0, 0, 1), respectively. Therefore, an l-bp nucleotide sequence s would be represented as an $l \times 4$ matrix.

Then, the encoded matrix is fed to the first CNN module (CNN-1), which consists of three parallel 1D-CNNs with different kernel sizes (1, 3 and 5, respectively) and filters (29, 121 and 467,

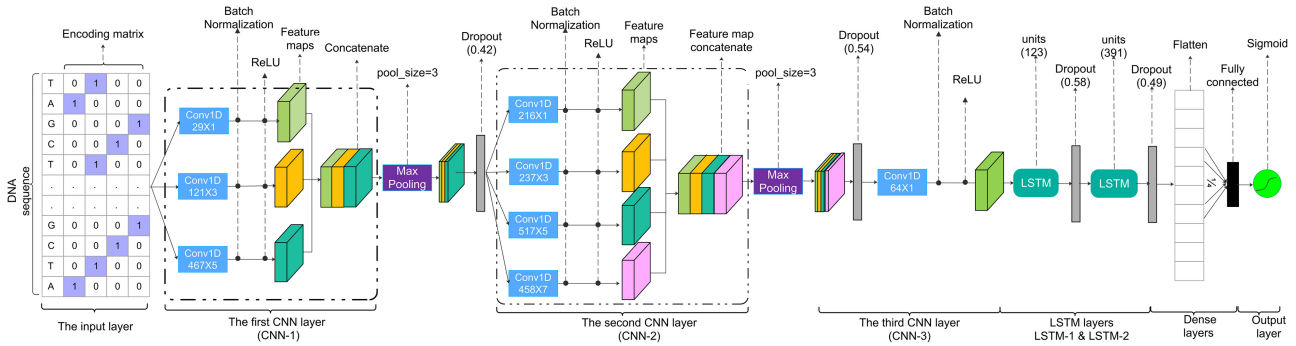

Fig. 1. An overview of the DeepGenGrep architecture

respectively) to capture various scales of the abstract features (Szegedy *et al.*, 2015). To improve the training speed and generalization ability, batch normalization (Ioffe and Szegedy, 2015) is used after each CNN, and the ReLU function is followed for activation. Then the three feature maps are concatenated into an integrated feature map, and a max-pooling layer is applied to reduce the spatial dimension. Finally, a dropout layer is followed to reduce overfitting. The second CNN layer (CNN-2) has four parallel 1D-CNNs with kernel sizes of 1, 3, 5 and 7, and corresponding filter sizes of 216, 237, 517 and 458, respectively. The following architecture is similar to CNN-1, equipping batch normalization, ReLU, max-pooling and dropout layers. The third CNN layer (CNN-3) consists of 64 1D-CNNs, all of which have kernel size 1 and are used as dimension reduction modules to shrink the number of parameters of the network (Szegedy *et al.*, 2015). Then the 1D-CNNs are followed by batch normalization and the ReLU activation function. By stacking these three CNN models (i.e. CNN-1, CNN-2 and CNN-3), DeepGenGrep constructs a deep CNN framework to learn high-level feature representations from the input matrix.

Then, the learned features are fed into two layers of LSTM, a type of recurrent neural network (RNN) that can capture the interdependencies across the sequence. The deep CNN in DeepGenGrep is therefore used as the pre-processing step for the LSTM, which has been shown as an excellent strategy to leverage the strengths of both CNN and RNN (Chollet, 2018). These two LSTM layers in DeepGenGrep can enable the learning of better feature representations based on the features learned by the deep CNN. To reduce the risk of overfitting, each LSTM is followed by a dropout layer. Finally, the output is flattened into a feature vector and fed into the fully connected layer with one output neuron. The output neuron uses the sigmoid function to generate the final prediction results.

The early stopping strategy is used to monitor the validation accuracy during the model training to avoid model overfitting. The patience epoch was set to 30, and the maximum number of epochs was set to 150. In addition, we employed the Bayesian optimization algorithm (Snoek *et al.*, 2012) to optimize the hyperparameters of DeepGenGrep. A detailed list of the predefined search ranges and the optimal values for each hyperparameter is provided in Supplementary Table S4.

2.3 Feature visualization

Visualization of the feature representations learned by deep learning models can help us better understand the learning ability of the trained model. Herein, we applied a two-step approach, which combines autoencoder with t-SNE (van der Maaten and Hinton, 2008), to evaluate the effectiveness of feature representations learned by DeepGenGrep. In the first step, we constructed an autoencoder to reduce the dimensionality of features learned by DeepGenGrep to 64D feature vectors. Then, in the second step, we used the t-SNE algorithm to visualize the 64D feature vectors obtained in the first step in two dimensions. We applied this two-step dimensionality reduction and visualization method considering although t-SNE can reduce the dimensionality of the input features, its dimensionality reduction performance is not satisfactory. Using an autoencoder to first reduce the dimensionality and then applying t-SNE for visualization can improve the quality of visualization results (Kingma and Welling, 2013). This strategy has been successfully used in a number of bioinformatics studies on feature visualization, such as representation of experimental replicates in each cell population (Macosko *et al.*, 2015), single-cell RNA sequencing analysis (Lin *et al.*, 2020) and multiple cell lines analysis (Boyd *et al.*, 2020).

The autoencoder architecture used in this study is illustrated in Supplementary Figure S1. As can be seen, the autoencoder architecture employs a neural network consisting of an encoder and a decoder. Generally, the encoder and decoder networks comprise a series of dense layers with the ReLU activation function. The encoder in our autoencoder architecture consists of three layers of dense networks with 256, 192 and 128 units, respectively. Accordingly, the decoder's three layers of dense networks have 128, 192 and 256 units, respectively. The encoder maps high-dimensional abstract features extracted from a specific layer to low-dimensional features in the latent layer, a dense layer with 64 units. Subsequently, the decoder restores the features in the latent layer to the original input high-dimensional abstract features as far as possible.

2.4 Performance evaluation metrics

We use five popular performance evaluation metrics, including sensitivity (*Sn*), specificity (*Sp*), *precision*, accuracy (*Acc*), *F1*-score and MCC, to evaluate and compare the predictive performance of DeepGenGrep with state-of-the-art approaches. These performance metrics are defined as follows:

$$\mathrm{Sn} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{2}$$

$$\mathrm{Sp} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}} \tag{3}$$

$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{4}$$

$$\mathrm{Acc} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \tag{5}$$

$$\mathrm{F1-score} = \frac{2 \times \mathrm{Precision} \times \mathrm{Se}}{\mathrm{Precision} + \mathrm{Se}} = \frac{2 \times \mathrm{TP}}{2 \times \mathrm{TP} + \mathrm{FP} + \mathrm{FN}} \tag{6}$$

$$\mathrm{MCC} = \frac{\mathrm{TP} \times \mathrm{TN} - \mathrm{FP} \times \mathrm{FN}}{\sqrt{(\mathrm{TP} + \mathrm{FP}) \times (\mathrm{TP} + \mathrm{FN}) \times (\mathrm{TN} + \mathrm{FP}) \times (\mathrm{TN} + \mathrm{FN})}} \tag{7}$$

where TP, TN, FP and FN denote the numbers of true positives, true negatives, false positives and false negatives, respectively. In addition, we also calculated the areas under the receiver-operating characteristic curve (AUC) and the precision-recall curve (AUPRC).

3 Results and discussions

In this section, we evaluated and compared the performance of DeepGenGrep with that of the state-of-the-art methods for predicting three genome signals, including TIS, PAS and splice sites. For TIS and PAS, the performance of DeepGenGrep was evaluated on the datasets of four species (*H.sapiens*, *M.musculus*, *B.taurus* and *D.melanogaster*) used in DeepGSR (Kalkatawi *et al.*, 2019). The performance of DeepGenGrep for predicting splice sites was assessed on the GWH genome dataset used in SpliceRover (Zuallaert *et al.*, 2018) and DBN (Lee and Yoon, 2015).

We used the same performance evaluation strategies as the DeepGSR to make fair and objective comparisons for the prediction of TIS and PAS signals. The dataset was first randomly split into the benchmark (75%) and independent test (25%) datasets for each species. Then, the benchmark dataset was further divided into the training and validation datasets with a ratio of 8:2. DeepGenGrep was trained on the training dataset and evaluated on the validation dataset during training. Then, its generalization ability was further tested on the independent test dataset. In contrast, for splice site prediction, the performance DeepGenGrep was evaluated by 10-fold cross-validation, which is in line with previous studies of SpliceRover and DBN.

3.1 Performance evaluations

3.1.1 Model training and validation

In this section, we trained the models of DeepGenGrep and evaluated its performance. Figure 2A shows the loss and accuracy curves on the training and validation datasets of *H.sapiens*. As can be seen, the training loss was continuously decreased from the first epoch to the 60th epoch. However, the validation loss was first reduced between the first and seventh epochs, where the model achieved the minimum loss, which then increased after that. Both the training and validation accuracies increased continuously at first and achieved the best results at the 30th epoch. The training was interrupted at the 60th epoch as the validation accuracy between the 30th and 60th epoch remained stable and was not further improved. Taken together, these results demonstrate the DeepGenGrep model achieved excellent performance for predicting TIS in *H.sapiens*. The training and validation performance results of DeepGenGrep for TIS prediction of *M.musculus*, *B.taurus* and *D.melanogaster* datasets are shown in Figure 2B–D, respectively. We can see that DeepGenGrep also achieved excellent training and validation performance for TIS recognition in *M.musculus*, *B.taurus* and *D.melanogaster*.

3.1.2 Feature representations learned by DeepGenGrep

To examine the effectiveness of feature representations learned by the trained DeepGenGrep model, we applied a two-level feature visualization strategy (refer to Section 2.3 for detail) to visualize such feature representations. In particular, to better understand the varying representation ability of learned features from layer to layer, we visualized abstract features in the input layer, three CNN layers and two LSTM layers of the trained DeepGenGrep model, respectively. Figure 3 illustrates feature representations of six layers of the model trained on the TIS dataset of *H.sapiens*. The input representation of the dataset is shown in Figure 3A. As can be seen, data points with TIS signals were mixed with those with non-TIS signals. After three layers of CNNs (Fig. 3B–D), data points with TIS signals were gradually separated from those with non-TIS signals. Based on the feature representations learned after the first LSTM layer of the model, we can see that data points were nearly partitioned into two

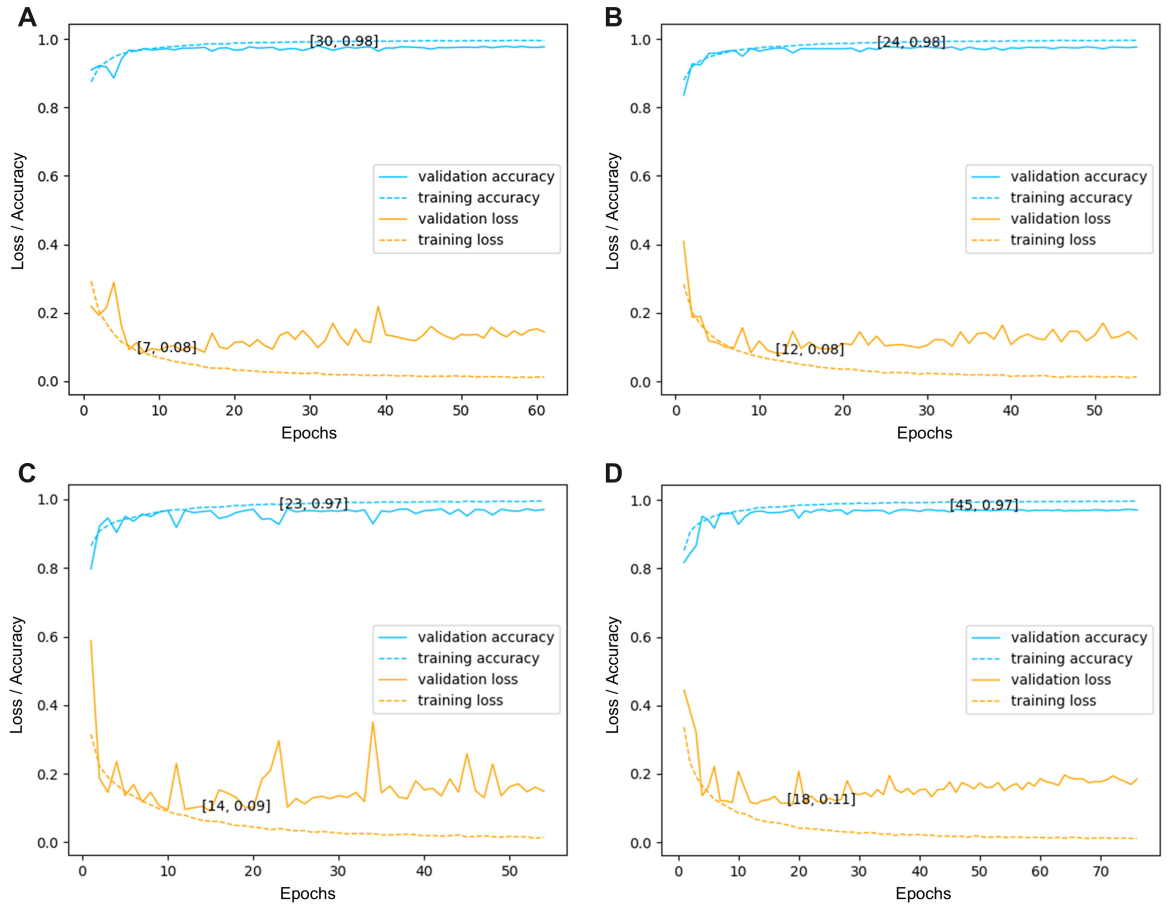

Fig. 2. Training and validation loss and accuracy of the DeepGenGrep model for predicting the TIS signal on the datasets of (**A**) *H.sapiens*, (**B**) *M.musculus*, (**C**) *B.taurus* and (**D**) *D.melanogaster*

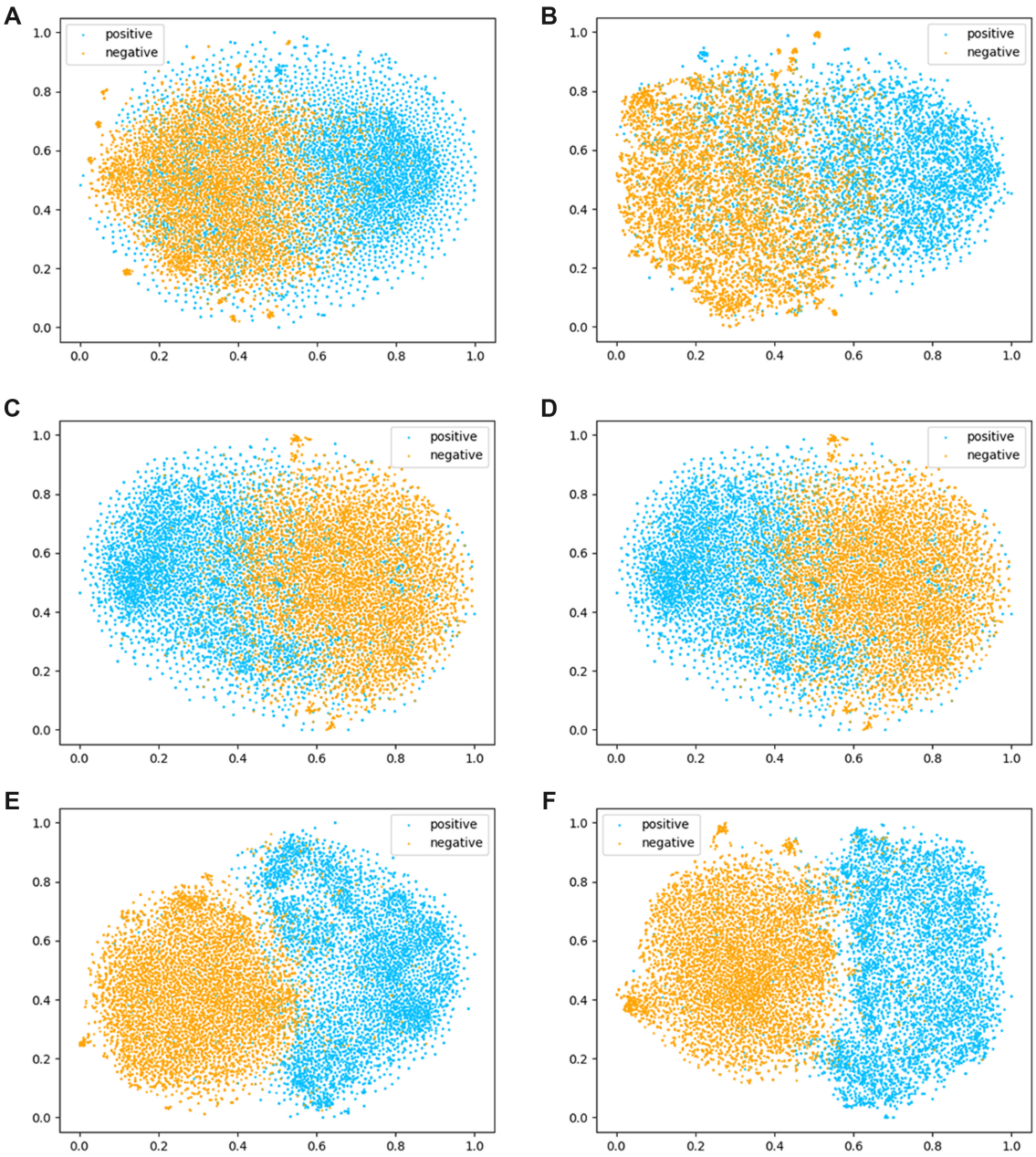

Fig. 3. t-SNE plots of (**A**) the original input encoding, (**B**)feature representation after the first convolution layers, (**C**) feature representation after the second convolution layers, (**D**)feature representation after the third convolution layers, (**E**) feature representation after the first LSTM layer and (**F**) feature representation after the second LSTM layer, for the model trained on the TIS dataset of *H.sapiens*

groups (Fig. 3E). After the second LSTM layer of the model, the data points appeared to be more distinctly separated into two groups (Fig. 3F). Moreover, we further employed UMAP (Mcinnes *et al.*, 2018) to examine the quality of feature representations and accordingly provided the feature visualization of the DeepGenGrep in Supplementary Figures S2–S16. As can be seen, UMAP generated similar feature representations for both TIS and PAS signals, thereby highlighting that DeepGenGrep is capable of effectively learning useful feature representations.

3.1.3 Ablation analysis

As mentioned before, the DeepGenGrep framework consists of three CNN modules (CNN_1, CNN_2 and CNN_3) and two LSTM layers (LSTM_1 and LSTM_2). We performed an ablation analysis to investigate the effectiveness of each layer in the DeepGenGrep to evaluate the models' performance by removing a specific module/layer from the framework (Li *et al.*, 2020; Ren *et al.*, 2017). We accordingly constructed five comparison frameworks by removing one of five modules/layers from DeepGenGrep to conduct the performance comparison with the full DeepGenGrep model. To effectively evaluate the model's performances, we performed 10-time tests on the *H.sapiens* species for TIS, PAS and splice acceptor for each comparison framework and the full DeepGenGrep framework, respectively. We randomly split three signal (TIS, PAS and splice acceptor) datasets on *H.sapiens* into training datasets and independent test datasets in each test, and the average results of 10 tests are reported. As a result, we run 60-time random experiments for each signal. The performance comparison results on the TIS are provided in Table 1. The 'Remove layer' column represents the modules/layers removed from the DeepGenGrep. As can be seen, the full DeepGenGrep model achieved the best performance in terms of *Acc*, *Precision* and *F1-score* in all six compared models for TIS of *H.sapiens*. The

Table 1. Ablation experiments on the TIS dataset of *H.sapiens*

Layer removed	Acc (%)	Se (%)	Sp (%)	Precision (%)	F1-score
CNN-1	97.03	95.90	98.16	96.00	0.971
CNN-2	97.14	96.18	98.10	96.26	0.972
CNN-3	96.86	95.13	**98.58**	95.31	0.969
LSTM-1	97.25	**96.70**	97.80	96.75	0.973
LSTM-2	97.31	96.22	98.38	96.31	0.973
None	**97.35**	96.62	98.07	**96.70**	**0.974**

Note: Bold entries denote the best performances on the evaluation metrics of the corresponding columns.

performance comparison results on the *H.sapiens* for the PAS and splice acceptor are provided in Supplementary Tables S5 and S6, respectively. As can be seen, the full DeepGenGrep model achieved the best performance in terms of *Acc*, *Se*, *Precision* and *F1-score* in all six compared models for predicting PAS in *H.sapiens*. The performance of the full DeepGenGrep model was close to that of the framework in which the LSTM_2 layer was removed for predicting the splice acceptor.

We further performed a student's *t*-test to examine the statistical difference in the performance between the compared frameworks and the full DeepGenGrep model in terms of *Acc*. The performance results of the prediction of TIS, PAS and splice acceptor are provided in Supplementary Tables S7–S9, respectively. As can be seen, when CNN-3 was removed from DeepGenGrep for TIS, the variance difference was not significant (P-value = 0.795); however, the mean difference was statistically significant (P-value = 0.021). These results indicate that the performance of the model could be significantly decreased when removing CNN-3 for *H.sapiens* TIS. In contrast, the model performance would not be substantially reduced when removing the other layers of the network for *H.sapiens* TIS. For *H.sapiens* PAS, when any layer except LSTM-2 was removed, the mean difference was statistically significant (Supplementary Table S8), which means that the performance of the model would be significantly decreased. For *H.sapiens* splice acceptor, when any layer except LSTM-1 and LSTM-2 was removed, the mean difference was statistically significant (Supplementary Table S9), which means the performance of the model would be significantly decreased. Although the full DeepGenGrep framework does not significantly outperform other compared frameworks under all the different scenarios, the ablation test shows that the DeepGenGrep framework is able to achieve a reasonable performance for recognizing GSRs in general.

3.2 Performance comparison with state-of-the-art methods

To assess the generalization capability of the model, we conducted independent tests on the recognition of TIS and PAS. In particular, we conducted 10-time independent tests for TIS and PAS prediction on each species. For each signal (TIS and PAS) dataset of the four species, we randomly split it into the training and independent test datasets during each test and averaged the performance of 10 tests. We also compared the performance of DeepGenGreap and several state-of-the-art methods for GSRs prediction, including DeepGSR, iTIS-PseTNC (Chen *et al.*, 2014), TITER (Zhang *et al.*, 2017), DPS (Kalkatawi *et al.*, 2013), HMM-SVM (Xie *et al.*, 2013) and Omni-polyA (Magana-Mora *et al.*, 2017). Amongst these compared methods, DeepGSR is the only one developed for the recognition of TIS and PAS in different species. iTIS-PseTNC and TITER are designed specifically for the prediction of TIS. DPS, HMM-SVM and Omni-polyA are PAS-specific prediction methods. The performance results of these methods were collected from the publication (Kalkatawi *et al.*, 2019). We conducted 10-fold cross-validation tests to evaluate the performance of DeepGenGreap for recognition of the splice sites and compared its performance with that of several state-of-the-art methods, including DeepGSR, SpliceMachine (Degroeve *et al.*, 2005), SpliceRover (Zuallaert *et al.*, 2018) and DBN (Lee and Yoon, 2015).

3.2.1 Performance comparison on the TIS recognition

Figure 4A [DeepGSR-TIS (ATG) motif versus DeepGenGrep-TIS (ATG) motif] shows independent test results of DeepGenGrep and DeepGSR for the recognition of TIS in four species in terms of *Acc*. We can see that DeepGenGrep clearly outperformed DeepGSR on all the four test datasets with an *Acc* of 97.35% versus 94.32% on *H.sapiens*, 97.43% versus 94.34% on *M.musculus*, 96.24% versus 92.67% on *B.taurus*, 97.05% versus 93.16% on *D.melanogaster*, respectively. Moreover, we also compared DeepGenGrep with other TIS-specific predictors in *H.sapiens*, including iTIS-PseTNC (Chen *et al.*, 2014) and TITER (Zhang *et al.*, 2017), and evaluated the performance improvement of DeepGenGrep using the reduction of the relative error rate (Kalkatawi *et al.*, 2019). The performance results are provided in Supplementary Table S10. We can see that DeepGenGrep has significantly reduced the error rate by up to 93.7%.

3.2.2 Performance comparison on the PAS recognition

The 'AATAAA' variant is the most common signal among 16 PAS variants ('motif') for each species (Kalkatawi *et al.*, 2019). The number of 'AATAAA' variants was more than half of the total sample number of 16 PAS variants (Supplementary Table S1). We conducted the same experiments on the 'AATAAA' variant as DeepGSR. The performance comparison results between DeepGenGrep and DeepGSR are shown in Figure 4A [DeepGSR-PAS (AATAAA) motif versus DeepGenGrep-PAS (AATAAA) motif]. The results demonstrate that DeepGenGrep achieved a better performance than DeepGSR across three species, with an *Acc* of 85.49% versus 84.29% on *M.musculus*, 84.66% versus 83.21% on *B.taurus*, 89.58% versus 87.23% on *D.melanogaster*, respectively.

Deep learning models should generally be trained on large training datasets; however, the numbers of samples of the other 15 PAS variants were relatively small (statistical summary of the other 15 PAS variants is provided in the Supplementary Table S1), which makes it difficult to train a robust and reliable model for each PAS variant. Therefore, for each species, we integrated all PAS variants to perform the independent test. The performance comparison results in terms of *Acc* between DeepGenGrep and DeepGSR are shown in Figure 4A [DeepGSR-PAS (all) motif versus DeepGenGrep-PAS (all) motif]. The results demonstrate that DeepGenGrep outperformed DeepGSR on each species with an *Acc* of 86.39% versus 83.71% on *H.sapiens*, 85.57% versus 82.79% on *M.musculus*, 84.96% versus 83.44% on *B.taurus*, 89.82% versus 87.94% on *D.melanogaster*, respectively.

Furthermore, we compared the performance of the DeepGenGrep-with other PAS predictors for the recognition of PAS (AATAAA) in *H.sapiens*, including DPS (Kalkatawi *et al.*, 2013), HMM-SVM (Xie *et al.*, 2013) and Omni-polyA (Magana-Mora *et al.*, 2017). The performance comparison results in terms of the relative error rate reduction are provided in Supplementary Table S11. As can be seen, DeepGenGrep reduced the error rate by up to 29%.

3.2.3 Performance comparison on the splice site recognition

Next, we compared the performance of DeepGenGrep with three state-of-the-art splice site predictors, including SpliceMachine (Degroeve *et al.*, 2005), SpliceRover (Zuallaert *et al.*, 2018) and DBN (Lee and Yoon, 2015) on the same dataset. To make a fair comparison, we performed 10-fold cross-validation, which is the same performance evaluation strategy used by the compared tools. As the datasets were relatively imbalanced (with the ratio of 1:5 between the positive and negative samples), we used the median *F1-scores* for performance comparison on the 10-fold cross-validation, and kept the same ratio of 1:5 between the positive and negative samples in each fold subset as the original dataset. In total, 24 (24 separate datasets) $\times$10 (10-fold) = 240 tests were performed on the acceptor/donor datasets, and the median *F1-score* of these 240 results was used for performance comparison. Figure 4B shows the performance results in terms of *F1-score* for DeepGenGrep and the other three predictors. The results show that DeepGenGrep achieved 3.6% to 16.8% and 4.3% to 18.0% higher F-score than the other methods on the GWH-donor and GWH-acceptor datasets. The

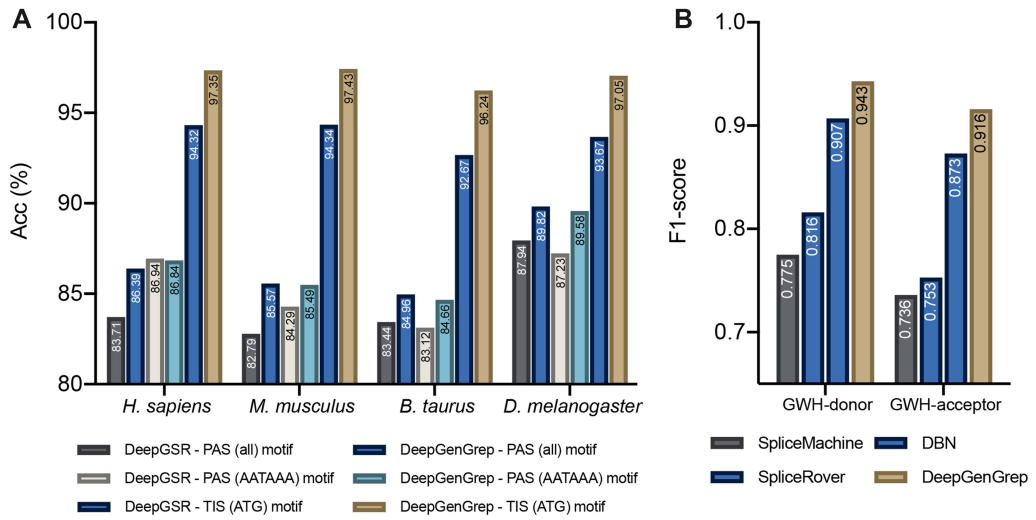

Fig. 4. Performance comparison between DeepGenGrep and DeepGSR for predicting TIS, PAS and splice site on the independent test datasets: (**A**) performance comparison for predicting TIS, PAS 'AATAAA' variant, and all PAS variants of each species. (**B**) Performance comparison for splice site prediction on both GWH donor and GWH acceptor

performance results of DBN and SpliceMachine were collected from the literature (Lee and Yoon, 2015), and the results of SpliceRover were obtained from its published literature (Zuallaert *et al.*, 2018). In addition, we also calculated the AUC and AUPRC of DeepGenGrep on the acceptor and donor datasets considering that they were imbalanced (ratio of 1:5 between the positive and negative samples). As a result, DeepGenGrep achieved AUC of 0.987 and 0.993, AUPRC of 0.966 and 0.982 on the acceptor and donor datasets, respectively.

Moreover, we also compared the performance of DeepGenGrep with DeepGSR by conducting the 10-fold cross-validation test on each chromosome of the human genome. The distributions of *F1-scores*, *Sp*, *Sn* and *Acc* of DeepGenGrep and DeepGSR in 240 executed tests on the acceptor and donor splice sites are plotted in Supplementary Figures S17 and S18, respectively. We can see that DeepGenGrep outperformed DeepGSR in terms of the medians of each evaluation metric for both splice sites, e.g. 0.943 versus 0.901 in terms of *F1-score* for the donor splice site, 0.916 versus 0.866 for the acceptor splice site, respectively. In addition, DeepGenGrep also had fewer outliers than DeepGSR for each performance metric on both acceptor and donor splice sites. Further statistical analysis on 240 tests (refer to Supplementary Table S12 for more detail) shows DeepGenGrep achieved smaller standard deviations for all the performance metrics than DeepGSR. These results indicate that DeepGenGrep has a more robust performance than DeepGSR.

Upon a closer inspection at the results, we identified ten outliers with *F1-score* <0.4 for SpliceRover (Zuallaert *et al.*, 2018) on the splice sites in the Y chromosome dataset, which had the smallest size of positive samples. DeepGSR had three outliers (*F1-score* equal to zero) on the acceptor splice site dataset and four outliers (*F1-score* equal to zero) on the donor splice site dataset of the Y chromosome. In contrast, DeepGenGrep had no outliers in terms of *F1-score* on each splice site of the Y chromosome dataset. Moreover, DeepGenGrep achieved a higher median *F1-score* than DeepGSR on both splice sites, e.g. 0.903 versus 0.837 for the acceptor splice site, 0.946 versus 0.827 for the donor splice site, respectively. Statistical analysis on 10 tests shows DeepGenGrep achieved small standard deviations than DeepGSR in terms of *F1-scores* (refer to the Supplementary Table S13 for more detail). The results indicated that DeepGenGrep was a more robust model than state-of-the-art tools, even for relatively small-size datasets.

3.2. 4 Cross-species conservation validation

To evaluate the transfer learning capability (He *et al.*, 2019) of DeepGenGrep across different species, we further performed cross-species conservation tests on the same experimental settings as DeepGSR. We employed 10 models randomly trained on *H.sapiens* data to predict the corresponding signals in the other three species. The average accuracies of 10 predictions are reported. The performance comparison results of DeepGenGrep and DeepGSR for different GSRs are provided in Supplementary Tables S14–S18. For TIS, we used the DeepGenGrep model trained on the *H.sapiens* TIS data (i.e. 10 models described in Section 3.2.1) to predict TIS in the other three species. The performance comparison results of DeepGenGrep and DeepGSR are accordingly provided in Supplementary Table S14. The results show that DeepGenGrep achieved a better transfer learning performance and capability than DeepGSR to predict TIS in the other three species based on the training using *H.sapiens* data.

For PAS, we used the DeepGenGrep model trained on the PAS 'AATAAA' motif of the human genome (i.e. 10 models trained in Section 3.2.2) to recognize the AATAAA variant and all mixed PAS variants in other species, respectively. Performance comparison results are shown in Supplementary Tables S15 and S16, respectively. Similarly, we used the model trained on all mixed PAS variants in the human genome to recognize AATAAA variant (Supplementary Table S17) and all mixed PAS variants in other species (Supplementary Table S18), respectively. From Supplementary Tables S15–S18, we can see that similar results were obtained for the cross-species PAS recognition.

3.2.5 Performance evaluation of different methods for the prediction of other GSRs

The experiments showed that DeepGenGrep achieved comparative performance on predicting TIS, PAS and splice sites. In this section, to examine if DeepGenGrep is a good general learning framework for multiple GSRs, we further evaluated its prediction performance of promoters, which are an important type of GSRs located in the proximity of the transcription start sites that define where the transcription of a gene begins. The majority of existing promoter prediction methods are species-specific and type-specific (with and without the TATA-box) (Zhu *et al.*, 2021). Here, we compared the predictive performance of DeepGenGrep for promoters with one of the most recently developed promoter predictors, Depicter (Zhu *et al.*, 2021). To enable a fair comparison, we applied the same training and independent test datasets (the *H.sapiens* TATA and non-TATA promoter datasets) used in Depicter to conduct the performance comparison experiments. The training dataset contained 25 548 promoter sequences (22 914 without ATAT-box and 2634 with ATAT-box, respectively), while the independent test dataset encompassed 2839 promoter sequences (2546 non-ATAT-box and

293 ATAT-box, respectively). The same number of negative samples were collected from the exon sequences of *H.sapiens*. We conducted five times 5-fold cross-validation and independent tests to compare the predictive performance of DeepGenGrep with that of Depicter. The performance comparison results on the 5-fold CV and independent test are provided in Supplementary Tables S19 and S20, respectively. We can see that DeepGenGrep outperformed Depicter in terms of all the six major performance metrics (i.e. *Acc*, *MCC*, *F1-score*, *Precision*, *Sp* and *Sn*) on both 5-fold CV and independent tests. These results demonstrate that DeepGenGrep represents a powerful deep learning architecture for the improved prediction of multiple GSRs.

3.3 Webserver implementation

As an implementation of the DeepGenGrep methodology, a user-friendly webserver has been developed and made publicly available at http://bigdata.biocie.cn/deepgengrep/home to facilitate the community-wide efforts to identify different types of GSRs. The webserver is implemented using the SpringBoot framework, Java Server Pages, bootstrap and JavaScript techniques. It is deployed in the Tomcat 7 container and configured in a Linux server with a 4-core CPU, 16 GB memory and a 500 GB hard disk. To utilize the webserver, users need to choose the tab page of a specific type of genomic signal, upload the DNA sequences of interest in the FASTA format, and then click the 'Submit' button to make the prediction. Once the submitted job is completed, the prediction result will be returned to the web page. Alternatively, users can enquire, retrieve and download the prediction results of their submissions from the job list page.

4. Conclusion

In this study, we have developed a general deep learning framework, termed DeepGenGrep, to identify multiple different types of eukaryotic GSRs from DNA sequences. DeepGenGrep equips CNN blocks consisting of multiple parallel 1D-CNNs with different kernel sizes to capture features in various scales. The feature representation capacity of DeepGenGrep was enhanced by stacking three CNN blocks and two layers of LSTMs. Ablation studies showed that the DeepGenGrep framework provides reasonable performance for the recognition of GSRs. Visualization of feature representations illustrates that DeepGenGrep can effectively learn feature representations to identify GSRs. Extensive benchmarking and comparative analysis showed that DeepGenGrep achieved a better performance than several state-of-the-art predictors for predicting PAS, TIS and splice sites. Moreover, cross-species prediction from *H.sapiens* to other three species show that DeepGenGrep achieved a better transfer learning performance and capability than DeepGSR, a state-of-the-art general predictor of multiple GSRs. DeepGenGrep is anticipated to be explored by the wider research community as a valuable tool for the high-throughput and cost-effective identification of putative GSRs in eukaryotic genomes.

Funding

This work was supported by the National Natural Science Foundation of China [61972322] and a Major Inter-Disciplinary Research Project awarded by Monash University.

Conflict of Interest: none declared.

Data availability statement

The data underlying this article are available at https://github.com/wx-cie/DeepGenGrep/tree/main/Data.

References

Albaradei,S. *et al.* (2020) Splice2Deep: an ensemble of deep convolutional neural networks for improved splice site prediction in genomic DNA. *Gene*, **763S**, 100035.

Arefeen,A. *et al.* (2019) DeepPASTA: deep neural network based polyadenylation site analysis. *Bioinformatics*, **35**, 4577–4585.

Bajic,V. *et al.* (2002) Artificial neural networks based systems for recognition of genomic signals and regions: a review. *Informatica*, **26**, 389–400.

Boyd,J.C. *et al.* (2020) Domain-invariant features for mechanism of action prediction in a multi-cell-line drug screen. *Bioinformatics*, **36**, 1607–1613.

Chen,W. *et al.* (2014) iTIS-PseTNC: a sequence-based predictor for identifying translation initiation site in human genes using pseudo trinucleotide composition. *Anal. Biochem.*, **462**, 76–83.

Chen,Z. *et al.* (2020) iLearn: an integrated platform and meta-learner for feature engineering, machine-learning analysis and modeling of DNA, RNA and protein sequence data. *Brief. Bioinform.*, **21**, 1047–1057.

Chen,Z. *et al.* (2021) iLearnPlus: a comprehensive and automated machine-learning platform for nucleic acid and protein sequence analysis, prediction and visualization. *Nucleic Acids Res.*, **49**, e60.

Chollet,F. (2018) *Deep Learning with Python*. Manning, New York, pp. 103.

Degroeve,S. *et al.* (2005) SpliceMachine: predicting splice sites from high-dimensional local context representations. *Bioinformatics*, **21**, 1332–1338.

Down,T.A. and Hubbard,T.J.P. (2002) Computational detection and location of transcription start sites in mammalian genomic DNA. *Genome Res.*, **12**, 458–461.

Eraslan,G. *et al.* (2019) Deep learning: new computational modelling techniques for genomics. *Nat. Rev. Genet.*, **20**, 389–403.

Gao,T. *et al.* (2010) Identifying translation initiation sites in prokaryotes using support vector machine. *J. Theor. Biol.*, **262**, 644–649.

Georgakilas,G. *et al.* (2014) microTSS: accurate microRNA transcription start site identification reveals a significant number of divergent pri-miRNAs. *Nat. Commun.*, **5**, 5700.

Georgakilas,G.K. *et al.* (2020) Solving the transcription start site identification problem with ADAPT-CAGE: a machine learning algorithm for the analysis of CAGE data. *Sci. Rep.*, **10**, 877.

Hartwell,L. *et al.* (2011) *Genetics: From Genes to Genomes*. The McGraw-Hill Companies, Inc., New York, NY.

He,W. *et al.* (2019) 4mCPred: machine learning methods for DNA N-4-methylcytosine sites prediction. *Bioinformatics*, **35**, 593–601.

Ioffe,S. and Szegedy,C. (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift. *Proc. 32nd Int. Conf. Mach. Learn.*, **37**, 448–456.

Ji,G.L. *et al.* (2018) TSAPA: identification of tissue-specific alternative polyadenylation sites in plants. *Bioinformatics*, **34**, 2123–2125.

Kalkatawi,M. *et al.* (2013) Dragon PolyA spotter: predictor of poly(A) motifs within human genomic DNA sequences (vol 28, pg 127, 2012). *Bioinformatics*, **29**, 1484.

Kalkatawi,M. *et al.* (2019) DeepGSR: an optimized deep-learning structure for the recognition of genomic signals and regions. *Bioinformatics*, **35**, 1125–1132.

Kingma,D.P. and Welling,M. (2013) Auto-Encoding Variational Bayes. *arXiv*: 1312.6114, doi:10.48550/arXiv.1312.6114.

Lee,T. and Yoon,S. (2015) Boosted categorical restricted Boltzmann machine for computational prediction of splice junctions. *Proc. 32nd Int. Conf. Mach. Learn.*, **37**, 2483–2492.

Li,F. *et al.* (2020) DeepCleave: a deep learning predictor for caspase and matrix metalloprotease substrates and cleavage sites. *Bioinformatics*, **36**, 1057–1065.

Li,F. *et al.* (2021a) Computational prediction and interpretation of both general and specific types of promoters in *Escherichia coli* by exploiting a stacked ensemble-learning framework. *Brief. Bioinform.*, **22**, 2126–2140.

Li,Z. *et al.* (2021b) DeeReCT-APA: prediction of alternative polyadenylation site usage through deep learning. *Genomics Proteomics Bioinform.*, doi: 10.1016/j.gpb.2020.05.004.

Lin,E. *et al.* (2020) A deep adversarial variational autoencoder model for dimensionality reduction in single-cell RNA sequencing analysis. *BMC Bioinformatics*, **21**, 64.

Liu,B.Q. *et al.* (2019) Computational prediction of sigma-54 promoters in bacterial genomes by integrating motif finding and machine learning strategies. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, **16**, 1211–1218.

Liu,Q. *et al.* (2021) DeepTorrent: a deep learning-based approach for predicting DNA N4-methylcytosine sites. *Brief. Bioinform.*, **22**, bbaa124.

Long Vo,N. *et al.* (2020) Identification of the human DPR core promoter element using machine learning. *Nature*, **585**, 459.
Macosko,E.Z. *et al.* (2015) Highly parallel genome-wide expression profiling of individual cells using nanoliter droplets. *Cell*, **161**, 1202–1214.
Magana-Mora,A. *et al.* (2017) Omni-PolyA: a method and tool for accurate recognition of poly(A) signals in human genomic. *BMC Genomics*, **18**, 620.
Mcinnes,L. *et al.* (2018) UMAP: uniform manifold approximation and projection for dimension reduction. *J. Open Source Softw.*, **3**, 861.
Mort,M. *et al.* (2014) MutPred splice: machine learning-based prediction of exonic variants that disrupt splicing. *Genome Biol.*, **15**, R19.
Morton,T. *et al.* (2015) TIPR: transcription initiation pattern recognition on a genome scale. *Bioinformatics*, **31**, 3725–3732.
Pedregosa,F. *et al.* (2011) Scikit-learn: machine learning in Python. *J. Mach. Learn. Res.*, **12**, 2825–2830.
Rahman,M.S. *et al.* (2019) iPromoter-FSEn: identification of bacterial sigma(70) promoter sequences using feature subspace based ensemble classifier. *Genomics*, **111**, 1160–1166.
Ren,S.Q. *et al.* (2017) Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, **39**, 1137–1149.
Snoek,J. *et al.* (2012) Practical Bayesian optimization of machine learning algorithms. *Adv. Neural Inf. Process. Syst.*, **25**, 2960–2968.
Sosa,R.P. *et al.* (2020) Interactions of upstream and downstream promoter regions with RNA polymerase are energetically coupled and a target of regulation in transcription initiation. *bioRxiv*, doi:10.1101/2020.05.13.070375.
Sterck,L. *et al.* (2012) ORCAE: online resource for community annotation of eukaryotes. *Nat. Methods*, **9**, 1041.
Szegedy,C. *et al.* (2015) Going deeper with convolutions. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, Boston, MA, pp. 1–9.
Umarov,R. *et al.* (2019) Promoter analysis and prediction in the human genome using sequence-based deep learning models. *Bioinformatics*, **35**, 2730–2737.
Umarov,R. *et al.* (2021) ReFeaFi: genome-wide prediction of regulatory elements driving transcription initiation. *PLoS Comput. Biol.*, **17**, e1009376.
van der Maaten,L. and Hinton,G. (2008) Visualizing data using t-SNE. *J. Mach. Learn. Res.*, **9**, 2579–2605.
Wang,R.H. *et al.* (2019) SpliceFinder: ab initio prediction of splice sites using convolutional neural network. *BMC Bioinformatics*, **20**, 561.
Xia,Z. *et al.* (2019) DeeReCT-PolyA: a robust and generic deep learning method for PAS identification. *Bioinformatics*, **35**, 2371–2379.
Xie,B. *et al.* (2013) Poly(A) motif prediction using spectral latent features from human DNA sequences. *Bioinformatics*, **29**, i316–i325.
Yu,H. and Dai,Z. (2020) SANPolyA: a deep learning method for identifying poly(A) signals. *Bioinformatics*, **36**, 2393–2400.
Zhang,X.H.F. *et al.* (2003) Sequence information for the splicing of human pre-mRNA identified by support vector machine classification. *Genome Res.*, **13**, 2637–2650.
Zhang,S. *et al.* (2017) TITER: predicting translation initiation sites by deep learning. *Bioinformatics*, **33**, I234–I242.
Zhang,M. *et al.* (2019) MULTiPly: a novel multi-layer predictor for discovering general and specific types of promoters. *Bioinformatics*, **35**, 2957–2965.
Zhang,M. *et al.* (2022) Critical assessment of computational tools for prokaryotic and eukaryotic promoter prediction. *Brief. Bioinform.*, **23**, bbab551.
Zhu,Y. *et al.* (2021) Computational identification of eukaryotic promoters based on cascaded deep capsule neural networks. *Brief. Bioinform.*, **22**, bbaa299.
Zuallaert,J. *et al.* (2017) Interpretable convolutional neural networks for effective translation initiation site prediction. In: Hu,X.H. *et al.* (eds.) *2017 IEEE International Conference on Bioinformatics and Biomedicine*. IEEE, Kansas City, MO, pp. 1233–1237.
Zuallaert,J. *et al.* (2018) SpliceRover: interpretable convolutional neural networks for improved splice site prediction. *Bioinformatics*, **34**, 4180–4188.