bioRxiv preprint doi: https://doi.org/10.1101/255257; this version posted January 29, 2018. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

COSSMO: Predicting Competitive Alternative Splice Site Selection using Deep Learning

Hannes Bretschneider[1,2], Shreshth Gandhi[1], Amit G Deshwar[1,3], Khalid Zuberi[1], and Brendan J Frey[1,2,3]

[1]Deep Genomics Inc., Toronto, ON
[2]Department of Computer Science, University of Toronto
[3]Edward S. Rogers Sr. Department of Electrical & Computer Engineering, University of Toronto

Abstract

Alternative splicing selection is inherently competitive and the probability for a given splice site to be used depend strongly on the strength of neighbouring sites. Here we present a new model named Competitive Splicing Site Model (COSSMO) that improves on the start of the art in predicting splice site selection by explicitely modelling these competitive effects. We model an alternative splicing event as the choice of a 3' acceptor site conditional on a fixed upstream 5' donor site, or the choice of a 5' donor site conditional on a fixed 3' acceptor site. Our model is a custom architecture that uses convolutional layers, communication layers, LSTMS, and residual networks, to learn relevant motifs from sequence alone. COSSMO is able to predict the most frequently used splice site with an accuracy of 70% on unseen test data, which compares to only around 35% accuracy for MaxEntScan.

1 Introduction

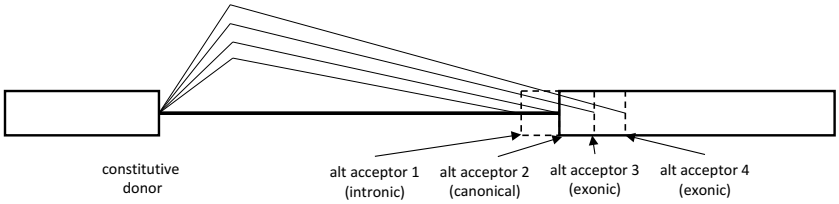
RNA splicing has long been known as a main driver of transcriptional diversity and a large number of regulatory mechanisms have been described. More recently, efforts have shifted from describing isolated regulatory mechanisms to building computational models, known as *splicing codes* [Wang and Burge, 2008], that can accurately predict splice site usage from sequence directly or a library of hand-engineered, sequence-derived features.

The first practical splicing code was introduced by Barash et al. [2010] and predicted tissue differences of cassette splicing events in mouse. Subsequent versions of the splicing code introduced a Bayesian neural network and predicted absolute splicing levels [Xiong et al., 2011]. Since then, Bayesian neural networks [Xiong et al., 2015] and deep neural networks [Leung et al., 2014] capable of predicting absolute splicing levels in human tissues as well as splicing changes caused by variants have improved on the state of the art in predicting exon skipping from sequence context.

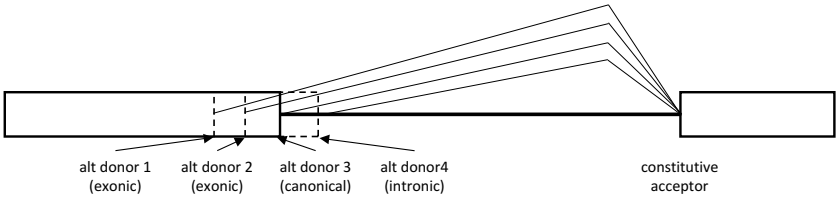
These splicing codes have so far all modelled cassette splicing events. Busch and Hertel [2015] present a model that differentiates between exons that are constitutively spliced, exons that undergo alternative 5' or 3' splice-site selection, and alternative cassette-type exons using support vector machines, but their model does not directly model splicing levels rather than just discriminate between different classes of alternatively spliced exons.

In this paper, we present a new splicing code that we call COSSMO, for *competitive splice site model*. COSSMO is much more general than previous splicing models and is capable of predicting a usage distribution of multiple splice sites conditional on a constitutive opposite site. In particular, we

bioRxiv preprint doi: https://doi.org/10.1101/255257; this version posted January 29, 2018. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.



(a) Alternative acceptor event.



(b) Alternative donor event.

Figure 1: Illustration of alternative acceptor and donor events modeled by COSSMO.

can model the usage distribution of multiple alternative acceptor sites conditional on a constitutive donor site or vice versa.

We train two versions of this model, one for alternative acceptor sites and one for alternative donor sites. Figure 1a shows an example of an alternative acceptor event. In the case of alternative acceptors we always condition on a constitutive donor site. In this example we model four alternative acceptor sites, but the model can dynamically adapt to any number of sites. The COSSMO model will use these alternative sites along with the constitutive site as input and predict a discrete probability distribution over the alternative sites indicating the frequency with which they are selected.

Figure 1b shows an analogous case of alternative donor selection, where multiple donor sites are selected amongst to be spliced to a constitutive acceptor site.

2 Dataset Construction

In this section we describe how to construct a genome-wide dataset of quantified *percent selected index* (PSI) values of alternative splice site events from genome annotations and RNASeq data from the *Genotype-Tissue Expression Project* (GTEx) [GTEx Consortium, 2013]. The PSI value is the frequency with which each splice-site is selected versus all other splice sites in the same event with PSI values for each event/training case summing up to one.

We create two datasets: one of alternative acceptor events conditional on constitutive donor sites and one of alternative donor sites conditional on constitutive acceptor sites.

We mine an initial set of splicing events from genome annotations and then expand it with de-novo splicing events detected from the aligned GTEx RNA-Seq data. We further augment this set with *decoy* splice sites. This is to increase the variability of features seen by the model and help train the model to discriminate between splice sites and non-splice sites.

Each example in the alternative acceptor dataset consists of multiple putative acceptor splice sites and a constitutive donor splice site. In the alternative donor dataset, each example consists of multiple putative donor splice sites and a constitutive acceptor splice site.

Following the construction of these events, we quantify the PSI distribution for each example from RNA-Seq junction reads using the Bayesian bootstrap estimation method by Xiong et al. [2016], allows us to quantify the uncertainty of our PSI values by producing a posterior distribution.

2.1 Genome Annontations

We use the Gencode v19 annotations to construct our initial dataset. We start with a dataset of all exon-exon junctions by creating a training example for each annotated donor site and adding

bioRxiv preprint doi: https://doi.org/10.1101/255257; this version posted January 29, 2018. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

	Acceptor	Donor
Number of events	173,164	169,650
Avg # of annotated sites/event	1.158	1.184
Avg # of *de-novo* sites/event	3.744	3.696
Avg # of soft negative sites/event	61.443	56.612
Avg # of hard negative sites/event	22.451	11.877

Table 1: Dataset statistics

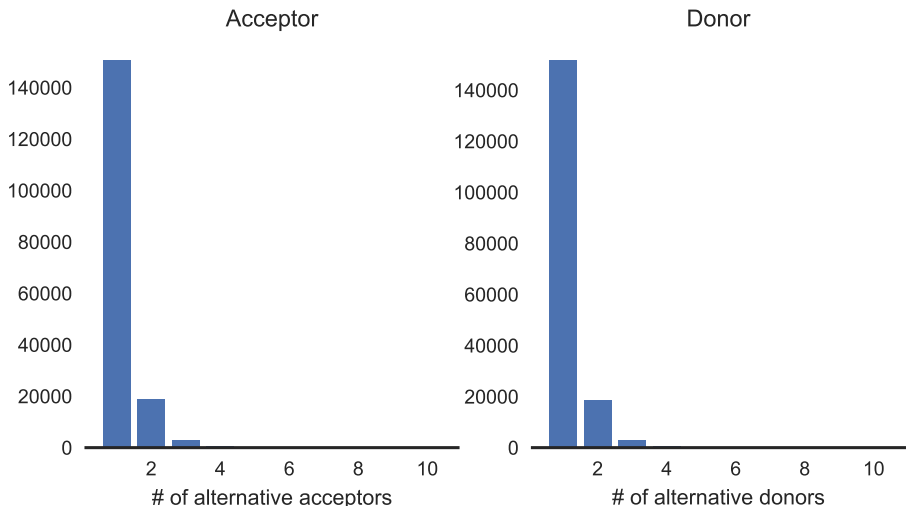


Figure 2: Histogram showing the complexity of *annotated* splicing events. For each annotated donor we count how many acceptors splice to it (left) and for each annotated acceptor we count how many donor sites are spliced to it (right).

all acceptor sites that splice to this donor to the training example as alternative acceptors of type *annotated*.

Then we construct the initial alternative donor dataset analogously by finding the set of donors that splice to each acceptor site.

Figure 2 shows histograms of the number of alternative acceptors per donor and alternative donors per acceptor, respectively. As can be seen from the histogram, we do not only include splice sites that are annotated as alternatively spliced, but also constitutive exons. We do this in order to learn a general model of splice site strength. Summary statistics of the dataset are given in Table 1.

2.2 *De-novo* Splice Sites from Gapped RNA-Seq Alignments

Parsing genome annotations provides us with an initial set of alternative splicing events to form the core of our data set. Next, we add *de-novo* events from the GTEx RNA-Seq data. We start by aligning the GTEx RNASeq reads to the genome using the HISAT2 aligner [Kim et al., 2015]. Next, we iterate through the data set we built in Section 2.1. For the alternative acceptor data set we iterate through all donor sites and for each site, we add all junction reads in GTEx that use this site as one end of a gapped alignment. We then add the other side of the gapped alignment as an alternative acceptor to the example. The procedure for adding *de-novo* splice sites to the alternative donor data set is analogous: we iterate over all acceptor sites, find junction reads that use that acceptor site as one side of a gapped alignment and add the other end of the gap as an alternative donor site.

Some filtering of those *de-novo* sites is necessary both because the RNA splicing process itself is noisy and sequencing and alignment introduce their own biases and artifacts, which would otherwise result in large numbers of low-certainty splice sites. We only utilize *de-novo* sites that are observed in at least two tissues from at least two subjects. This procedure results in a large expansion of

bioRxiv preprint doi: https://doi.org/10.1101/255257; this version posted January 29, 2018. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

possible splice sites, adding an average of 3.74 *de-novo* splice sites to each acceptor event, and 3.70 *de-novo* sites to each donor event (Table 1).

2.3 Negative splice site examples

The above procedure gives us a data set of high-confidence annotated and *de-novo* splice sites. However, we supplement this data set with additional, verified, *non-splice sites*. This is done to increase the amount of variation in the features when training the model. For example, the core di-nucleotides `GU` at the donor site and `AG` at the acceptor site are almost always a necessary feature for splicing to occur and will therefore be present in all examples of real splice sites (apart from examples using the *minor spliceosome*, Turunen et al. [2013]). If we trained a splicing model only on true splice sites, the model might in fact learn to ignore the core di-nucleotide motif since it will not provide any signal to the model. However, if we add some examples of non-splice sites to the data set, then the model has an opportunity to learn that the presence of a core di-nucleotide motif is necessary for recognition of the site. We call examples like this *hard negatives* and we generate them by simply sampling random locations from a region that (when constructing an alternative acceptor dataset) starts 20nt downstream of the donor site and ends 300nt downstream of the most distant alternative acceptor site.

Secondly, the genome also contains a large number of *decoy splice sites*. These are sites that look very similar to real splice sites and in many cases have a core di-nucleotide motif, but are nevertheless not used as splice sites. For example they may lack other necessary features such as a polypyrimidine tract or a branch point, or are adjacent to a silencer motif. These cases are also beneficial to include in our training set because they help the model detect the more subtle signals beyond the consensus sequence that are necessary for a splice site to be actually recognized by the spliceosome. We sample decoy splice sites by using MaxEntScan [Yeo and Burge, 2004] to scan the intron and exon for any sites that have a score greater than 3.0. We then remove from this set all sites that are either annotated as splice sites or if there are any junction reads aligned to them in the GTEx RNA-Seq data. The remaining sites are therefore locations that are assigned a high score by MaxEnt Scan and look very similar to true splice sites but for which there is no evidence that they are ever used as splice sites from annotations or GTEx.

On average each acceptor event contains 22.45 soft negative sites and 61.44 hard negative sites. Each donor event has on average 11.88 soft negative and 56.61 hard negative sites (Table 1).

2.4 PSI Estimation

After building the data sets of alternative acceptors and donors, we need to quantify the frequency with which these sites are used. Our method for estimating PSI values is based on counting the number of junction reads that are aligned to a splice site pair.

Assume, we are interested in a constitutive donor site with multiple alternative acceptor sites. We could obtain a naive PSI estimate by counting the number of junction reads spanning from the donor to each alternative acceptor and normalizing those values to obtain a probablility distribution. In practice, it is well known that RNA-Seq alignments exhibit many biases such as read stacks and other positional biases that make such naive estimates unreliable. Many methods have been developed to ameliorate these issues as well as to obtain variance estimates of PSI values. We use the positional bootstrap method by Xiong et al. [2016]. This method estimates a non-parametric posterior distribution of PSI using a Bayesian positional bootstrap procedure. For the purposes of estimating PSI from the GTEx RNA-Seq data, we pool the reads from all GTEx samples to estimate an average PSI across all subjects and tissues.

3 Model

3.1 Motivation

We call our model COSSMO, short for *Competitive Splice Site Model*. Our primary design goal is to construct a model that is able to dynamically predict the *relative* utilization of any number of competing alternative splice sites. The inputs to the model are DNA and RNA sequences from a window around the alternative splice sites as well as the intron length (distance between acceptor

bioRxiv preprint doi: https://doi.org/10.1101/255257; this version posted January 29, 2018. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

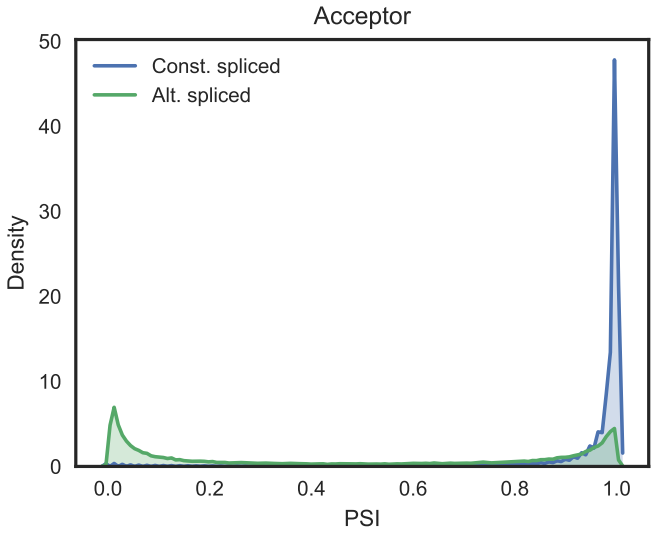


Figure 3: Kernel density estimator plot of PSI values in constitutively versus alternatively spliced events. Constitutively spliced events are defined as having exactly one annotated splice site while alternatively spliced events are all events with two or more annotated sites. Only acceptor dataset is shown.

and donor sites) and the output is PSI (*percent selected index*), or the frequency at which a given splice site is selected. Building this model requires a dynamic architecture that is invariant to the number of splice sites. We achieve this by a network architecture that shares the same weights between all splice sites and uses a *dynamic softmax* function at the output to adjust the size of the output layer to the number of alternative splice sites in the example.

We first present the *acceptor model*, which predicts the PSI of multiple acceptor sites conditional on a constitutive donor site, but the donor model is constructed completely analogously by swapping *acceptor* and *donor*, as well as *3'* and *5'* everywhere.

Given a constitutive donor splice site $const$ and K alternative acceptor splice sites $alt_1, alt_2, \ldots, alt_K$, COSSMO is a function f that predicts the conditional frequency of selecting the i-th splice site,

$$p(alt_i | alt_{k' \neq i}, const) = f_i(alt_{1,\ldots,K}, const), i \in 1, \ldots, K. \tag{1}$$

3.2 Features

We train COSSMO on both DNA and (spliced) mRNA sequence from a window around the splice site and a single numerical feature which is the length of the intron. All sequence features are represented as *one-hot* encoding using four channels that represent the four possible nucleotides.

For each splice site we extract the following sequence inputs:

> *Alternative DNA sequence:* This is the pre-splicing sequence from a 80nt window around the alternative acceptor site.
>
> *Constitutive DNA sequence:* This is the pre-splicing sequence from a 80nt window around the constitutive donor site (and thus it is the same for all splice sites in the same event).
>
> *mRNA sequence:* This is the spliced mRNA sequence obtained by concatenating 40bp of exonic sequence upstream from the donor site and 40nt of exonic sequence downstream from the acceptor site.

To these sequence inputs we add a single feature representing the intron length, obtained by computing the distance between the constitutive donor and alternative acceptor site and normalizing this

bioRxiv preprint doi: https://doi.org/10.1101/255257; this version posted January 29, 2018. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

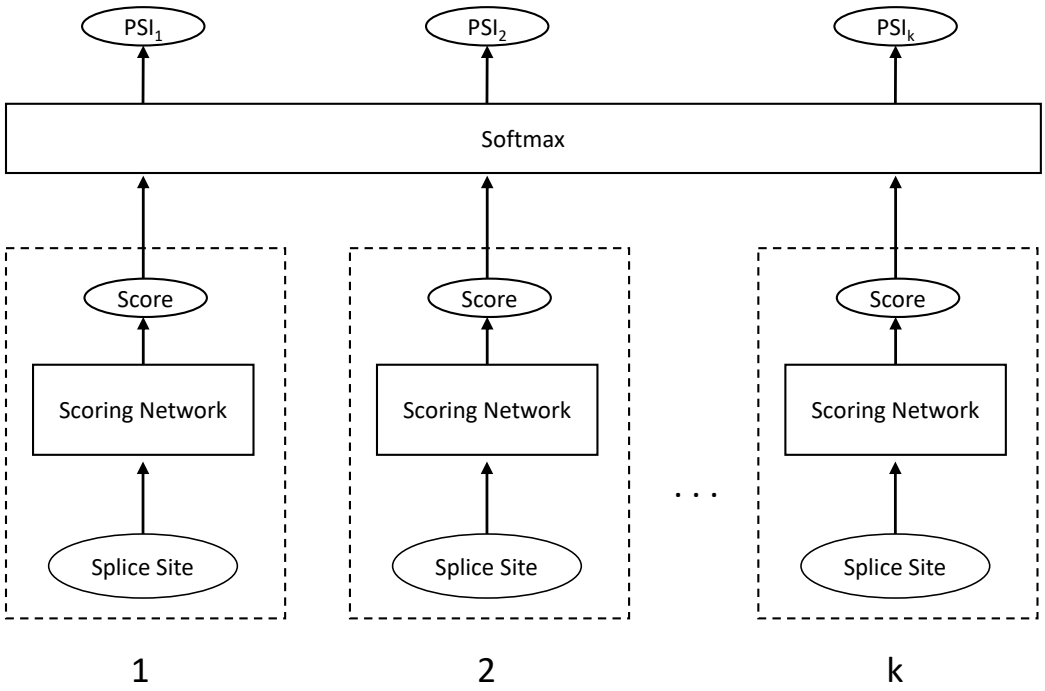


Figure 4: COSSMO Architecture: A score is computed for each splice site separately using identical sub-networks with weight sharing. The scores are then normalized with a softmax layer, allowing the number of splice sites to vary.

distance by the mean and standard deviation of the length of an intron in the human genome [Hong et al., 2006] to make it more numerically stable.

The COSSMO model consists of two primary components: a *scoring network* that produces a scalar unnormalized score for a single splice site and a softmax layer that normalizes scores from multiple scoring networks.

The requirements for the scoring network are that it will take a single splice site's sequence as input and produce a single scalar score. The output layer simply accepts the scalar scores as input, normalizes them, and outputs the predicted PSI distribution over the alternative splice sites. The high-level architecture is illustrated in Figure 4.

While the scoring network's task is to predict a scalar score for each alternative splice site, we also need an output layer that normalizes those scores to obtain a valid probability distribution over the splice sites. The output layer applies a softmax function to the splice site strength score from the scoring network. The softmax layer itself does not contain any variables, so it is compatible with any number of alternative splice sites. Typically, we train and predict on mini-batches which contain examples with different numbers of alternative splice sites. To accomodate such *ragged* arrays we pad rows in the matrix of splice site scores that are shorter than the maximum number of columns with the `MIN_FLOAT` value which ensures that the softmax value of those elements is zero. When training the model, we also ensure that masked out values in the target matrix are set to zero to ensure that the gradients computed for the masked out matrix entries are always zero.

3.3 Scoring Network

As explained above, the scoring network takes inputs derived from one alternative splice site and outputs an unnormalized strength score. Let s_i denote the strength of the i-th alternative splice site such that

$$s_i = S(alt_i, const|\theta), \tag{2}$$

bioRxiv preprint doi: https://doi.org/10.1101/255257; this version posted January 29, 2018. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

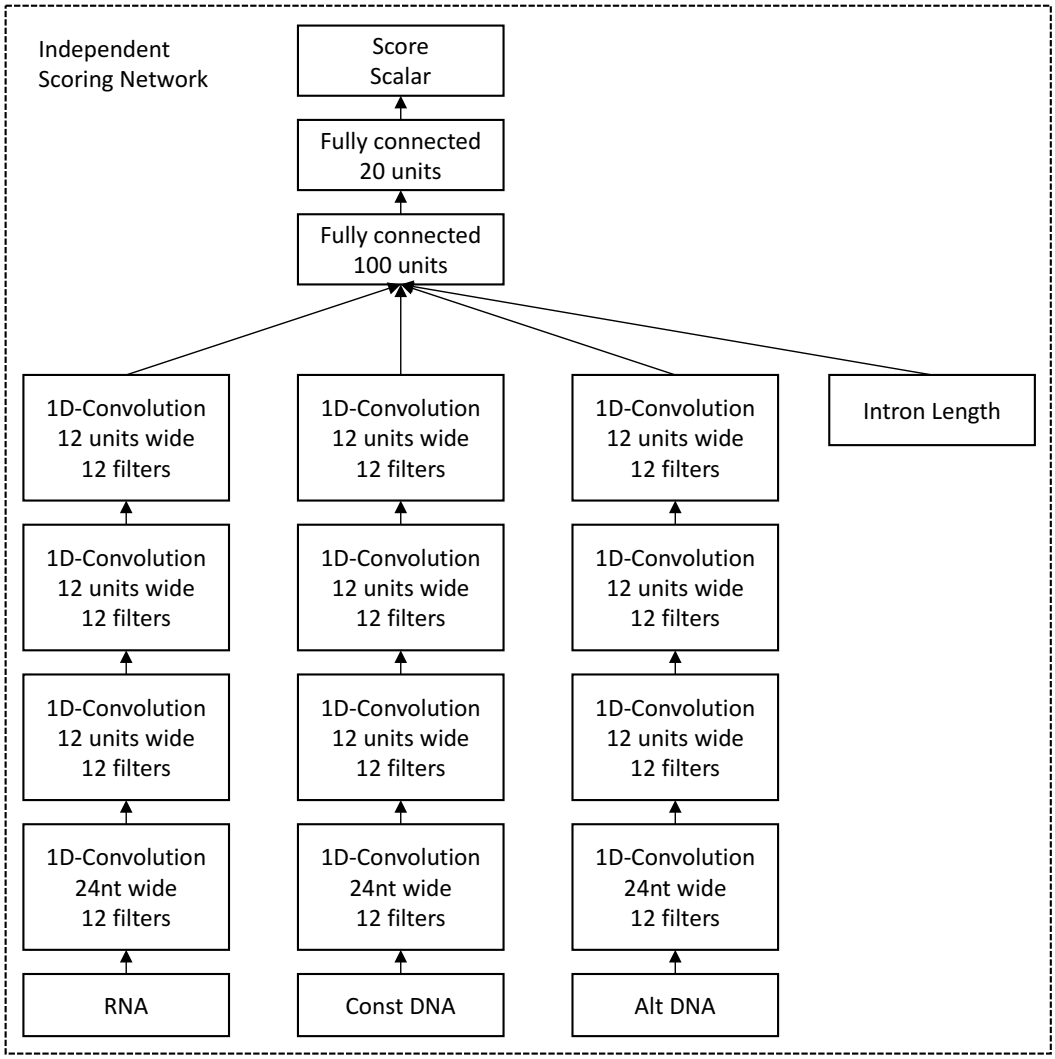


Figure 5: Independent convolutional scoring network

where S denotes the scoring network and θ represents the parameters of the scoring network S. Importantly, the parameters θ do not depend on i, but are shared between all splice sites.

We have implemented a variety of different architectures for the scoring network and evaluated their performance carefully. The simpler architectures we evaluated use scoring networks that are independent, such that the *competitive* behaviour of the model is achieved only through the requirement that the $PSI_i, i \in 1, 2, \ldots, K$ sum to 1.

However, we also evaluate a number of architectures in which we allow *lateral* connections between the alternative splice sites within the same event. In particular we utilize *communication networks* [Sukhbaatar et al., 2016] and LSTMs, which are described in Sections 3.3.2, 3.3.3, and 3.3.4 respectively.

3.3.1 Independent scoring networks

These types of networks take as input a set of splice site derived features and predict a scalar unnormalized score as in Eq (2) without any lateral connections to the other scoring networks. We use an architecture in which we use a stack of multiple convolutional layers on each of the input sequences. The outputs of each stack are concatenated with the intron length feature and followed by multiple fully connected layers.

bioRxiv preprint doi: https://doi.org/10.1101/255257; this version posted January 29, 2018. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

Figure 5 shows the architecture of the independent convolutional scoring network. We use three columns of stacked convolutions with independent parameters that each use one of the three sequence windows described in Section 3.2 as input. A *convolution module* contains the convolutional layer itself, followed by a ReLU non-linearity, and a batch-normalization layer [Ioffe and Szegedy, 2015]. We typically do not use any pooling, because some filters, like the core splice site motif, can be highly sensitive to the precise location and are not invariant to shifts.

Following any number of convolution modules, the final filtermaps from the three convolution columns are flattened and concatenated with the auxiliary features. These activations are the input to the following sequence of fully connected modules, each consisting of a fully connected layer, a ReLU function, and a batch-normalization layer.

The final fully connected module, as in all other architectures, has an output size of one to connect to the softmax output layer.

3.3.2 Communication Networks

When the scoring networks for competing splice sites are independent, they can only interact linearly through the softmax layer in the output network. We use an approach that is similar to Communication networks [Sukhbaatar et al., 2016] to model more complicated interactions between the splice sites, which we adapt to convolutional layers.

In our version of communication networks, a convolutional layer uses two sets of filters: one which operates on the current splice site and one which operates on a shared buffer that contains the average inputs of all splice sites. Each splice site's consecutive hidden layer then takes both the shared communication buffer's activations and the activations from the splice site's own previous hidden layer as input.

We apply communication to the convolutional layers in the following way: the output of a convolution layer with communication for a given splice site k is the sum of a global bias term, the response of a filter to the k-th input and the response of a second set of filters to the inputs averaged across all splice sites $\tilde{k} = 1, \ldots, K$.

The output $y_{k,g,j}$ for splice site k, output filter g at position j is

$$y_{k,g,j} = b_g + \sum_{f=1}^{F} \sum_{i=1}^{l} \left(w_{g,f,i} x_{k,f,j+i} + v_{g,f,i} \frac{1}{K} \sum_{k'=1}^{K} x_{k',f,j+i} \right), \tag{3}$$

where b_g is the bias of the g-th filter, l is the filter width, $w_{g,f,i}$ is the filter value of the g-th filter to the f-th channel at position i, $x_{k,f,j+i}$ is the input value of splice site k in the f-th channel at the $j+i$-the position, $v_{f,i}$ is the weight of the g-th communication filter to the f-th channel, at position i, and $\frac{1}{K}\sum_{k'=1}^{K} x_{k',f,j+i}$ is the input averaged across all splice sites in channel f at the $j+i$-th position.

Figure 6 further illustrates the communication network concept.

3.3.3 Output LSTM

Communication networks are one possibility to model the interaction between splice sites, but an alternative that explicitely takes the ordering of the splice sites into account are *Long Short-Term Memory* (LSTM) networks [Hochreiter and Schmidhuber, 1997]. LSTM networks are a type of recurrent neural network architecture, that uses memory cells with gates that control the flow of information and can be learning by backpropagation.

LSTMs can be used on any kind of sequential or time-series data such as strings of words in Machine Translation or frames of speech recordings. For the purpose of alternative splicing prediction we apply LSTMs to the sequence of splice sites. We use *bidirectional LSTMs* with one LSTM cell modelling the sequence of splice sites from 5' to 3' and one LSTM cell from 3' to 5'.

A single LSTM cell for splice site k outputs a value m_k depending on an input x_k as

bioRxiv preprint doi: https://doi.org/10.1101/255257; this version posted January 29, 2018. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

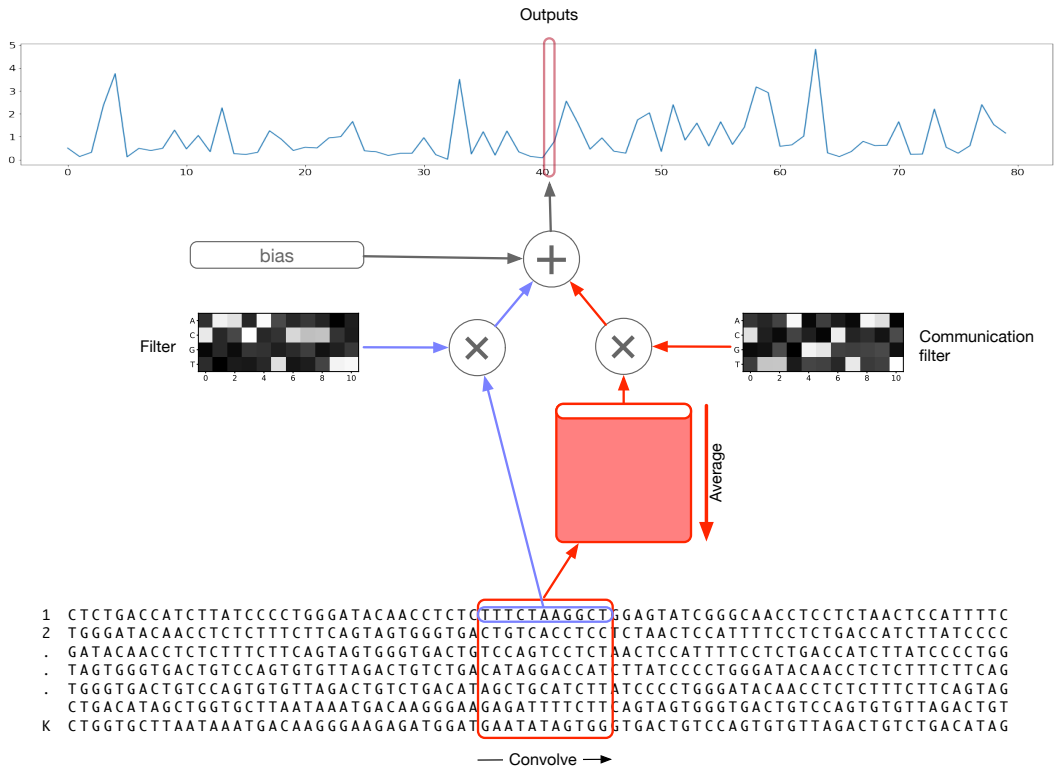


Figure 6: *Convolutional layer with communication*: This is a variant on the typical 1-D convolutional layer in case we apply a convolution to multiple sequences such as a set of alternative splice site. For a given sequence $k = 1, \ldots, K$, the output of this layer is the sum of a bias term, the filter responses to the sequence k, as well the responses to a second set of filters which are applied to the average of all inputs across k.

$$m_k = o_k \odot \tanh(c_k) \tag{4}$$
$$o_k = \sigma(W_{ox}x_k + W_{om}m_{k-1} + b_0) \tag{5}$$
$$c_k = f_k \odot c_{k-1} + i_k \odot \tanh(W_{cx} + W_{cm}m_{k-1} + b_c) \tag{6}$$
$$f_k = \sigma(W_{fx}x_k + W_{fm}m_{t-1} + b_f) \tag{7}$$
$$i_k = \sigma(W_{ix}x_k + W_{im}m_{t-1} + b_i) \tag{8}$$

In the equations above, m_k is the cell output activation vector at step k, o_k is the output gate, c_k is the cell activation vector, f_k is the forget gate, and i_k is the input gate. W denote the weight matrices (e.g. W_{ix} is the weight matrix from the input gate to the input) and b variables denote the bias terms (i.e. b_i is the input gate bias vector).

We implement a hybrid architecture in which we keep the lower convolutional layers from the independent scoring networks (Section 3.3.1) and replace that models fully connected layers with a bidirectional LSTM which consists of one LSTM running from 3' to 5' and the other one from 5' to 3' (this is easily implemented by replacing $k-1$ by $k+1$ everywhere in Eqs.(4)–(8).

Figure 7 shows the architecture we have chosen. As in the *independent scoring network*, three different sequence windows from each splice site are first processed by a series of 1D convolutional modules before being concatenated with the intron length and are further passed through several fully connected modules. However, now the signal from splice site k is further processed by two LSTM cells which will connect to the adjacent splice sites $k-1$ and $k+1$, allowing for propagation of information between the splice sites.

bioRxiv preprint doi: https://doi.org/10.1101/255257; this version posted January 29, 2018. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

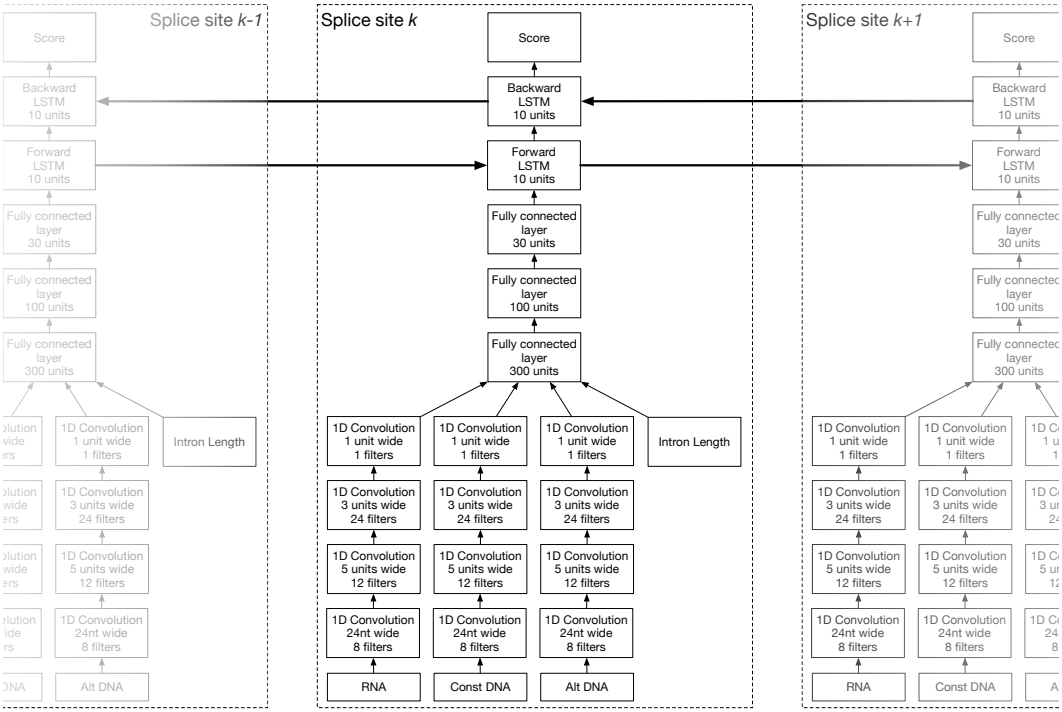


Figure 7: *Architecture of our Bidirectional LSTM Model.* Features for splice site k are extracted from the DNA/RNA sequences using a series of convolutional modules. The filtermaps for the different sequences are concatenated with the intron length feature. The concatenated features are then futher processed by several fully connected modules and two LSTM cells. The forward cell is connected to splice site $k-1$ and the backwards cell is connected to splice site $k+1$.

3.3.4 Resnet + LSTM

Resnets [He et al., 2015] are a class of models that enable training neural networks that are much deeper than previously possible by explicitly reformulating the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. Instead of learning a mapping $H(x)$ directly, a resnet implements the residual mapping $F(x) = H(x) - x$. The desired mapping $H(x)$ can then be reformulated as $F(x) + x$. In practice, resnets add shortcut connections that skip one or several layers. These shortcut connections can be added almost anywhere, for example around convolutional or fully connected layers.

In our implementation we replace the convolutional layers from the LSTM model (Section 3.3.3) with the Resnet-26 model [He et al., 2015], while keeping the higher level fully-connected layers and LSTM the same. Table 2 shows the parameters of the Resnet-26 architecture.

4 Results

4.1 Genome-wide performance

We trained each of the models presented above on both our acceptor and donor datasets. We use five-fold cross-validation by splitting our datasets according to the following method: using a transcript database, we start with a set of intervals that each span two consecutive genes initially. Then we iteratively merge all regions that are less than 250nt apart until merging is no longer possible. Then we randomly split the resulting genomic regions into five folds for cross validation.

Figure 8 plots the cross-entropy error over time during training. It is evident that the LSTM and Resnet models are able to significantly better fit the training data and achieve a lower final training loss.

bioRxiv preprint doi: https://doi.org/10.1101/255257; this version posted January 29, 2018. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

	Parameters			
Bottleneck Block	width	out_dim	oper_dim	stride
1	24	8	1	1
2	8	32	8	1
3	8	64	16	2
	8	64	16	1
4	8	128	32	2
	8	256	64	1
5	8	256	64	1
	8	256	64	1

Table 2: *Resnet-26 architecture:* This residual network replaces the convolutional layers in the LSTM model for the Resnet+LSTM model. Each bottleneck block can be skipped with a residual connection. Within each bottleneck block there are one or more convolutional layers parametrized by *width* (the width of the convolutional filter), *out_dim* (the number of output filters), *oper_dim* (the number of filters in the middle of the bottleneck block), and *stride* (the stride of the convolution operation).

We train each model by training on four of the folds and using the held out data as test set. We compute the accuracy and loss on each fold, then we report the mean score across the folds as well as the standard error. Table 3a shows the accuracy of the different COSSMO models as well as MaxEntScan on the same data. We define the accuracy as the frequency with which COSSMO correctly predicts the splice site with the maximal PSI value.

Whereas MaxEntScan can only predict the strongest splice site with a probability of 32.2% (acceptor), or 37.1% (donor), even the simple independent COSSMO model is able to significantly outperform it (52.0% on acceptor and 56.9% on donor). Our best model which uses an LSTM, achieves 70.0% on the acceptor dataset and 71.1% on the donor. Despite being a deeper, more powerful model, the Resnet with an LSTM achieves slightly worse performance than the "LSTM only" model. As a conclusion we recommend the LSTM model as the one that achieves the best balance of capacity and generalization error.

Accuracy is intuitive to interpret but only takes into account whether COSSMO predicts the strongest splice site in a large set correctly, but not how well it fits the complete PSI distribution. Table 3b shows the cross-entropy error of the four different COSSMO models. It is evident that COSSMO vastly outperforms MaxEntScan as well on negative cross-entropy, meaning COSSMO is not only more likely to predict the correct dominant splice site but will also fit the PSI distribution better overall.

4.2 Performance on alternatively spliced events

The majority of the events in our dataset has only one annotated splice site (Figure 2). To gain a deeper understanding of COSSMO's performance versus MaxEntScan, it also helps to stratify the dataset and only look at events that are geniunely alternatively spliced (have more than one *annotated* splice site). The performance of MaxEntScan and COSSMO on this subset can be seen on the right hand column of Table 3. This is a much more challenging subset of the data and performance of MaxEntScan and COSSMO is both impacted when more than one annotated site is present. However, while MaxEntScan accuracy drops by almost half on this subset (56%), COSSMO's relative drop in performance is much lower, with the more complicated models ("LSTM" and "LSTM + Resnet") seeing a smaller relative drop in accuracy than the simpler models ("Independent" and "Comm-net").

5 Discussion

In this work we introduced *COSSMO* - a computational model that for the first time enables accurate prediction of competitive alternative splice site selection from sequence context alone. We described

bioRxiv preprint doi: https://doi.org/10.1101/255257; this version posted January 29, 2018. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

Model	All events		Alternatively spliced	
	Acceptor $n = 345,840$	Donor $n = 344,420$	Acceptor $n = 40,802$	Donor $n = 49,898$
MaxEntScan	0.322	0.371	0.181	0.210
Independent	0.520 ($\pm$ 0.004)	0.569 ($\pm$ 0.003)	0.331 ($\pm$ 0.003)	0.376 ($\pm$ 0.002)
Comm-net	0.540 ($\pm$ 0.008)	0.653 ($\pm$ 0.001)	0.350 ($\pm$ 0.007)	0.482 ($\pm$ 0.005)
LSTM	0.700 ($\pm$ 0.004)	0.711 ($\pm$ 0.002)	0.533 ($\pm$ 0.006)	0.548 ($\pm$ 0.005)
Resnet + LSTM	0.698 ($\pm$ 0.005)	0.701 ($\pm$ 0.003)	0.526 ($\pm$ 0.008)	0.540 ($\pm$ 0.005)

(a) Accuracy of our COSSMO models. Accuracies are computed as the average of five cross-validation folds with standard errors given in brackets. The right columns shows accuracy only for those events that contain two or more annotated splice sites.

Model	All events		Alternatively spliced	
	Acceptor $n = 345,840$	Donor $n = 344,420$	Acceptor $n = 40,802$	Donor $n = 49,898$
MaxEntScan	2.836	2.244	3.819	3.043
Independent	1.583 ($\pm$ 0.008)	1.401 ($\pm$ 0.020)	2.384 ($\pm$ 0.006)	2.253 ($\pm$ 0.041)
Comm-net	1.490 ($\pm$ 0.021)	1.107 ($\pm$ 0.003)	2.254 ($\pm$ 0.018)	1.777 ($\pm$ 0.020)
LSTM	1.017 ($\pm$ 0.012)	0.944 ($\pm$ 0.006)	1.689 ($\pm$ 0.021)	1.602 ($\pm$ 0.012)
Resnet + LSTM	1.014 ($\pm$ 0.010)	0.955 ($\pm$ 0.009)	1.688 ($\pm$ 0.010)	1.603 ($\pm$ 0.009)

(b) Cross-entropy loss of our COSSMO models. Loss is computed as the average of five cross-validation folds with standard errors given in brackets. The right columns shows loss only for those events that contain two or more annotated splice sites.

Table 3: Performance of COSSMO and MaxEntScan on our datasets.

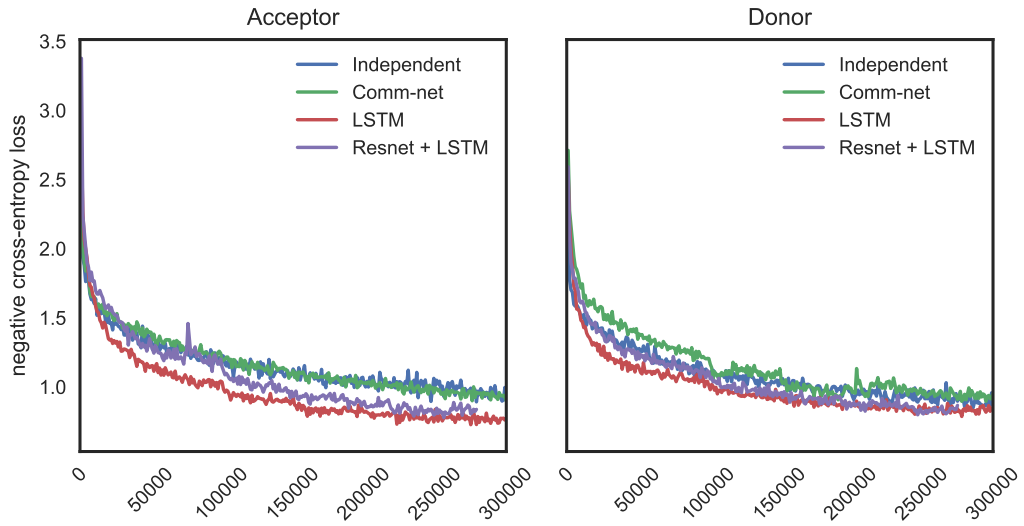


Figure 8: Training cross-entropy loss.

how we generated a genome-wide dataset for training and evaluation by combining positive splice site examples from genome annotations and large-scale RNA-Seq datasets, as well as negative examples from random genomic background sequences as well as decoy splice sites that receive high scores from MaxEntScan, but lack evidence of splice site usage from RNA-Seq data.

We designed several novel neural network architectures that can adapt to a variable number of alternative splice sites and carefully evaluated them using genome-wide cross-validation and found that our LSTM-based model achieves close to twice the accuracy of MaxEntScan.

bioRxiv preprint doi: https://doi.org/10.1101/255257; this version posted January 29, 2018. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

Our work demonstrates that it is possible to use deep learning to predict splice site choice with high accuracy which can be extended to predict how genomic variation affects splice site choice through mechanisms like splice site variants or cryptic splice site activation.

References

Yoseph Barash, John A Calarco, Weijun Gao, Qun Pan, Xinchen Wang, Ofer Shai, Benjamin J Blencowe, and Brendan J Frey. Deciphering the splicing code. *Nature*, 2010.

Anke Busch and Klemens J Hertel. Splicing predictions reliably classify different types of alternative splicing. *RNA*, 2015.

The GTEx Consortium. The Genotype-Tissue Expression (GTEx) project. *Nature Genetics*, 2013.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL `http://arxiv.org/abs/1512.03385`.

S Hochreiter and J Schmidhuber. Long short-term memory. *Neural Computation*, 1997.

Xin Hong, Douglas G Scofield, and Michael Lynch. Intron size, abundance, and distribution within untranslated regions of genes. *Molecular Biology and Evolution*, 2006.

Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv.org*, 2015.

Daehwan Kim, Ben Langmead, and Steven L Salzberg. HISAT: a fast spliced aligner with low memory requirements. *Nature Methods*, 2015.

Michael K K Leung, Hui Yuan Xiong, Leo J Lee, and Brendan J Frey. Deep learning of the tissue-regulated splicing code. *Bioinformatics (Oxford, England)*, 2014.

Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Learning Multiagent Communication with Backpropagation. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

Janne J. Turunen, Elina H. Niemelä, Bhupendra Verma, and Mikko J. Frilander. The significant other: splicing by the minor spliceosome. *Wiley Interdisciplinary Reviews: RNA*, 4(1):61–76, 2013. ISSN 1757-7012. doi: 10.1002/wrna.1141. URL `http://dx.doi.org/10.1002/wrna.1141`.

Z Wang and C B Burge. Splicing regulation: from a parts list of regulatory elements to an integrated splicing code. *RNA*, 2008.

H Y Xiong, B Alipanahi, L J Lee, H Bretschneider, D Merico, R K C Yuen, Y Hua, S Gueroussov, H S Najafabadi, T R Hughes, Q Morris, Y Barash, A R Krainer, N. Jojic, S W Scherer, B J Blencowe, and B.J. Frey. The human splicing code reveals new insights into the genetic determinants of disease. *American Association for the Advancement of Science. Science*, 2015.

Hui Yuan Xiong, Yoseph Barash, and Brendan J Frey. Bayesian prediction of tissue-regulated splicing using RNA sequence and cellular context. *Bioinformatics (Oxford, England)*, 2011.

Hui Yuan Xiong, Leo J Lee, Hannes Bretschneider, Jiexin Gao, Nebojsa Jojic, and Brendan J Frey. Probabilistic estimation of short sequence expression using RNA-Seq data and the positional bootstrap. *bioRxiv*, 2016.

Gene Yeo and Christopher B Burge. Maximum entropy modeling of short sequence motifs with applications to RNA splicing signals. *Journal of Computational Biology*, 2004.