Inference of splicing motifs through visualization of recurrent networks

Aparajita Dutta,[†,§] Aman Dalmia,[‡,§] Athul R,[‡] Kusum Kumari Singh,[¶] and Ashish Anand[*,†]

†*Department of CSE, Indian Institute of Technology, Guwahati, India*

‡*Department of EEE, Indian Institute of Technology, Guwahati, India*

¶*Department of BSBE, Indian Institute of Technology, Guwahati, India*

§*Equal contribution*

E-mail: anand.ashish@iitg.ac.in

Abstract

Neural models have been able to obtain state-of-the-art performances on several genome sequence-based prediction tasks. Often such models take only nucleotide sequences as input and learn relevant features on its own. However, extracting the interpretable motifs from the model remains a challenge. This work explores four existing visualization techniques in their ability to infer relevant sequence patterns learned by a recurrent neural network (RNN) on the task of splice junction identification. The visualization techniques have been modulated to suit the genome sequences as input. The visualizations inspect genomic regions at the level of a single nucleotide as well as a span of consecutive nucleotides. This inspection is performed from the perspective of the overall model as well as individual sequences. We infer canonical and non-canonical splicing motifs from a single neural model. We also propose a cumulative scoring function that ranks the combination of significant regions across the sequences involved

in non-canonical splicing event. Results indicate that the visualization technique giving preference to k-mer patterns can extract known splicing motifs better than the techniques focusing on a single nucleotide.

Introduction

The recent trend in genome sequence analysis is the application of neural network models that learn features from the sequence *de-novo*[1,2]. The primary motivation to let the model learn relevant features by itself is to avoid the existing knowledge bias. Several deep and shallow neural network models have been successfully applied on genome sequence-based tasks, such as, identification of transcription factor binding sites[3], microRNA target prediction[4], and prediction of DNA methylation states[5]. However, the inference of significant features or genomic regions from the learned models remains a challenge. In the domain of computer vision, image processing, and natural language processing (NLP), several visualization techniques have been effectively applied to analyze learned models and infer relevant features. Our work aims to apply visualization techniques to identify the significant regions of the genome sequence, in the form of *sequence patterns or motifs*, that contribute to the prediction performance. Towards this aim, we also ask the following two principal questions- *Q1: How can various visualization techniques be adapted for identifying significant genomic regions for a particular task? Q2: Do all visualization techniques deliver similar results or are one method superior to the other?*

We employ four different visualization techniques by modulating them to suit the genome sequences as input. The characteristics of the chosen visualization methods can be broadly classified into two categories based on their ways of identifying motifs. One set of the visualization techniques identifies motifs considering all the sequences present in the dataset, whereas the other set identifies motifs based on individual sequences. The four visualization techniques are as follows:

1. The *attention mechanism* to obtain significant genomic regions captured by the overall learning model.

2. *Smooth gradient* of noisy nucleotide embeddings to access the impact incurred on the classification score by a small change in any nucleotide of a given sequence.

3. *Omission of a single nucleotide* to access the importance of each nucleotide present in a genome sequence.

4. *Occlusion of k-mers* to derive fixed as well as variable length sequence patterns from a genome sequence.

Among the visualization techniques, the attention mechanism falls in the first category whereas the rest three fall in the second category.

Towards our aim, we consider the task of splice junction classification and evaluate the different visualization techniques in their ability to infer known splicing motifs. Splice junction classification is an important sub-task of genome annotation. This task involves identification of exon-intron (donor site) and intron-exon (acceptor site) boundaries which are usually characterized by canonical motif dimers GT and AG respectively (further explained in Supplementary Materials (Section 1.1)). However, there are exceptions to these consensus motifs which yield the non-canonical motifs that correspond to non-canonical or unconventional splicing events[6]. Most of the existing computational methods focus only on the identification of canonical splice junctions due to the lack of consistent non-canonical consensus. Nevertheless, the non-canonical sequence patterns are equally important in understanding the splicing phenomenon[6], and hence this remains an interesting area to be investigated further. In summary, our additional objective is to extract relevant canonical and non-canonical splicing motifs from the same model.

Motivated by application of RNN in sequence-based bioinformatics problems[4,7,8], we have further explored its application in splice junction prediction by employing bidirectional long short-term memory (BLSTM) units[9] in the hidden layer. Since a BLSTM network

reads the context from both the ends, incorporating bidirectionality is expected to boost the performance of prediction models[1]. We further superimposed the model with an attention[10] layer to add interpretability to the model.

The contributions of this paper can be summarized as follows:

- We explore the application of BLSTM network along with attention mechanism for the prediction of splice junctions. The proposed architecture achieves the state-of-the-art performance.
- We generated two different types of negative dataset to test the consistency of the various recurrent neural models. Comprehensive analysis of splicing motifs is performed with the proposed architecture as it is the most consistent in its performance with both the dataset.
- We have redesigned some of the effective visualization techniques, available in the literature, to be capable of comprehending genome sequences as inputs.
- We infer splicing motifs for both canonical and non-canonical splicing events. The canonical patterns are validated with the existing knowledge from literature.
- We propose a scoring function, named *cumulative deviation score*, to rank the most significant combinations of sequence patterns present across the sequences belonging to non-canonical splicing events.

Related work

Visualization of sequence motifs

With the objective of deciphering the reason behind the outstanding performance of the various learning models, various attempts have been made to monitor the change in model weights as learning progresses[4,11].[5] extracted sequence motifs by aligning sequence fragments

that maximally activated the filters of the convolutional layer for predicting single-cell methylation states.[12] performed one-dimensional global average pooling on the attention weighted output of an RNN to discover the parts of the sequence that are significant for identifying pre-miRNAs.

Lanchantin et al. in[13] explored various sequence-specific as well as class-specific visualization techniques to obtain the important nucleotide positions present in a genome sequence for classification of transcription factor binding sites. Based on a similar objective, we have incorporated variable length occlusion to obtain variable length canonical as well as non-canonical splicing motifs apart from accessing the importance of each nucleotide position using attention and omission scores. We have also used the smooth gradient of noisy nucleotide embeddings, rather than raw gradients, to generate sharper *sensitivity maps*[14].

[15] predicted splice sites using a deep convolutional neural network (CNN) having five convolutional layers. They have also identified significant genomic regions based on a visualization technique *DeepLIFT*[16] which assigns a contribution score to each nucleotide in a sequence based on its significance on the predicted result. However, their model predicts either the donor or the acceptor splice sites based on the dataset used. Also, they have considered only canonical splice junctions for visualizing the important genomic regions.

Splice junction classification

Several computational methods have been proposed for splice junction classification. In recent times, advanced sequencing technologies like RNA-seq have produced a plethora of sequenced genome. The abundance of annotated data has boosted both alignment based and machine learning based methodologies for predicting splice junctions. Alignment based methods identify splice junctions via map-assemble strategy where numerous short reads are first mapped to a reference genome after which they are assembled to identify distinct clusters representing exonic regions[17]. However, there is a possibility of a short read randomly matching a large reference genome containing multiple occurrences of the short read

sequence[18]. Also, the existing alignment based methods[17,19] consider only canonical splice junctions in the prediction task[11].

The traditional machine learning based splice junction predictors use hand-crafted features like presence or absence of specific nucleotide patterns around the splice junctions[20–22]. Since all the splicing signals are still not known, the hand-engineered features may adversely affect the accuracy of prediction models due to the inclusion of irrelevant features as well as high dimensionality. There have been attempts at handling the issue of high dimensionality by optimizing the features using feature selection techniques[23,24]. Nonetheless, limited biological knowledge still ensued the inclusion of irrelevant features and hence revealed the necessity of applying learning techniques that can capture, by itself, the complex splicing signals in the form of features from the genome sequence.

Lee et al. proposed a splice junction prediction model based on deep Boltzmann machine[11]. Zhang et al. employed a deep CNN, named *DeepSplice*[25] that predicts novel splice junctions. Lee et al. have explored different RNN units in the hidden layers of a deep network for predicting splice junctions[7]. Dutta et al. proposed distributed feature representations of splice junctions, named *SpliceVec*[26], which captures splicing features to be classified by a multilayer perceptron (MLP). These prediction models yielded promising accuracy in the prediction of splice junctions. However, most of these models fail to extract the sequence motifs that govern the splicing phenomenon due to lack of interpretability in the model.

Methods

In this section, we introduce the neural architecture employed for classification of true and decoy splice junctions. Further, we discuss the visualization techniques applied to analyze the patterns learned by the model.

Neural architecture

The proposed architecture is shown in Figure 1. The entire workflow is explained in the following subsections.

Input representation

Input is a putative splice junction sequence consisting of five nucleotide codes, *A (Adenine), C (Cytosine), G (Guanine), T (Thymine) and N (representing any one of the four nucleotides)*, where each of the nucleotide code is represented as an integer. We use a dense vector representation for each of the five nucleotide codes. Each input sequence is passed through an embedding layer which transforms each input splice junction sequence of length n into an $n \times 4$ dimensional dense vector that gets updated while training the neural network.

Modeling splice junctions using BLSTM network

The $n \times 4$ dimensional dense input vectors are fed in mini-batches into both the forward and backward LSTM layers configured as a BLSTM (details on LSTM and BLSTM provided in

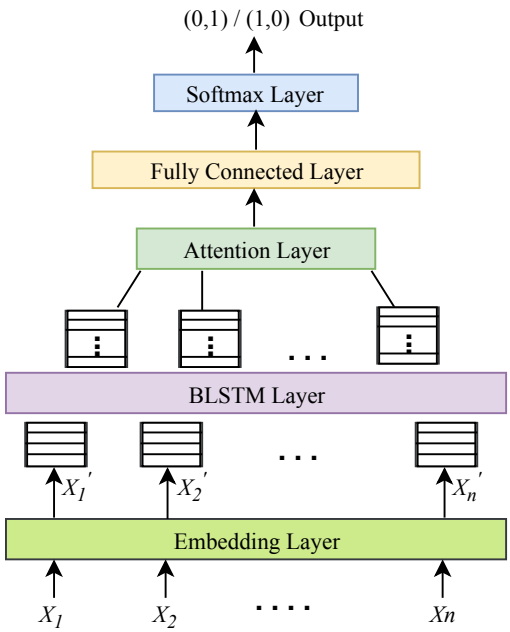


Figure 1: An overview of the proposed architecture.

Supplementary Materials (Section 1.2)). Both the LSTM layers learn meaningful features in a supervised manner to generate an $n_l \times n$ dimensional vector representation, where n_l is the number of hidden units in each LSTM layer. Both the vectors generated by the forward and backward LSTM layers are concatenated to generate a $2n_l \times n$ dimensional vector representing the learned features of each splice junction.

Feature interpretation using attention layer

This layer is added to the model for obtaining a more targeted model which can capture the role each nucleotide in the input sequence plays in the classification of the splice junction. The attention mechanism is explained in Supplementary Materials (Section 1.2). The $2n_l \times n$ representation of each sequence obtained from the BLSTM network is next fed into the attention layer. The attention weights obtained from this layer are fed into a fully connected layer and eventually to a softmax layer to obtain the classification results. We have used binary cross-entropy and Adam[27] as the loss function and the optimizer respectively.

Visualization techniques

The visualization techniques rely on measuring the change in performance of the learned model effected by changing either the input sequence or the embedding space of the model for a genomic region. The change can be implemented at a single nucleotide or a span of consecutive nucleotides. In some sense, these techniques mimic the site-specific mutagenesis techniques performed in a wet lab setup. The visualization techniques are applied to define a scoring function, referred to as the *deviation value*, which reflects the contribution of genomic regions to the classification score. We have inferred splicing motifs from the significant regions identified by the visualization techniques based on the *deviation value*. The various visualization techniques employed are described in the following subsections.

Smooth gradient with noisy embeddings

In image classification tasks, the gradient of the unnormalized output probabilities with respect to the input image, referred to as *sensitivity maps*[14], indicate how much can a tiny change in that pixel affect the final output. In the case of images, minutely changing the pixel values does not change the image significantly as the image still looks the same[28]. Whereas genome sequences, comprising of discrete nucleotides, can significantly alter the underlying biology on replacing a nucleotide by another.

Thus, to incur a slight change, we add noise to the embeddings of the nucleotides and compute the change in classification score. However, the sensitivity maps resulting from raw gradients are usually noisy. Therefore, based on the concept of smooth gradient[28], we average out the gradients obtained from several different noisy embeddings for each position of a sequence. The average gradient at each sequence position, named *smooth gradient*, is the *deviation value* in this case. As a result, one might expect that the resulting averaged sensitivity map would crisply highlight the key regions.

Omission of a single nucleotide

The feature vector obtained from the fully connected layer of the proposed architecture represents the complete input sequence. To measure the significance of each sequence position, we calculate its *omission score*[29]. The omission score of the jth position p_j^i in a sequence s_i is given by

$$omission(p_j^i, s_i) = 1 - cosine(V_{s_i}, V_{s_i \setminus p_j^i}) \tag{1}$$

where V_{s_i} is the feature representation obtained from the fully connected layer for the sequence s_i and $V_{s_i \setminus p_j^i}$ is the feature representation obtained from the same layer for the same sequence with the nucleotide at position p_j^i replaced by N. $Cosine(V_{s_i}, V_{s_i \setminus p_j^i})$ measures the similarity of the two vectors and is calculated as

$$cosine(V_{s_i}, V_{s_i \setminus p_j^i}) = \frac{V_{s_i} \cdot V_{s_i \setminus p_j^i}}{\|V_{s_i}\| \|V_{s_i \setminus p_j^i}\|} \tag{2}$$

Therefore, omission score measures the *deviation value* of the vector representations of the sequence, with and without the omitted nucleotide. Higher deviation implies a higher significance of that sequence position.

Occlusion of k-mers

We occluded portions of a sequence to observe the variation in the predicted output. This approach has its motivation from.[30] For each sequence, we run a sliding window w_l of length l [1], centered at nucleotide number $(l+1)/2$, and replace the characters within the particular window with N. We pass the modified sequence through the model to obtain the *deviation value* given by the absolute difference of the model outputs with and without the occlusion. For occlusion of a window of length l, the *deviation value* is stored in the center, that is, in position $(l+1)/2$, of the window.

We generated *deviation values* for test sequences in batches. For a batch of size B, the *deviation values* are in the form of a matrix of size $B \times n$ where each input sequence is of length n. The naive implementation (iterative) takes much time. But after some modifications (explained in Supplementary Materials (Section 2.1)), a batch implementation is performed which reduces the computation time significantly.

We propose two variations of occlusion described as follows:

Fixed length occlusion: In this case, we have considered occlusion of only 3-mers to compute the *deviation value* at each sequence position. *Deviation values* at boundary indices are computed by occluding the first two or the last two indices only. The *deviation value* $dev_3^{i,j}$ at index j of sequence s_i is assigned to the middle index of the 3-mer. The significance

[1] We take window of length l with $(l-1)/2$ nucleotides to the left and $(l-1)/2$ nucleotides to the right of the center nucleotide.

of a genomic region is proportional to the corresponding *deviation value*.

Variable length occlusion: Fixed length occlusion has a limitation of considering only fixed length genomic regions whereas in real scenario there may be sequence patterns of variable lengths that regulate splicing. Hence, we incorporate variable length occlusion where for each index j of a sequence s_i, we have occluded a window w_l of length $l \in \{3, 5, 7, 9, 11\}$ and computed the *deviation values* denoted by $dev_l^{i,j}$, for each window length l, with and without occlusion. The *deviation value* assigned to position $(l+1)/2$ of window w_l is given by $max_l(dev_l^{i,j})$ for $l \in \{3, 5, 7, 9, 11\}$. The window length $argmax_l(dev_l^{i,j})$ corresponding to index j of sequence s_i is stored in the jth column of the ith row of a *window matrix*. Therefore, the value at the jth column of the ith row of the *window matrix* signifies the length of the pattern, centered at index j of sequence s_i, that contributes maximum to the prediction of the model.

Prerequisites for visualization

For each of the visualization techniques, we obtain 15 and 7 sets of randomly selected 50 true and 50 decoy splice junctions from canonical and non-canonical test sequences respectively. Going forward, the splice junction sequences comprising the canonical dimer motifs ($GT - AG$) are referred to as the canonical sequences and likewise for the non-canonical sequences. We execute the following steps to obtain the splicing motifs.

Generating heat maps

For each of the visualization techniques, we generate heat maps for each set. The heat maps are generated from the corresponding matrix of *deviation values*, named *deviation matrix*. Further, each set is sampled to obtain a final heat map comprising of random sequences from all the sets. Each sequence in the heat map is 164 nucleotide (nt) long. The first 82 nt of a sequence are the upstream and downstream regions of the donor site whereas the next 82

nt are the upstream and downstream regions of the acceptor site of a junction pair (dataset explained in the next section). The heat map consists of true test sequences only.

Identifying the number of significant indices in a genomic region

To reveal the splicing motifs using the visualization techniques, we need to first identify the number of sequence indices which can be considered influential in splice junction identification. For this, we arrange the indices of a sequence in non-increasing order of weights in the *deviation matrix*. Next, we obtain the frequency of occurrence of each index in the top T positions among all the sequences present in the sets, depicted by the *occurrence-frequency graph*. In other words, we consider the first T positions of the ordered *deviation matrix* for all the sequences across all the sets and count the number of times each index occurs in the first T positions. Here, the term 'position' is used relative to the length of the *deviation matrix* whereas 'index' refers to the absolute position based on the indexing explained later.

Based on the occurrence-frequency graph, we select a *significant number* $'K'$ which indicates the number of sequence indices that play a significant role in the predicted output of the model. To quantify the value of K, we consider it as the number of indices whose frequency of occurrence among the top T positions is more than 25% of the total number of sequences across all the sets.

Generating splicing motifs

In a particular set, the top K indices across all the sequences are calculated as follows. For each sequence s_i where $i \in \{1, 2, ..., 50\}$, we select the top K indices $p^i_{j'}$, in non-increasing order of *deviation values*. Therefore, we obtained a $50 \times K$ *position matrix* from which we select the top K indices p_k with maximum weighted occurrences across all the sequences in the set. The weighted occurrence of each index p_k is calculated as the sum of the weights of each occurrence of that index across all the sequences in the set. Here, the highest weight 'K' is added if the index occurs in the first column of the position matrix and the lowest

weight '1' is added if the index occurs in the last (K^{th}) column of the position matrix. This can be represented using the formula

$$p_k = \sum_{i=1}^{50} \sum_{j'=1}^{K} [p^i_{j'} = k], \forall k \in \{1, 2, ..., 164\} \tag{3}$$

$$where, [p^i_{j'} = k] = \begin{cases} K - j' + 1, & \text{if } p^i_{j'} = k \text{ is true} \\ 0, & \text{otherwise} \end{cases}$$

On obtaining the top K indices of each set, we select the overall top K indices across all the sets for canonical sequences, and across all the sets for non-canonical sequences. This is again performed using Equation (3). The splicing motifs are generated for these top K indices across all the sequences present in all the sets.

Experimental setup

Positive data generation

We use GENCODE annotations[31], based on human genome assembly version *GRCh38*, to train and test our model. We target to access the model's performance on the prediction of novel splice junctions. For this, we train the model using an earlier release and test the model on only the newly added splice junctions in a later release, as described in[25]. The two versions used are version 20 (released in August 2014) and version 26 (released in March 2017).

We extract 291,831 and 294,576 unique splice junction pairs from version 20 and version 26 respectively. The junction pairs are extracted from protein-coding genes only. Each junction pair comprises an intron with flanking upstream and downstream exonic regions. The intronic length is observed to vary from 1 to 1,240,200 nt. A previous study found that the shortest known eukaryotic intron is 30 base pairs (bp) long belonging to the human

MST1L gene[32]. Introns shorter than 30 bp usually result from sequencing errors in genome. Therefore, we consider introns of length greater than 30 bp only. This reduces the number of junction pairs to 290,502 and 293,889 in versions 20 and 26 respectively. We exclude, from version 26, all the junction pairs present in version 20. This leaves us with 5,612 novel junction pairs in version 26, which composes our test data.

Negative data generation

Existing works adopt one of the two techniques to generate negative data. We adopt both the ways of generating the negative data to have comprehensive and unbiased analysis. The Type-1 dataset is generated similar to the standard procedure described in.[33] We extract a portion of the sequence from the center of each true intron to generate a pseudo sequence. The length of the extracted portion is kept equal to the length of an input sequence. This set of negative data captures the non-randomness of DNA sequences. We obtain 290,502 false samples for training data and 5,612 false samples for testing data using this procedure.

More than 98% of splice junctions contain the consensus dimer GT and AG at the donor and acceptor junctions respectively[34]. However, the occurrence of the consensus dinucleotide is far more frequent compared to the number of true splice junctions in the genome. The neat exclusion of the pseudo sites by the splicing mechanism suggests the presence of other subtle splicing signals which play an important role in the process. The presence of consensus dimer in all the negative samples will result in the model learning the remaining splicing patterns present in the vicinity of the splice junctions.

Therefore, we generate the Type-2 dataset based on the procedure described in[25,26] where the negative data is randomly sampled from the human genome assembly version *GRCh38*. For each decoy junction pair, we randomly search for the consensus dimer GT and AG such that both lie in the same chromosome and the distance between them lies in the range of 30 and 1,240,200 nt. We obtain a huge number of such samples using this procedure, out of which we randomly select 290,502 false samples for training data and 5,612 false samples

for testing data. In both the dataset, the positive junction pairs are the same. Both the scenarios are pictorially depicted in Figure 4 of Supplementary Materials (Section 2.2).

Training and Hyperparameter tuning

Each input splice junction is truncated to 40 nt upstream and downstream flanking regions of the consensus dimer GT or AG, thus obtaining an 82 nt sequence. Both the donor and acceptor junctions of a junction pair are concatenated to form a 164 nt sequence. The effect of variation in the flanking region on model accuracy is shown in Table 1 of Supplementary Materials (Section 2.3).

The proposed architecture can be represented as a (1-4-100-100-2048-2) architecture where 4-dimensional embeddings are passed through a BLSTM, attention and fully connected layer with 100, 100 and 2048 units respectively. Values for batch size, dropout, recurrent dropout, and epochs are tuned to 128, 0.5, 0.2 and 50 respectively. The hyperparameters are tuned by partitioning the training data from version 20 into 90% training and 10% validation data. All experiments were carried out on an NVIDIA GeForce GTX 980 Ti GPU machine with 6GB memory. We evaluate the performance of the classifier based on precision, recall, accuracy, and F1 score. Table 2 in Supplementary Materials (Section 2.4) shows the variation in the performance of the model with variation in the number of hidden layers.

Baselines

We implemented the following state-of-the-art models as baselines and compared the results obtained by the various models on the same set of training and testing data. The hyperparameters for the baselines (details in Supplementary Materials (Section 2.5)) were tuned using the same process mentioned in the previous subsection.

1. DeepSplice: This model classifies input sequences using a deep CNN comprising of

two convolutional layers[25]. Input sequences are represented in the form of a 4×30 matrix comprising of the upstream and downstream flanking regions of both acceptor and donor splice junctions.

2. SpliceVec-MLP: This model learns feature vectors of the entire intronic region along with flanking upstream and downstream exonic regions to be classified using an MLP[26]. The input formation for SpliceVec-MLP is described in Supplementary Materials (Section 2.6).

3. Vanilla LSTM: This model comprises an embedding layer, two hidden LSTM layers, and a softmax output layer as proposed in[7].

4. Vanilla LSTM with attention: We replaced the hidden units of the proposed architecture with LSTM units.

Existing consensus for analyzing visualizations

We compare the obtained visualization results with the existing consensus of canonical sequences. Figure 2 shows the extended donor site consensus 9-mer $[AC]AGGTRAGT$ and the extended acceptor site consensus 15-mer $Y_{10}NCAGG$, known from existing studies[35]. The acceptor site consensus mostly consists of the polypyrimidine tract (PY-tract) Y_{10}. Our dataset comprises 164 nt long junction pairs where the first 82 nt represents the donor region, and the next 82 nt represents the acceptor region. Both the donor and acceptor regions comprise the consensus dimer with 40 nt upstream and 40 nt downstream regions each. Both

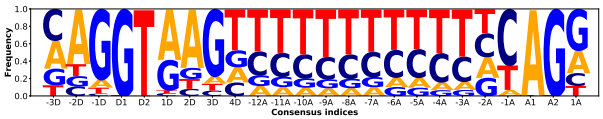


Figure 2: **Existing consensus in the canonical positive sequences of the dataset.** The consensus comprises the extended donor site consensus 9-mer $[AC]AGGTRAGT$ ranging from indices -3D to 4D and the extended acceptor site consensus 15-mer $Y_{10}NCAGG$ ranging from indices -12A to 1A in a 164 nt long junction pair.

the consensus dimers GT and AG, at donor and acceptor sites respectively, are indexed as $D1D2$ and $A1A2$. The upstream flanking region of the donor site is indexed -1D through -40D starting from the closest to the farthest of the consensus dimer. Similarly, the downstream flanking region of the donor site is indexed 1D through 40D starting from the position closest to the consensus dimer. Indexing is similar for acceptor site flanking regions with only D replaced by A.

Results

Quantitative analysis of prediction performance

We evaluate the performance of the proposed architecture using both Type-1 and Type-2 dataset. We compare the predictive performance with the baselines described in the previous section.

Table 1: Performance of the proposed architecture compared with the state-of-the-art models. Accuracy (Ac), Precision (Pr), Recall (Re) and F1 Score (F1) are computed in percentage.

Model	Type-1 Dataset	Type-2 Dataset
	$Ac, Pr, Re, F1(\%)$	$Ac, Pr, Re, F1(\%)$
Proposed architecture	98.24, 99.66, 96.81, 98.21	98.38, 99.38, 97.38, 98.37
LSTM	98.19, 99.45, 96.91, 98.16	98.36, 99.38, 97.34, 98.35
LSTM with attention	98.24, 99.70, 96.77, 98.21	97.68, 98.88, 96.45, 97.65
SpliceVec-MLP	71.57, 71.42, 72.28, 71.72	93.98, 96.43, 91.40, 93.78
DeepSplice	89.81, 94.77, 84.43, 89.05	92.85, 96.43, 89.03, 92.56

Table 1 shows the comparison of the proposed architecture with the baselines. We observe a performance improvement of the proposed architecture, over other neural network based baselines, in the range of 8%-26% for the Type-1 dataset and an improvement in the range of 4%-5% for the Type-2 dataset. The performances of other variants of RNN are comparable to the proposed architecture. The RNN based models are consistent in its performance with

both the dataset. We analyze the performance of the proposed architecture on the Type-2 dataset.

Qualitative analysis of the feature representation

To access the quality of the feature embedding obtained from the dense layer of the proposed architecture, we plot the 2048-dimensional vector by reducing it to a two-dimensional vector using Stochastic Neighbor Embedding (t-SNE)[36]. Although the dimension reduction procedure involves some loss of information, we still observe that the projected representations are distinctly separable in the two-dimensional feature space. Figure 3 shows a t-SNE plot of randomly selected 1000 true and 1000 decoy splice junctions. Four other plots of randomly selected true and decoy splice junctions are shown in Figure 5 of Supplementary Materials (Section 3.1). [2]

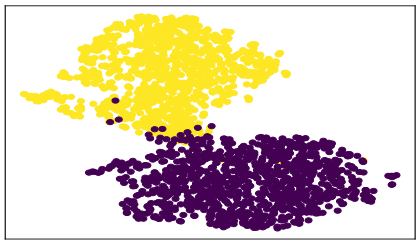


Figure 3: t-SNE plot of 1000 true and 1000 decoy splice junctions. Points in blue represent true whereas points in yellow represent decoy splice junctions.

Visualization techniques

For each of the visualization techniques, we generate 15 sets of randomly selected 50 true and 50 decoy splice junctions from 10853 canonical test sequences and 7 sets from 371 non-canonical test sequences. Each of the canonical and non-canonical sets is further sampled to obtain the final canonical and non-canonical sets for generating heat maps. The final

[2]We have plotted 1000 sequences considering the execution time required for larger sample sizes.

canonical set consists of 75 sequences, sampled at 5 sequences from each of the 15 sets whereas the final non-canonical set consists of 70 sequences, sampled at 10 sequences from each of the 7 sets. For each of the visualization techniques, we generate heat maps of the *deviation matrices* obtained from each set as well as the final sampled set. The final heat map for the various visualization techniques is shown in Figure 6 and Figure 7 of Supplementary Materials (Section 3.2) for canonical and non-canonical sequences respectively.

Interpretation of attention weights

Donor site consensus captured well for canonical sequences: To obtain the significant number K, we plot the occurrence-frequency graph (Figure 4) of top 20 indices in non-increasing order of attention weights. The value of K is considered as 10 in this case as 10 indices have frequency more than 25% of the 750 canonical sequences. Figure 5(a) shows the splicing motif obtained for all the 750 canonical true splice junction pairs based on the top 10 indices across all the sets. We observe that the attention model recognizes a few upstream and downstream indices of the donor site consensus and only the first base of the acceptor site consensus dimer.

Non-canonical splicing motif obtained is minutely different from canonical: The sequence pattern obtained with K = 10 (Figure 8 of Supplementary Materials (Section 3.3)) across all the non-canonical sets is shown in Figure 5(b). Here, the significant indices ob-

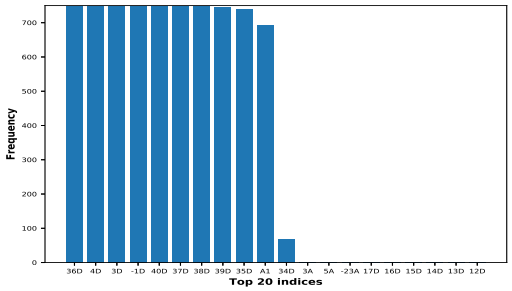


Figure 4: Occurrence-frequency graph based on the attention weights of top 20 indices among all the canonical true sequences present in the 15 sets.

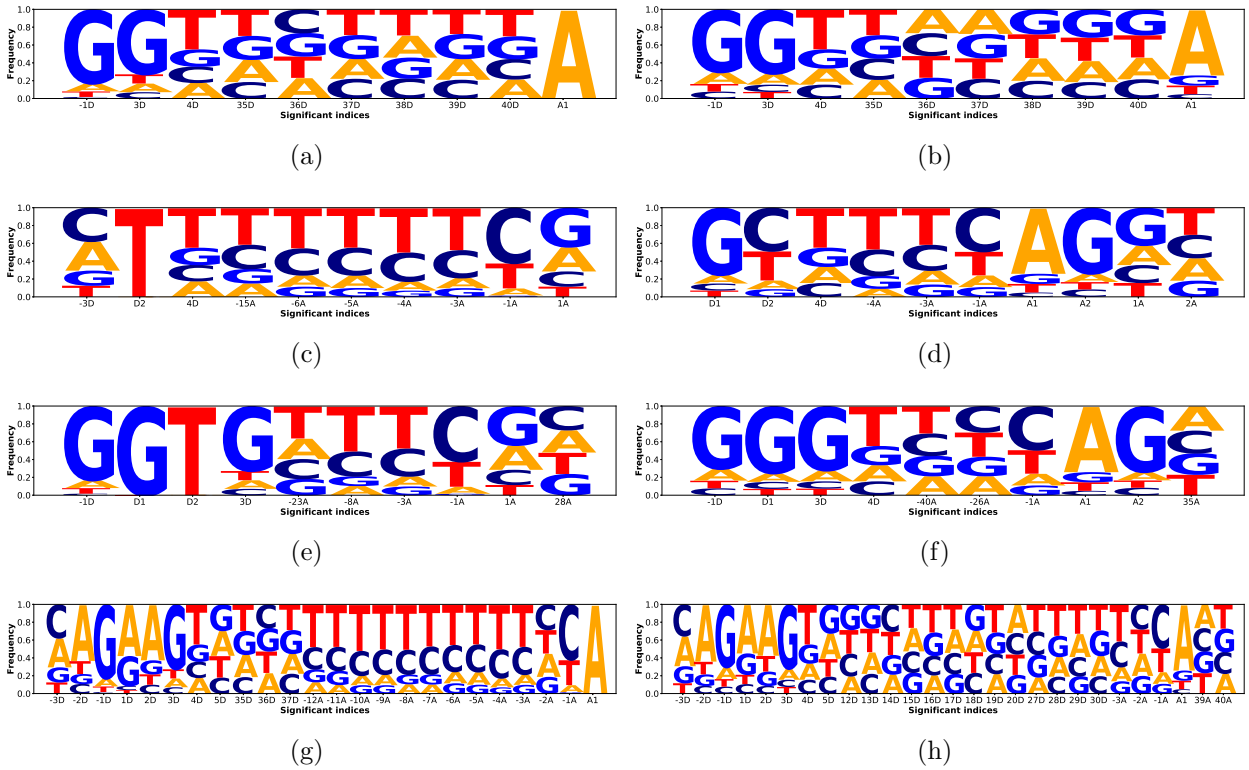


Figure 5: **Splicing motifs of the top K indices obtained from the occurrence-frequency graph of various visualization techniques.** The motifs on the left are obtained from the canonical sequences whereas the motifs on the right are obtained from the non-canonical sequences. The indices along the x-axis are arranged in increasing order of its appearance in the input sequences. The frequency plotted along the y-axis is the relative frequency of a nucleotide with respect to other nucleotides at a given index. The motifs are generated from (a) and (b) Attention weights with K = 10 (c) and (d) Smooth gradients with K = 10 (e) and (f) Omission scores with K = 10 (g) and (h) 3-mer occlusion weights with K = 15.

tained for both canonical and non-canonical sequences are the same because the model is trained irrespective of the canonical and non-canonical sequences. In the sequence pattern, we see a minutely increasing frequency of G and T in the downstream of donor site around indices 38D to 40D for both canonical and non-canonical sequences. This can be the possible reason for higher attention weights in spite of no known consensus in this region. The attention mechanism captures features considering the overall model and not the individual sequences. As a result, it misses out the subtle patterns and captures only the most dominating consensus around the donor and the acceptor sites.

Smooth gradient with noisy embeddings

Mostly captures the PY-tract along with partial donor site consensus in canonical sequences: For each input sequence, we generate 50 samples of embeddings by adding Gaussian noise with a standard deviation of 0.15. We average the gradient of the classification score with respect to each of the 50 embeddings for each sequence position. This generates the gradient map, commonly known as the sensitivity map. We consider K = 10 for uniformity although the value obtained from the occurrence-frequency graph (Figure 9 of Supplementary Materials (Section 3.3)) is much less. The sequence pattern obtained with K = 10 is shown in Figure 5(c). It mostly captures the PY-tract along with few other upstream and downstream consensus at the donor and acceptor sites.

Most frequent alternate consensus captured: Figure 5(d) shows the sequence pattern considering K = 10 (obtained from Figure 10 of Supplementary Materials (Section 3.3)) for the non-canonical sequences. The pattern captures the weak PY-tract in the upstream of the acceptor site. Interestingly, it captures $GC - AG$ as the most frequent alternative consensus for non-canonical splice junctions which is consistent with the literature[37].

Omission of a single nucleotide

Donor site consensus dimer GT captured in canonical sequences: We obtain the corresponding sequence pattern with K = 10 (obtained from Figure 11 of Supplementary Materials (Section 3.3)) as shown in Figure 5(e). We observe that this pattern captures the donor site consensus dimer GT which was not revealed by the previous techniques. This technique also reveals a few other indices in the vicinity of the consensus dimer. It partially captures the PY-tract in the upstream of the acceptor site.

Weak PY-tract captured upstream of the acceptor site of non-canonical sequences: The corresponding sequence pattern with K = 10 (Figure 12 of Supplementary

Materials (Section 3.3)) is shown in Figure 5(f). The pattern captures a dominant G at indices -1D and 3D around the donor site. It also captures a weak PY-tract at the upstream of the acceptor site at indices -40A, -26A and -1A which is also a validation of the displacement of the PY-tract in case of non-canonical sequences[6]. The consensus dimer AG is captured as the dominant acceptor site consensus.

Occlusion of k-mers: fixed-length occlusion

Both donor and acceptor site extended consensus captured in canonical sequences: We plot the extended 3-mer window with K = 15 (Figure 13 of Supplementary Materials (Section 3.3)) to generate the splicing motif as shown in Figure 5(g). For consecutive indices, the overlapping regions are merged to obtain a continuous extended region. We observe that the sequence indices in the pattern cover almost the complete extended donor site 9-mer and most of the extended acceptor site 15-mer, particularly the PY-tract in the upstream of the acceptor site. Due to its window length, occlusion is capable of capturing the maximum information in comparison to the other techniques.

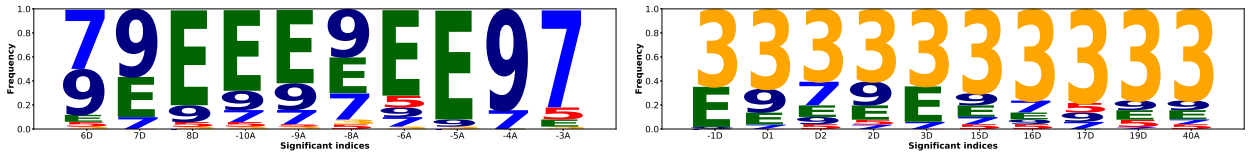
Donor site consensus similar in canonical and non-canonical sequences: We generate the splicing motif (Figure 5(h)) considering K = 15 (Figure 14 of Supplementary Materials (Section 3.3)). We observe the pattern $[AC]AGRAG[GT]R$ at indices [-3D to -1D] and [1D to 5D]. The model also captures the presence of Y at indices [-3A to -1A] as well as the presence of A at index A1. The indices [12D to 20D], extracted as significant, do not display any pattern whereas [27D to 30D] reveals a very weak increase in T. These two regions can be interesting for further experimental verification.

Although occlusion covers most of the important splice patterns, we still cannot conclude it to be the best technique for visualization of genomic sequences, because the other techniques also capture few crucial signals that were missed by occlusion.

Occlusion of k-mers: variable length occlusion

9-mers and 11-mers influence canonical splicing: We generate the pattern (Figure 6(a)) of the *window matrix* for the top 10 indices across all the 15 sets of canonical sequences. The window lengths are arranged in non-increasing order of frequency from top to bottom at each index of the pattern. Symbol $'E'$ stands for a window length of 11 nt. We can infer that, in case of canonical sequences, the model displays maximum deviations in case of 9-mer occlusion near the donor site and 11-mer occlusion near the PY-tract Y_{10}. Moreover, the window length decreases from 11-mer to 7-mer as it approaches the acceptor site.

3-mers influence non-canonical splicing: The pattern of the *window matrix* for the non-canonical sequences across all the 7 sets is shown in Figure 6(b). We observe that the most frequent window length across the top 10 indices is 3 which suggests that most of the non-canonical regulatory patterns, present across these indices, are 3-mers.



(a) (b)

Figure 6: **Pattern generated for frequencies of various window lengths at top 10 indices.** The top 10 indices are based on deviation scores obtained by considering variable length occlusions for (a) 750 canonical sequences from 15 sets and (b) 350 non-canonical sequences from 7 sets. The indices in x-axis are arranged in increasing order of its appearance in the input sequences. The frequency in y-axis is the relative frequency of a window length with respect to other window lengths at a given index. Symbol 'E' stands for window length 11 nt.

Cumulative scoring function for non-canonical patterns

The non-canonical sequences usually lack one or more known canonical consensus, or the consensus may be distally located resulting in the existence of novel recognition pathways for non-canonical splicing[6]. These subtle patterns occur in different combinations, and so they are not visible in the sequence patterns.

To overcome this limitation, we formulate a scoring function to capture the combination of sequence patterns that have the maximum impact on the classification score for a particular sequence. Having obtained the *deviation values* for each index of a particular sequence, we choose the top 10 indices in non-increasing order of *deviation values*. We calculate the

Table 2: The top 10 significant indices obtained from the top 5 non-canonical sequences, based on the cumulative scoring function, for each visualization.

Visualization	Indices
Attention weights	-2D 2D 3D 34D 35D 36D 37D 38D 39D -1A
	-2D 2D 3D 34D 35D 36D 37D 38D 39D -1A
	-2D 2D 3D 34D 35D 36D 37D 38D 39D -1A
	-2D 2D 3D 34D 35D 36D 37D 38D 39D -1A
	-2D 2D 3D 34D 35D 36D 37D 38D 39D -1A
Smooth gradients	-39D -35D -23D -20D -16D -5D 17D 24D 32D A2
	-24D D1 1D 10D 13D 26D 35D -22A -17A 1A
	-30D D1 8D 35D -35A -28A -10A -7A -2A A2
	-9D -4D D1 D2 19D -40A -21A -12A -9A A2
	-30D -16D -7D D2 5D 39D -17A -1A A2 3A
Omission scores	-23D -11D 2D 19D 40D -34A -15A -7A -1A 4A
	-34D -30D -16D -32A -13A -12A -10A -1A A1 12A
	-20D -1D D2 2D 3D 19D 26D -31A -19A 13A
	-31D -22D -21D -3D -31A -19A A1 14A 28A 30A
	-40D -16D -9D -6D -2D 2D 23D -27A 3A 36A
3-mer occlusion	-2D -1D D1 D2 -21A -19A -18A -16A -15A -1A
	19D 20D 22D 26D 27D 28D 29D -1A A1 A2
	7D 8D 9D 10D 11D 12D 13D 14D 12A 13A
	-2D -1D D1 D2 21D 34D -32A -31A -30A -26A
	-13A -12A -11A -10A -9A -8A -5A -4A -3A -2A

cumulative deviation score (D_{cum}) as the summation of these top 10 *deviation values* per sequence. For each visualization technique, we choose the top 5 non-canonical sequences with the maximum D_{cum} across the sequences of all the 7 sets. Table 2 shows the significant indices obtained from top 5 sequences for each visualization. The corresponding sequence patterns are shown in Table 3 of Supplementary Materials (Section 3.3).

The top 5 non-canonical sequence patterns obtained from attention, occlusion, and smooth gradient mostly comprise indices in the vicinity of the donor site. Only the omission scores have a balance between the number of donor site and acceptor site indices. This again suggests that the sequence pattern in the donor site plays a more important role in non-canonical splicing compared to the acceptor site sequence pattern. Also, we observe that most of the donor site significant indices are in the intronic region (downstream of the donor site).

Conclusion

In this work, we explore the application of BLSTM network along with attention mechanism for the prediction of splice junctions. The proposed architecture achieves the state-of-the-art performance. We redesigned some of the effective visualization techniques, from the domain of computer vision, NLP and image classification, to be capable of comprehending genome sequences for inferring canonical and non-canonical splicing motifs. We validate the splicing motifs inferred from the canonical sequences by comparing it with the existing consensus known from literature. We also obtain non-canonical splicing motifs detected by the visualization techniques along with some regions in the sequence that can be experimentally investigated. Further, we propose a scoring function (cumulative deviation score) to rank the most significant combinations of sequence patterns present across the sequences belonging to non-canonical splicing event.

References

(1) Li, Y.; Chen, C.-y.; Kaye, A. M.; Wasserman, W. W. *Biosystems* **2015**, *138*, 6–17.

(2) Min, S.; Lee, B.; Yoon, S. *Briefings in bioinformatics* **2017**, *18*, 851–869.

(3) Alipanahi, B.; Delong, A.; Weirauch, M. T.; Frey, B. J. *Nature biotechnology* **2015**, *33*, 831.

(4) Lee, B.; Baek, J.; Park, S.; Yoon, S. deepTarget: end-to-end learning framework for microRNA target prediction using deep recurrent neural networks. Proceedings of the 7th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics. 2016; pp 434–442.

(5) Angermueller, C.; Lee, H. J.; Reik, W.; Stegle, O. *Genome biology* **2017**, *18*, 67.

(6) Murray, J. I.; Voelker, R. B.; Henscheid, K. L.; Warf, M. B.; Berglund, J. A. *Genome biology* **2008**, *9*, R97.

(7) Lee, B.; Lee, T.; Na, B.; Yoon, S. *arXiv preprint arXiv:1512.05135* **2015**,

(8) Hill, S. T.; Kuintzle, R.; Teegarden, A.; Merrill III, E.; Danaee, P.; Hendrix, D. A. *Nucleic acids research* **2018**, *46*, 8105–8113.

(9) Graves, A.; Schmidhuber, J. *Neural Networks* **2005**, *18*, 602–610.

(10) Bahdanau, D.; Cho, K.; Bengio, Y. *ICLR* **2015**,

(11) Lee, T.; Yoon, S. Boosted categorical restricted Boltzmann machine for computational prediction of splice junctions. International Conference on Machine Learning. 2015; pp 2483–2492.

(12) Park, S.; Min, S.; Choi, H.-S.; Yoon, S. Deep recurrent neural network-based identification of precursor micrornas. Advances in Neural Information Processing Systems. 2017; pp 2891–2900.

(13) Lanchantin, J.; Singh, R.; Wang, B.; Qi, Y. Deep motif dashboard: Visualizing and understanding genomic sequences using deep neural networks. PACIFIC SYMPOSIUM ON BIOCOMPUTING 2017. 2017; pp 254–265.

(14) Baehrens, D.; Schroeter, T.; Harmeling, S.; Kawanabe, M.; Hansen, K.; MÃžller, K.-R. *Journal of Machine Learning Research* **2010**, *11*, 1803–1831.

(15) Zuallaert, J.; Godin, F.; Kim, M.; Soete, A.; Saeys, Y.; De Neve, W.; Hancock, J. *Bioinformatics* **2018**,

(16) Shrikumar, A.; Greenside, P.; Kundaje, A. Learning Important Features Through Propagating Activation Differences. International Conference on Machine Learning. 2017; pp 3145–3153.

(17) Au, K. F.; Jiang, H.; Lin, L.; Xing, Y.; Wong, W. H. *Nucleic acids research* **2010**, *38*, 4570–4578.

(18) Li, B.; Dewey, C. N. *BMC bioinformatics* **2011**, *12*, 323.

(19) Trapnell, C.; Pachter, L.; Salzberg, S. L. *Bioinformatics* **2009**, *25*, 1105–1111.

(20) Reese, M. G.; Eeckman, F. H.; Kulp, D.; Haussler, D. *Journal of computational biology* **1997**, *4*, 311–323.

(21) Pertea, M.; Lin, X.; Salzberg, S. L. *Nucleic acids research* **2001**, *29*, 1185–1190.

(22) Degroeve, S.; Saeys, Y.; De Baets, B.; Rouzé, P.; Van de Peer, Y. *Bioinformatics* **2004**, *21*, 1332–1338.

(23) Degroeve, S.; De Baets, B.; Van de Peer, Y.; Rouzé, P. *Bioinformatics* **2002**, *18*, S75–S83.

(24) Saeys, Y.; Degroeve, S.; Van de Peer, Y. Digging into acceptor splice site prediction: an iterative feature selection approach. European Conference on Principles of Data Mining and Knowledge Discovery. 2004; pp 386–397.

(25) Zhang, Y.; Liu, X.; MacLeod, J. N.; Liu, J. DeepSplice: Deep classification of novel splice junctions revealed by RNA-seq. Bioinformatics and Biomedicine (BIBM), 2016 IEEE International Conference on. 2016; pp 330–333.

(26) Dutta, A.; Dubey, T.; Singh, K. K.; Anand, A. *Computational biology and chemistry* **2018**, *74*, 434–441.

(27) Kingma, D. P.; Ba, J. *ICLR* **2015**,

(28) Smilkov, D.; Thorat, N.; Kim, B.; Viégas, F.; Wattenberg, M. *arXiv preprint arXiv:1706.03825* **2017**,

(29) Kádár, A.; Chrupała, G.; Alishahi, A. *Computational Linguistics* **2017**, *43*, 761–780.

(30) Zeiler, M. D.; Fergus, R. Visualizing and understanding convolutional networks. European conference on computer vision. 2014; pp 818–833.

(31) others,, et al. *Genome research* **2012**, *22*, 1760–1774.

(32) Piovesan, A.; Caracausi, M.; Ricci, M.; Strippoli, P.; Vitale, L.; Pelleri, M. C. *DNA Research* **2015**, *22*, 495–503.

(33) Noordewier, M. O.; Towell, G. G.; Shavlik, J. W. Training knowledge-based neural networks to recognize genes in DNA sequences. Advances in neural information processing systems. 1991; pp 530–536.

(34) Burset, M.; Seledtsov, I.; Solovyev, V. *Nucleic acids research* **2000**, *28*, 4364–4375.

(35) Zhang, X. H.; Heller, K. A.; Hefter, I.; Leslie, C. S.; Chasin, L. A. *Genome Research* **2003**, *13*, 2637–2650.

(36) Maaten, L. v. d.; Hinton, G. *Journal of machine learning research* **2008**, *9*, 2579–2605.

(37) Mount, S. M. *The American Journal of Human Genetics* **2000**, *67*, 788–792.