Novel Insights into RNA Splicing from Interpretable Machine Learning

Mukund Sudarshan[*] **Susan E. Liao**[*] **Oded Regev**[*]
[*]Courant Institute of Mathematical Sciences
New York University

1 Introduction

The advent of high-throughput next-generation nucleic acid sequencing revolutionized biomedical research by enabling the acquisition of large DNA and RNA datasets. Intense efforts have been directed towards developing experimental and computational techniques to use next-generation sequencing to gain insights into nucleic acid processes.

However, despite the high quality of sequencing data, the use of modern machine learning techniques such as deep learning has been limited due to their lack of interpretability. Accurate predictions alone are insufficient for advancing our understanding of the underlying nucleic acid processes. In order to drive biological discovery, models must be designed with interpretability in mind [12].

RNA splicing: Messenger RNA undergoes extensive nuclear processing. One of the most important of these processes is RNA splicing [7]. During this process, introns are removed from the transcript, and the remaining exons are joined. The "splicing code" encompasses the RNA sequences that inform splicing decisions on where cleavage and ligation takes place [5]. While the core splicing signals (5' and 3' splice sites and branch point sequences) are relatively well understood [15], exons and introns contain *splicing motifs* (also known as splicing regulatory elements), which are short sequence patterns (or *motifs*) that play a key role in splicing decisions, and are not as well understood. These splicing motifs exert their effect by serving as binding sites for *splicing factor* proteins.

Contribution: We show how convolutional networks can be used to extract a succinct and biologically informative set of rules for splicing. Using a published dataset from a splicing reporter assay [10], our method is able to produce a short list of only 6 splicing motifs extracted from the filters of these networks that together enable highly accurate predictions of splicing outcomes (Fig. 1). Remarkably, even though the list is produced automatically from sequencing data alone without any prior knowledge about biology, the discovered motifs all closely match motifs of known splicing factors that were previously implicated in splicing. Interestingly, one of the strongest motifs, corresponding to the SRSF3 splicing factor, was missed in a previous analysis of this dataset [10], demonstrating the advantages of our method.

In addition, we use our motif identification method to understand how the various motifs combine to form splicing decisions. In particular, we can quantify the strengths of the various splicing motifs (Fig. 1D). One novel observation is the identification of SRSF3 as a particularly strong splicing enhancer, more than twice as strong as SRSF1.

As further evidence that the discovered motifs correspond to a biological ground truth, we apply our method to four independent biological experiments. Notably, our method discovers the same motifs in all four experiments (Fig. 2).

Together, these results demonstrate how interpretable models can be used to derive biological insights from high throughput sequencing data. Our method can also be applied to other recent high throughput assays, including those focusing on alternative polyadenylation [1], 5'UTR regulation [11], mRNA degradation [8], and more.

Technical contribution: A major obstacle to deriving biologically meaningful motifs is that motifs extracted from convolutional networks are generally not reproducible. Due to the non-identifiable nature of neural

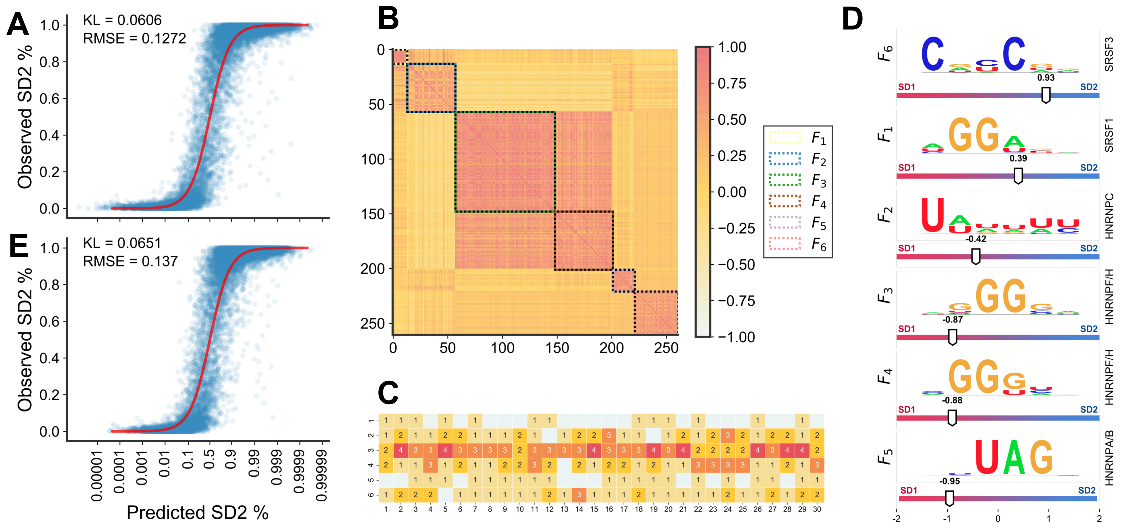

Figure 1: The 6 representative motifs found in our dataset are sufficient to achieve high performance in predicting splicing. (A) Scatter plot of the observed SD2% versus predicted SD2% generated using the median $\hat{q}_{\text{conv}}$ model across bootstraps in terms of held-out KL. This neural network has 24 filters with no prior on the weights. This network achieves a KL of 0.0606 and an RMSE of 0.1272 on the test set. The red line shows how perfect predictions would appear on the graph. (B) Correlation plot between the embeddings of all filters collected across 30 bootstraps after a screening step to remove noisy filters. The filters were clustered into six clusters, $F_1, \ldots, F_6$ and shown in this order. (C) Number of times a filter from each cluster is learned by a bootstrapped model. Filters that are strongly predictive of splicing can be learned multiple times in a single bootstrapped model, while filters that have less of an effect may be missed in several bootstraps. (D) The logo sequence for the representative filter in each cluster found by our method, ranked by splicing decision score (shown in the sliders). For example, the presence of the motif in F_6 will increase the probability of splicing at SD2, while the presence of the motif in F_5 will have the opposite effect. We also show the best matched splicing factor for each filter [2, 4, 9]. (E) Scatter plot of observed SD2% versus the predicted SD2%. The network used to generate this plot was trained using only the 6 filters identified by our method (shown in (D)). Only the linear layer weights were re-trained. This network achieves a KL of 0.0651 and an RMSE of 0.1370 on the test set. The red line shows how perfect predictions would appear on the graph.

network parameters, each time we train the network (with a different random seed or bootstrap) we get different motifs. These irreproducible motifs are unlikely to be biologically meaningful.

Our main technical contribution is a method to extract a small list of motifs from splicing prediction models in a reproducible way. The method involves grouping convolutional filters learned during multiple training runs on bootstraps of the training data, and then extracting a representative motif from each group of filters. We then show how this small list of filters can be used in downstream analyses to understand how motifs combine to form a splicing decision. We observe that a model that uses only the discovered motifs incurs very little loss in accuracy, despite a significant reduction in the number of filters (from 24 to 6 in our case). More importantly, we observe that the resulting motifs all seem biologically informative, as they agree with previous biochemical data.

Comparison to previous work: A prior analysis of the same splicing dataset was based on a direct k-mer enrichment analysis [10]. This analysis outputs a list of scores for all 4096 possible 6-mers, and as such is difficult to interpret. For instance, while some splicing motifs were manually identified among the top scoring 6-mers (such as those corresponding to hnRNPF/H or hnRNPA/B splicing factors), other important motifs were missed (such as those corresponding to SRSF3 or hnRNPC splicing factors). Our method replaces such an opaque list of k-mer scores with an interpretable shortlist of motifs.

2 Interpreting RNA splicing with models

Dataset: We use a previously published splicing dataset [10], which we filtered to remove experimental artifacts (see appendix). It consists of RNA sequences $\mathbf{s}_i \in \{A, C, G, U\}^d$ of length $d = 83$. Each $\mathbf{s}_i$ is associated with a response $\mathbf{y}_i \in [0, 1]$ representing the observed probability of splicing in one of two competing splice sites.

Model design: It was previously observed that splicing choices are based on the *additive contribution* of short sequence motifs [10]. That is, it is possible to assign a score to each of the 4^k possible k-mers (typically $k = 6$), such that the output $\mathbf{y}_i$ agrees with a sigmoid-like function of the sum of the scores of the $d - k + 1$ k-mers in the sequence $\mathbf{s}_i$.[1] The scores appeared to correlate with the presence of short motifs; for instance, 6-mers containing GGG as a subsequence all have low scores. This agrees with the biological fact that this motif serves as a binding site for the hnRNP F/H splicing factors [3].

Based on these observations, and in order to ensure interpretability of the trained network, we decided to use a simple 2-layer network. Briefly, the first layer is a 1-dimensional convolutional layer applied to a one-hot encoding of the input sequence $\mathbf{s}_i$. The second and final layer is a linear layer with sigmoid activation. The model is fit using KL minimization between $\mathbf{y}$ and its predictions.

Accuracy: We split the dataset into a training set of size 29,118 and a test set of size 12,480. Despite its simplicity, this model performs remarkably well in terms of accuracy as shown in Fig. 1A. Specifically, a single model fit to the training set achieves a held-out KL divergence of only 0.0602 ± 0.0011 (corresponding to 0.1271 ± 0.0018 RMSE). Such good agreement with the biological data suggests that interpretations of the model would yield accurate understanding of the underlying splicing code.

Interpretability: The network's simple design facilitates interpretability. Specifically, the filters in the convolutional layer provide potential splicing motifs. Moreover, the weights in the linear layer provide a "strength" to each such motif, as well as a "direction" (increasing splice site usage if positive, and decreasing it if negative).

Unfortunately, we observed that the motifs discovered in this way are not reproducible. While some of them showed up consistently across runs and agreed with the biological literature (e.g., the triple-G motif), most seemed like artifacts. This led us to develop a method to robustly identify motifs, as described next.

2.1 Method for identifying reproducible filters

The premise behind our method is that each filter is drawn from a neighborhood of filters in a latent embedding space. In each neighborhood of this space, there may be many possible filters that all correspond to a single motif. The goal of our method therefore, is to model this embedding space of convolutional filters across multiple runs and identify groups of filters. We then identify the motif that each group corresponds to.

More specifically, our method involves several stages. We first fit several models to bootstraps of the data. For each filter in each model, we generate an embedding vector. We then cluster all filters across all bootstraps using their embedding vectors. Finally, each cluster is assigned a representative motif.

Bootstrapping models: We first bootstrap the training data $\{\mathbf{x}_i\}_{i=1}^N$ B times. Each bootstrapped dataset consists of N samples that are drawn uniformly at random from the training set with replacement. To each bootstrapped dataset, we fit a two-layer convolutional network $\hat{q}_{\text{conv}}(\mathbf{y} \mid \mathbf{x})$ to get the set of models $\{\hat{q}^{(b)}_{\text{conv}}(\mathbf{y} \mid \mathbf{x})\}_{b=1}^B$. The first layer is a one-dimensional convolutional layer with J filters $\{\mathbf{f}_j\}_{j=1}^J$ of width W applied to $\mathbf{x}_i$, the one-hot encoding of an RNA sequence, followed by the second and final linear layer, whose output predicts $\mathbf{y}_i$. Since $\hat{q}_{\text{conv}}$ is able to model the data so well, it makes an excellent candidate for interpreting splicing patterns.

Generating embeddings: For each model $\hat{q}^{(b)}_{\text{conv}}$, for each filter $\mathbf{f}^{(b)}_j$, we compute an embedding vector $\mathbf{z}^{(b)}_j \in \mathbb{R}^N$ as follows:

$$\mathbf{z}^{(b)}_j = [\langle \theta^{(b)}_j, \sigma(\text{Convolution}(\mathbf{f}^{(b)}_j, \mathbf{x}_1))\rangle, \ldots, \langle \theta^{(b)}_j, \sigma(\text{Convolution}(\mathbf{f}^{(b)}_j, \mathbf{x}_N))\rangle]$$

where $\theta^{(b)}_j$ is a vector of the linear layer parameters for the bth neural network that corresponds to the jth filter and $\sigma(\mathbf{a}) = (1 + \exp(-\mathbf{a}))^{-1}$ is the sigmoid activation function. Note that the ith coordinate of the embedding vector $\mathbf{z}^{(b)}_j$ is simply the contribution of filter $\mathbf{f}^{(b)}_j$ to the output of the bth bootstrap model on input $\mathbf{x}_i$ (before the final sigmoid activation).

Clustering embeddings: We now cluster the filters using the embeddings of all $J \cdot B$ filters. We first compute the pairwise correlations between all $J \cdot B$ filters. We then remove any filter that does not correlate highly with any other filter, as such filters likely represent artifacts. The remaining filters are clustered using the Voorhees algorithm [14], which groups filters by their minimum similarity (a user-defined hyperparameter) with other filters. As a similarity metric for this algorithm, we use the Pearson distance (one minus the correlation between two embeddings).

[1]This is actually a weighted sum with weights depending on the position along the sequence.

Representative motifs: Finally, we generate a representative motif for each cluster of filters. For each filter $\mathbf{f}_j^{(b)}$ in cluster $c_k \in \{c_1, \ldots, c_K\}$, we compute a vector of synthetic activations

$$\mathbf{g}_j^{(b)} := [\sigma(\text{Convolution}(\mathbf{f}_j^{(b)}, \mathbf{m}_1)), \ldots, \sigma(\text{Convolution}(\mathbf{f}_j^{(b)}, \mathbf{m}_{4^W}))]$$

where $\{\mathbf{m}_i\}_{i=1}^{4^W}$ is the set of all possible k-mers (consecutive subsequences) of length W. Then, for each cluster c_k, a representative filter $\mathbf{f}_k$ is chosen by selecting the one whose corresponding $\mathbf{g}_j^{(b)}$ has the highest mean correlation with the other filters in c_k.

3 Experiments

To evaluate our method, we use the dataset from [6, 10]. Each sequence $\mathbf{s}_i$ in the dataset consists of 83 nucleotides. For our experiments, we consider two competing splice sites: SD1 and SD2. The corresponding labels $\mathbf{y}_i$ represent $p_{\text{SD2}}/(p_{\text{SD1}} + p_{\text{SD2}})$, where p_{SD1} and p_{SD2} are the observed probabilities of a splicing event at SD1 or SD2. We split the dataset into a training set of size 29,118, and a test set of size 12,480. We now apply the steps outlined in the previous section.

Bootstrapping models: To extract motifs that correspond to splicing, the training set is first bootstrapped 30 times to fit the models $\{\hat{q}_{\text{conv}}^{(b)}(\mathbf{y} \mid \mathbf{x})\}_{b=1}^{30}$. Each $\hat{q}_{\text{conv}}^{(b)}$ consists of 24 convolutional filters followed by a linear layer with sigmoid activation. We performed a basic grid search over the number of filters and found the performance of the networks to saturate after 24 filters. These models perform very well in terms of accuracy. The mean held-out KL the bootstrapped models achieve is 0.0602, with a standard deviation of 0.0011 (RMSE of 0.1271 ± 0.0018), indicating that the performance is very consistent across bootstraps (Fig. 1A). This supports our initial hypothesis that these neural networks are ideal candidates for interpreting the splicing code, despite their simple architecture.

Generating and clustering embeddings: We generate embeddings for each filter, then screen noisy filters by dropping the ones whose embeddings have correlation less than 0.35 with all other filter embeddings. This weak filtering step highlights the fact that many filters show up only once across many bootstraps. The remaining filters each have strong correlations with several other filters, as we will discuss shortly. We visualize the correlation matrix of the embeddings of each of the remaining filters in Fig. 1B.

Next, using the Voorhees algorithm with a minimum similarity threshold of 0.9, we cluster the filters using their embeddings. This means that filters whose embedding vectors have a correlation of 0.9 or greater will be in the same cluster. This process yields 6 clusters, shown as boxes on Fig. 1B.

Using these clusters, we also visualize the frequency of observing a filter from each cluster across bootstraps (Fig. 1C). Note that even strong motifs like UAG (F_5) do not show up in many of the bootstrapped runs, highlighting the importance of running the bootstraps.

Representative motifs: We compute the representative filter for each cluster. The motif discovered by each of these filters is visualized in Fig. 1D (using [13]), along with its best matched splicing factor.

Assigning reproducible scores to the robust motifs: Having demonstrated a method to reproducibly identify motifs in the splicing dataset, we now seek to explain how they combine to form a splicing decision. We therefore fit a new $\hat{q}_{\text{conv}}$ model whose convolutional layer is fixed to be only the six representative filters. Remarkably, this network is able to achieve performance on par with much larger networks as shown in Fig. 1E, further demonstrating the utility of the discovered motifs. Since the resulting optimization problem (over the linear layer only) is convex, the learned weights in the linear layer are reproducible, and as such are directly interpretable. To visualize these weights, we average them across positions in the input and arrive at one numerical *splicing decision score* for each motif, showing the contribution of each motif towards splicing in SD2 (sliders in Fig. 1D).

Consistency across biological experiments: We next demonstrate that the interpretation given by our method is reproducible not just across bootstraps but also across independent biological experiments. For that, we apply our method to three additional additional biological experiments. We observe that the resulting motifs and their scores are highly consistent across all four experiments (Fig. 2 and Fig. 1D).

To summarize, our method is able to reproducibly identify motifs and assign to them scores that together provide a comprehensive picture of splicing decisions.

References

[1] Nicholas Bogard, Johannes Linder, Alexander B. Rosenberg, and Georg Seelig. “A Deep Neural Network for Predicting and Engineering Alternative Polyadenylation”. In: *Cell* 178.1 (2019), 91–106.e23.

[2] Y Cavaloc, C F Bourgeois, L Kister, and J Stévenin. “The splicing factors 9G8 and SRp20 transactivate splicing through different and specific enhancers.” In: *RNA* 5.3 (1999), pp. 468–483.

[3] Cyril Dominguez, Jean-Francois Fisette, Benoit Chabot, and Frederic H.-T. Allain. “Structural basis of G-tract recognition and encaging by hnRNP F quasi-RRMs”. In: *Nature Structural & Molecular Biology* 17.7 (2010), pp. 853–861.

[4] Daniel Dominguez et al. “Sequence, Structure, and Context Preferences of Human RNA Binding Proteins”. In: *Molecular Cell* 70.5 (2018), 854–867.e9.

[5] Klemens J. Hertel. “Combinatorial control of exon recognition”. In: *The Journal of Biological Chemistry* 283.3 (2008), pp. 1211–1215.

[6] Johannes Linder, Nicholas Bogard, Alexander B. Rosenberg, and Georg Seelig. “A Generative Neural Network for Maximizing Fitness and Diversity of Synthetic DNA and Protein Sequences”. In: *Cell Systems* (2020).

[7] R. A. Padgett, P. J. Grabowski, M. M. Konarska, S. Seiler, and P. A. Sharp. “Splicing of messenger RNA precursors”. In: *Annual Review of Biochemistry* 55 (1986), pp. 1119–1150.

[8] Michal Rabani, Lindsey Pieper, Guo-Liang Chew, and Alexander F. Schier. “A Massively Parallel Reporter Assay of 3’ UTR Sequences Identifies InVivo Rules for mRNA Degradation”. In: *Molecular Cell* 68.6 (2017), 1083–1094.e5.

[9] Debashish Ray et al. “A compendium of RNA-binding motifs for decoding gene regulation”. In: *Nature* 499.7457 (2013), pp. 172–177.

[10] Alexander B. Rosenberg, Rupali P. Patwardhan, Jay Shendure, and Georg Seelig. “Learning the Sequence Determinants of Alternative Splicing from Millions of Random Sequences”. In: *Cell* 163.3 (2015), pp. 698–711.

[11] Paul J. Sample et al. “Human 5’ UTR design and variant effect prediction from a massively parallel translation assay”. In: *Nature Biotechnology* 37.7 (2019), pp. 803–809.

[12] Ammar Tareen and Justin B. Kinney. “Biophysical models of cis-regulation as interpretable neural networks”. In: *bioRxiv* (2020), p. 835942.

[13] Ammar Tareen and Justin B. Kinney. “Logomaker: beautiful sequence logos in Python”. In: *Bioinformatics (Oxford, England)* 36.7 (2020), pp. 2272–2274.

[14] Ellen M Voorhees. “The cluster hypothesis revisited”. In: *Proceedings of the 8th annual international ACM SIGIR conference on Research and development in information retrieval*. 1985, pp. 188–196.

[15] Zefeng Wang and Christopher B. Burge. “Splicing regulation: From a parts list of regulatory elements to an integrated splicing code”. In: *RNA* 14.5 (2008), pp. 802–813.

A Consistent motifs across independent experiments

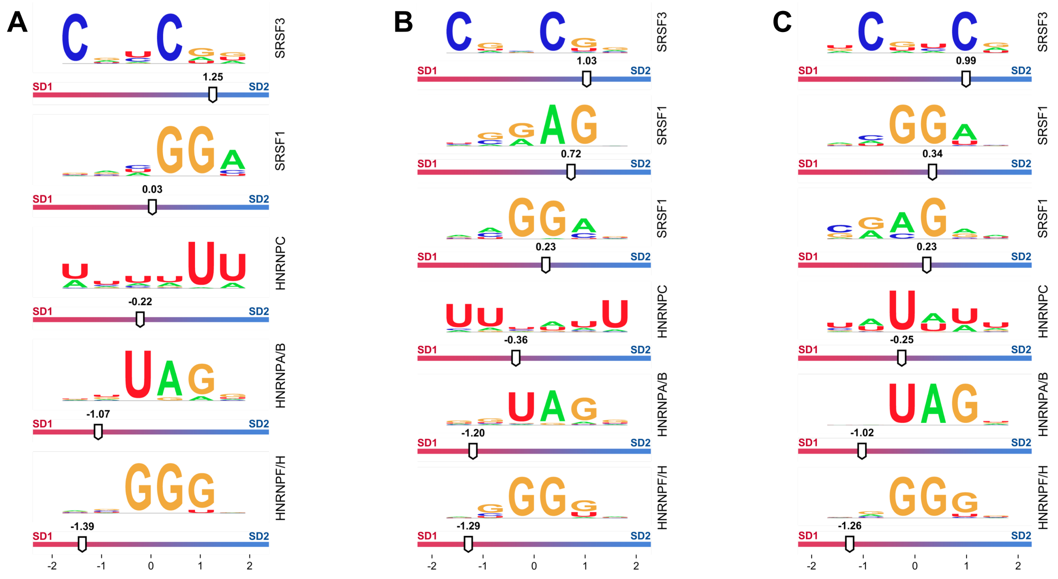

Figure 2: Our method identifies highly consistent motifs across additional independent experiments. The filters for each cell line are clustered separately in three additional biological experiments performed in three different cell lines: (A) MCF7, (B) HEK, and (C) HeLa.

B Data and preprocessing

Dataset: We used the results of a previously published assay [6, 10] which includes splicing outcomes for over $3 \cdot 10^5$ random test constructs. The test constructs all have two competing splice sites, called SD1 and SD2 (see Fig. 3). In between the splice sites is a randomized region of 25 nucleotides. An additional randomized region of 25 nucleotides is positioned downstream of SD2.

For each test construct, the dataset stores the random sequence contained in it, as well as the splicing measurement outcomes, namely, the number of RNA reads with splicing in SD1 and the number with splicing in SD2. Occasionally, splicing would occur in other positions along the sequence (cryptic splice sites), and these reads are recorded too.

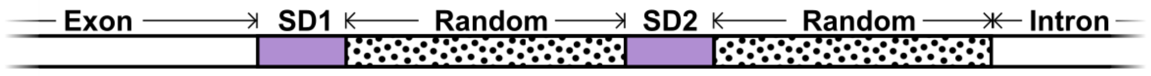

Figure 3: Splice site assay design from [10]

Artifact removal: We noticed that a significant fraction of the sequences in the dataset are mislabeled. That is, the splicing measurement results do not correspond to the provided sequence. We traced this artifact to badly coupled barcodes in the DNA library preparation. To remove these artifacts, we obtained the raw Illumina FASTA files from the NIH SRA database, and applied a stringent quality filter. Specifically, we filtered out all sequences with only one DNA read supporting the coupling between the barcode and the randomized sequence. We also filtered out sequences whose RNA sequencing suggested bad coupling (such as reads with SD2 splicing but with a different sequence in the first randomized region). This step reduced the size of the dataset by over 50% to 145,431 constructs. Despite that decrease in size, we noticed a significant improvement in prediction accuracy, as expected from a cleaner dataset.

Filtering: After artifact removal, we further filtered the dataset. First, we only kept sequences for which there were at least 20 reads. This is to ensure a sufficiently accurate estimate of the splice site usage probability. Moreover, we removed all constructs for which more than 10% of the reads came from splicing products outside SD1 and SD2. These constructs often correspond to cryptic splice sites occurring inside the randomized sequence, which we chose to ignore for the present analysis. The final filtered dataset had 41,598 constructs.