# Predicting RNA splicing branchpoints

Antonio Jovanović*¶, Israa Alqassem†‡¶, Nathan Chappell§, Stefan Canzar†, Domagoj Matijević*
* Department of Mathematics, University of Osijek, Osijek, Croatia
† Gene Center, Ludwig-Maximilians-Universität München, Munich, Germany
‡ NEC Laboratories Europe, Heidelberg, Germany
§ Mono Ltd., Osijek, Croatia
¶ These authors contributed equally
ajovanov@mathos.hr, israa.alqassem@neclab.eu

***Abstract*—RNA splicing is a process where introns are removed from pre-mRNA, resulting in mature mRNA. It requires three main signals, a donor splice site (5'ss), an acceptor splice site (3'ss) and a branchpoint (BP). Splice site prediction is a well-studied problem with several reliable prediction tools. However, branchpoint prediction is a harder problem, mainly due to varying nucleotide motifs in the branchpoint area and the existence of multiple branchpoints in a single intron. An RNN based approach called LaBranchoR was introduced as the state-of-the-art method for predicting a single BP for each 3'ss. In this work, we explore the fact that previous research reported that 95% of introns have multiple BPs with an estimated average of 5 to 6 BPs per intron. To that end, we extend the existing encoder in the LaBranchoR network with a PointerNetwork decoder. We train our new encoder-decoder model, named RNA PtrNets, on 70-nucleotide-long annotated sequences taken from three publicly available datasets. We evaluate its accuracy and demonstrate how well the predictor can generate multiple branchpoints on the given datasets.**

***Keywords*—RNA splicing, branchpoint prediction, recurrent neural network**

## I. Introduction

RNA splicing is a process in molecular biology where a pre-mRNA transcript is transformed into a mature messenger RNA (mRNA). It works by removing non-coding regions of RNA, i.e., introns, and joining together coding regions, i.e., exons. The two-step process of RNA splicing requires three main signals which work together in forming mRNA. The pre-mRNA molecule is first cut at the donor splice site and the 5'ss of the intron is joined to the branchpoint site (BP), forming a lariat intermediate molecule. Then, the pre-mRNA molecule is cut at the acceptor splice site (3'ss), after which two exons are joined together (see Fig. 1)

Previous research shows that alternative BP selection plays a role in alternative splicing and diseases (e.g. [1], [2]). Splice donor and acceptor sites are well studied in literature (e.g. [3], [4]) and there exist several reliable tools for splice site prediction, e.g., SpliceAI [5] and Splicing Prediction in Consensus Elements (SPiCE) [6]. However, the study of BP position has lagged behind due to two main challenges: (1) identifying BP positions is difficult in wet-lab experiments because lariats exist for a short period of time as low-abundance RNAs, and (2) predicting BP positions is difficult and we lack an agreed upon genomewide BP annotation due to two reasons [7]:

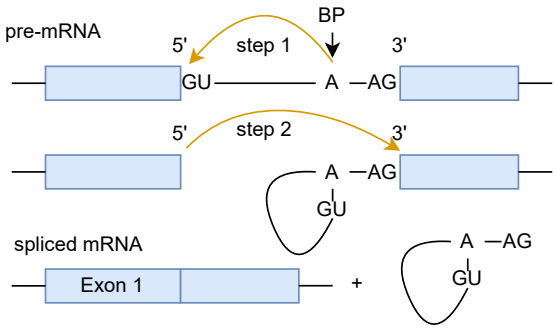


Fig. 1: The two-step mechanism of RNA splicing. Splicing begins with the dinucleotide sequence Guanine and Uracil (GU) at the 5'ss and ends with the dinucleotide sequence Adenine and Guanine (AG) at the 3'ss. Branchpoints (BPs) most frequently are Adenine (A).

- nucleotide motifs vary in the BP area;
- multiple BPs exist within the same intron.

Pineda and Bradley [8] reported that 95% of introns have multiple BPs with an average estimate of 5 to 6 BPs per intron.

In theory, the position of BPs can be determined using RNA-seq reads spanning the 5'ss and BP junctions. However, the intron lariat degenerates quickly during RNA splicing which makes such reads less frequent [9]. To date, human BP annotations are still incomplete and only a small fraction of BPs are experimentally verified.

### A. Related work

Standard machine learning techniques have been widely used on existing (incomplete) catalogues of human BPs. Corvell et al. [1] employs a Support Vector Machine (SVM-BP). Zhang et al. [10] adopts a mixture model and expectation maximization algorithm to tackle this problem. Signal et al. [11] presented Branch-Pointer, a gradient boosted trees based algorithm. However, all these approaches rely on genome annotation and a decent amount of feature engineering prior to the training step. Recently, with the emergence of deep learning methods, Paggi and Bejerano [12] introduced LaBranchoR (Long short-term memory network Branchpoint Retriever) as the current

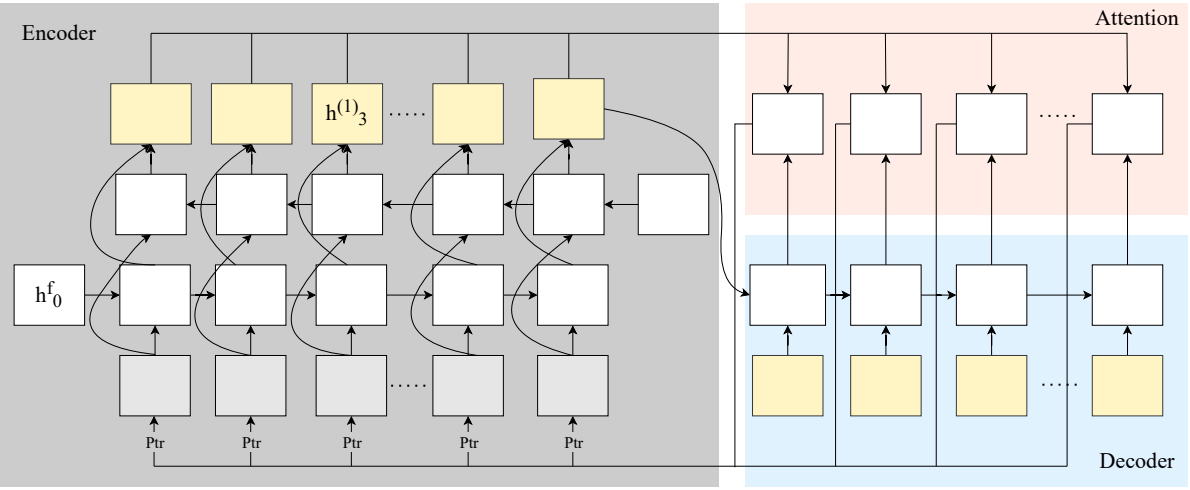


Fig. 2: Pointer network architecture diagram. Inputs to the network are represented as a sequence of tokens $x_1, \ldots, x_n$, where $x_1 = \$$ is a symbol different from all nucleotide bases. Forward and backward encoder hidden states are concatenated into $h_i^{(1)}$. The decoder is composed of LSTM cells which take $h_{k_l}^{(1)}$ as inputs and $h_{l-1}^{(2)}$ as previous hidden states, where $k_l$ is the previous index generated by the decoder. The LSTM cell outputs a hidden state $h_l^{(2)}$, which is then used to calculate the pointer vector $a_l$. Decoding stops when $a_m$ points to $x_1$.

state-of-the-art method for predicting a single BP for each 3'ss. LaBranchoR uses two layers of a bidirectional long short-term memory network (LSTM), the output of which is fed to a simple feedforward neural network. In their comparative benchmark analysis, LaBranchoR showed superior performance compared to previous traditional machine learning methods. Moreover, LaBranchoR was applied in a recent study [13] to examine cancer-related mutations of the splicing factor SF3B1.

### B. Our results

Only a small fraction of BPs are experimentally verified. The major goal of this work is therefore to investigate the feasibility of an extension of the architectural design underlying LaBranchoR, the current state-of-the-art method, to be able to recognize multiple potential BPs instead of just a single one. We achieved this by completely replacing the simple feedforward neural network in LaBranchoR with a decoder RNN, utilizing a specific attention mechanism first introduced in [14]. We refer to our architecture as RNA PtrNets. Additionally, we implemented various strategies for training, i.e., teacher forcing [15] and a curriculum learning strategy [16]. The model was evaluated using a beam search heuristic [17] during inference. We benchmark our approach against LaBranchoR, where we train both models on the same training set and we test them on two unseen BP sets.

## II. Models

The LaBranchoR model is a bidirectional LSTM encoder which feeds into a standard feedforward network. This last layer is interpreted as calculating the posterior probability that an index of the input is a branch point in the sequence, the index with the highest posterior probability is then considered to be the output of the model. While this is a straightforward and well-understood approach, it is inherently limited to outputting a single branch point or a posterior distribution over the input sequence.

Our model is a sequence-to-sequence recurrent neural network (RNN), the standard encoder-decoder neural model used in many modern machine translation services [18]. Our encoder RNN is essentially the same as in the LaBranchoR approach: we used bidirectional LSTM. However, instead of using a standard feedforward network on the output, our model uses the attention mechanism and decoder RNN introduced in Pointer Network (PtrNet) [14] that was successfully used for solving different combinatorial optimization problems.

More formally, let $(h_t^{(1)})_{t=1}^n$ and $(h_t^{(2)})_{t=1}^m$ denote hidden states from the encoder and decoder, respectively. Note that in the case when the encoder RNN is bidirectional LSTM, the forward LSTM is fed the input sequence and the backward LSTM is fed the reverse of the input sequence. Forward state outputs are not connected to backward state inputs and vice versa, i.e., the input sequence and its reverse are processed independently. Forward states are concatenated with the corresponding backward states in order to form the final sequence of encoder hidden states $(h_t^{(1)})$. Our decoder RNN is forward LSTM on top of which a particular attention mechanism is implemented computing the final output. In [14] the attention vector $a^i$ at each output time $i$ in decoder is computed as follows:

$$\begin{aligned} u_j^i &= v^T \tanh(W_1 h_j^{(1)} + W_2 h_i^{(2)}) \quad j \in \{1, \ldots, n\} \\ a^i &= \text{softmax}(u^i) \end{aligned} \tag{1}$$

Note that vector $v$ and matrices $W_1$ and $W_2$ are learnable

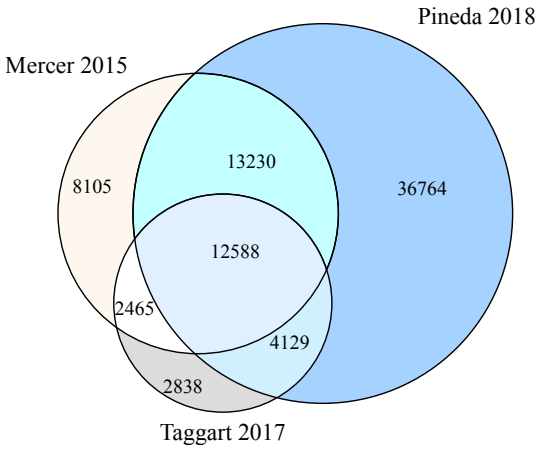


Fig. 3: Total number of 70 nucleotide-long RNA sequences associated with BPs reported in each dataset. We evaluated the model on a subset of the Mercer and Taggart datasets, disjoint from the Pineda dataset.

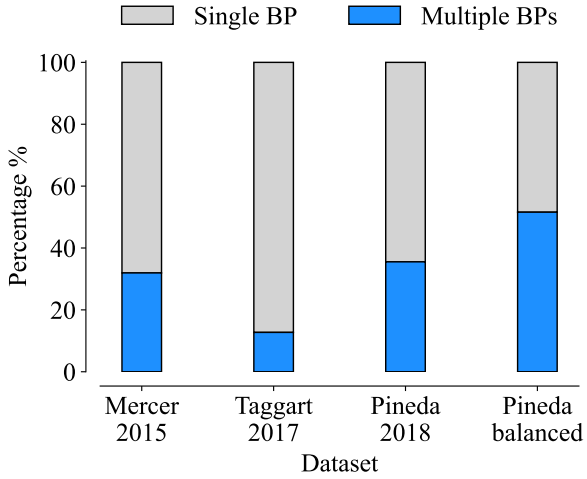


Fig. 4: Ratios of sequences with single versus multiple BPs.

parameters of the model. Moreover, attention vector $a^i$ of length $n$ is a probability vector over an input sequence, as a result of a softmax function. The final 'trick' is to interpret the index in $a^i$ with the highest probability as an output in time $i$, i.e., the output argmax$(a^i)$ in time $i$ is a 'pointer' that points to the argmax$(a^i)$ position in the input sequence (hence the name Pointer Network). To calculate the next decoder hidden state, we use $h^{(1)}_{\mathrm{argmax(a^i)}}$ as the next LSTM input. See Fig. 2 for the network diagram. Since we consider only sequences composed of 4 different nucleotides (nt), each base is represented as a 4-dimensional one-hot vector. Moreover, during the inference we implement and use a beam search heuristic to find the best possible sequence given a beam width.

| Dataset | BPs | Number of introns | Intron percentage |
|---|---|---|---|
| Mercer et al. 2015 | 59,359 | 36,388 | 17.4% |
| Taggart et al. 2017 | 36,078 | 22,020 | 16.8% |
| Pineda et al. 2018 | 130,294 | 66,711 | 37% |

TABLE I: Existing human BP annotation datasets.

## III. Experimental results

In this section we present and interpret the results of our model. We use the three publicly available datasets from Mercer et al. [19], Taggart et al. [9] and Pineda et al. [8]. Table I provides an overview of the existing human BP annotation datasets in terms of the total number of identified BPs, the number and percentage of introns with one or many experimentally verified BPs. Fig. 3 shows the number of unique and common sequences in these datasets. In total, there are approximately $81K$ intronic sequences with at least a single experimentally verified BP.

For training we used Pineda dataset to include as many sequences as possible, and for testing we used the subsets of Mercer and Taggart that are disjoint from the training set as illustrated in Fig. 3. Pineda dataset is imbalanced, with over 65% of the sequences having a single annotated BP. Since we are interested in the multiple-BP prediction problem, we generated a "Pineda balanced dataset" where we included all sequences with multiple BPs, and uniformly at random we included half of the sequences with a single BP. Balancing the dataset encourages the model to properly learn to predict either single or multiple BPs. Fig. 4 shows the ratios of sequences with single versus multiple BPs in each dataset. We split the training dataset following the 80/20 rule, where 80% of the sequences are used for training and the rest is held out for validation and choosing the optimal model.Since most BPs occur in the branchpoint area, i.e., 18 to 44 nt upstream of the 3'ss, we consider sequences where a BP is located 5 to 60 nt upstream of a 3'ss and extract 70 nt-long sequences as in [12] which are then used as inputs to the network.

To measure model performance we use *precision* and *recall* metrics:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

$$\text{Recall} = \frac{TP}{TP + FN}, \tag{3}$$

where $TP$ is the count of predicted BPs matching the true (i.e., experimentally verified) BPs, $FP$ is the count of predicted BPs not matching true BPs and $FN$ is the count of true BPs not predicted by the model.

Model training was conducted on a workstation with an AMD Threadripper 3990X processor, 256 GB of DDR4 memory and two GeForce RTX 3090 graphic cards running Ubuntu 20.04.3 LTS. The network is implemented using Python 3.8.8 and the deep learning library PyTorch 1.10 to fully utilize available graphic cards. Code and results are publicly available at https://gitlab.com/ajovanov/rna-branchpoint-predictor.

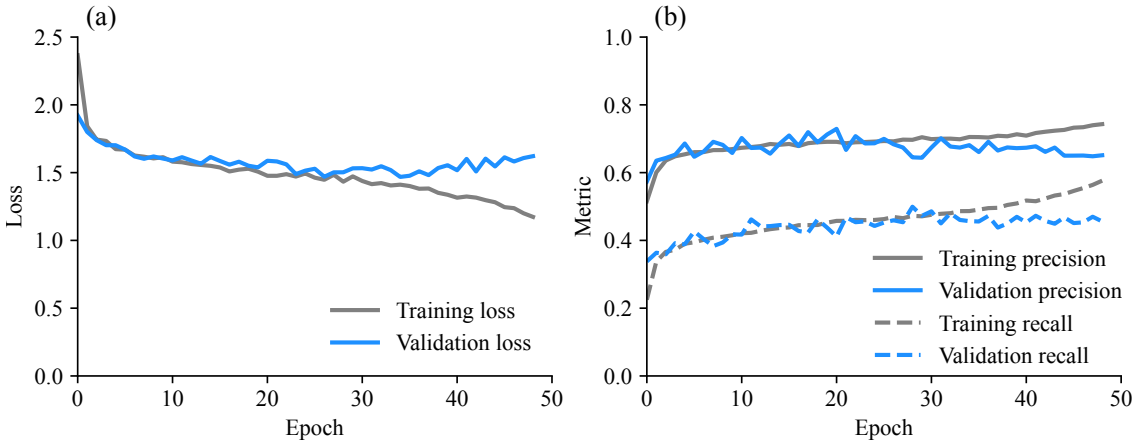


Fig. 5: The performance of our model in each epoch during training. (a) Evaluation of the loss value in training and validation sets. (b) Performance evaluation using precision and recall on training set versus validation set.

### A. *Hyperparameters, teacher forcing and training*

We trained the model for a maximum of 200 epochs with batch size 256 and AdamW optimizer with default learning rate $10^{-3}$ and weight decay $10^{-5}$. The encoder hidden state dimension is 128, while the decoder hidden state dimension is 256 to account for encoder hidden state concatenation. For calculating loss we used the cross entropy loss function.

Teacher forcing [15] is an RNN training technique which uses the last ground truth element $y_t$ instead of the last output $\hat{y}_t$ to generate the next output $\hat{y}_{t+1}$. However, during inference we can only supply the last output as the true output is unknown, which leads to large discrepancies between training and inference. To reduce this discrepancy, we use a curriculum learning strategy [16] and choose whether to use the ground truth or the generated output with teacher forcing ratio, i.e., probability of 0.5 for each batch.

During training we used a patience parameter of 20 to stop training if we see no improvement in precision and recall on the validation set. Training stopped after 49 epochs, which means that maximum precision and recall were achieved on epoch 29 or earlier. Figs. 5 (a) and (b) show signs of overfitting after epoch 30.

### B. *Model performance*

For testing the model performance we considered two model states, i.e., when maximum precision and maximum recall are achieved on the validation set. For inference we used a beam search heuristic with a beam width of 4. We also benchmarked against the LaBranchoR model trained on the same balanced Pineda dataset. LaBranchoR focuses on predicting the most likely BP associated with each 3'ss, while RNA PtrNets model aims to predict all BPs associated with each 3'ss. Our task is more challenging, therefore, this comparative benchmark analysis aims solely to set a baseline and show how our model compares to the state-of-the-art method of LaBranchoR. Fig. 6 shows an illustrative example from Mercer test set where RNA PtrNets correctly predicts all BP labels, while LaBranchoR only predicts a single BP.

| | RNA PtrNets High-precision | RNA PtrNets High-recall | LaBranchoR |
|---|---|---|---|
| Mercer dataset | | | |
| Precision | 0.642 | 0.537 | **0.663** |
| Precision ±2nt | **0.762** | 0.683 | 0.730 |
| Recall | 0.545 | **0.648** | 0.544 |
| Recall ±2nt | 0.680 | **0.786** | 0.568 |
| Taggart dataset | | | |
| Precision | 0.253 | 0.211 | **0.276** |
| Precision ±2nt | **0.611** | 0.553 | 0.575 |
| Recall | 0.252 | **0.295** | 0.266 |
| Recall ±2nt | 0.606 | **0.707** | 0.431 |

TABLE II: Precision and recall achieved by RNA PtrNets and LaBranchoR on mutually exclusive sequences from Mercer and Taggart test sets.

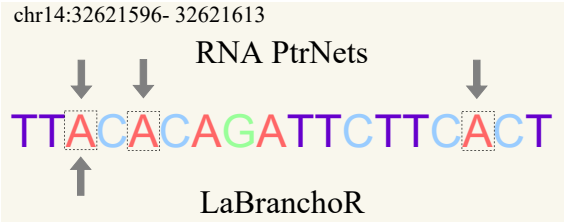


Fig. 6: An example from Mercer test set where RNA PtrNets correctly predicts all BP labels, while LaBranchoR predicts a single BP. True BPs appear inside the dashed rectangles. At the top, the arrows indicate the labels predicted by RNA PtrNets. At the bottom, the arrow indicates the label predicted by LaBranchoR. We only show a subsequence of the intronic segment that overlaps the genomic location shown on the top left corner.

Table II shows precision and recall on mutually exclusive sequences from Mercer and Taggart datasets (see Fig. 3). We notice a significant trade-off between metrics on relevant model states, where the model with high

precision predicts a single BP and the model with high recall predicts multiple BPs at the cost of producing many false positives. Considering that only 17.4% of introns have experimentally verified BP labels in the Mercer dataset, we expect that some predicted false positives are indeed positions missing from the annotated dataset due to the difficulties of labelling those BPs [8]. Compared to Mercer BP sites, Taggart BP sites generally disagreed with our predictions, resulting in lowered performance. Similar behaviour is reported in [12].

We also notice that predicted BPs often disagree with experimental ones by small shifts, so we report approximate precision and recall values labelled '$\pm$2nt', where we consider a predicted BP overlapping with an experimental one if it lies at most 2nt upstream or downstream of the experimental BP. We observe a significant increase in results on the Taggart dataset with this approximation.

Compared to LaBranchoR, our model achieves lower precision and higher recall on both datasets. When considering 2 positions upstream or downstream of experimental BPs, our model achieves both higher precision and recall.

## IV. Conclusion

RNA PtrNets extended LaBranchoR to handle output dictionaries whose size depends on the length of input indices, the outputs of RNA PtrNets model indicate pointers to the input indices. In this work, we aimed to develop a model for multiple BPs prediction. We generated a balanced training dataset from the Pineda BP set, which helped the model during training observe equal numbers of RNA sequences with multiple and with single BPs.

LaBranchoR applies a sigmoid function to output a probability density function over the input nucleotide indices. RNA PtrNets contains an LSTM and attention-based decoder whose outputs are pointers to different nucleotide positions.

RNA PtrNets performance fluctuates and there is a noticeable trade-off between precision and recall. When compared to LaBranchoR, RNA PtrNets model often provided higher recall at the cost of losing precision (i.e., it produced many false positives). Accurately predicting multiple BPs per 3'ss is more challenging than predicting a single BP label. RNA PtrNets results showed that our model predicted more multiple BPs per sequence compared to the experimentally verified ones perhaps due to missing BP labels from exiting annotation, e.g., in Mercer data set only 17.4% of introns have experimentally verified BP labels. Thus, we expect that a subset of these false positives to be missing labels from the original data sets.

## References


[1] A. Corvelo, M. Hallegger, C. W. Smith, and E. Eyras, "Genome-wide association between branch point properties and alternative splicing," *PLoS Comput Biol*, vol. 6, no. 11, p. e1001016, Nov 2010.

[2] S. Alsafadi, A. Houy, A. Battistella, T. Popova, M. Wassef, E. Henry, F. Tirode, A. Constantinou, S. Piperno-Neumann, S. Roman-Roman, M. Dutertre, and M. H. Stern, "Cancer-associated SF3B1 mutations affect alternative splicing by promoting alternative branchpoint usage," *Nat Commun*, vol. 7, p. 10615, Feb 2016.

[3] M. Burset, I. A. Seledtsov, and V. V. Solovyev, "Splicedb: database of canonical and non-canonical mammalian splice sites," *Nucleic Acids Res.*, vol. 29, no. 1, pp. 255–259, 2001. [Online]. Available: https://doi.org/10.1093/nar/29.1.255

[4] R. Castelo and R. Guigó, "Splice site identification by idlBNs," *Bioinformatics*, vol. 20 Suppl 1, pp. 69–76, Aug 2004.

[5] K. Jaganathan, S. Kyriazopoulou Panagiotopoulou, J. F. McRae, S. F. Darbandi, D. Knowles, Y. I. Li, J. A. Kosmicki, J. Arbelaez, W. Cui, G. B. Schwartz, E. D. Chow, E. Kanterakis, H. Gao, A. Kia, S. Batzoglou, S. J. Sanders, and K. K.-H. Farh, "Predicting splicing from primary sequence with deep learning," *Cell*, vol. 176, no. 3, pp. 535–548.e24, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0092867418316295

[6] R. Leman, P. Gaildrat, G. Le Gac, C. Ka, Y. Fichou, M. P. Audrezet, V. Caux-Moncoutier, S. M. Caputo, N. Boutry-Kryza, M. Léone, S. Mazoyer, F. Bonnet-Dorion, N. Sevenet, M. Guillaud-Bataille, E. Rouleau, B. Bressac-de Paillerets, B. Wappenschmidt, M. Rossing, D. Muller, V. Bourdon, F. Revillon, M. T. Parsons, A. Rousselin, G. Davy, G. Castelain, L. Castéra, J. Sokolowska, F. Coulet, C. Delnatte, C. Férec, A. B. Spurdle, A. Martins, S. Krieger, and C. Houdayer, "Novel diagnostic tool for prediction of variant spliceogenicity derived from a set of 395 combined in silico/in vitro studies: an international collaborative effort," *Nucleic Acids Res*, vol. 48, no. 3, pp. 1600–1601, 02 2020.

[7] I. A. F. Al-Qassem, "Computational methods for rna splicing," Ph.D. dissertation, LMU, 2021.

[8] J. M. B. Pineda and R. K. Bradley, "Most human introns are recognized via multiple and tissue-specific branchpoints," *Genes Dev*, vol. 32, no. 7-8, pp. 577–591, 04 2018.

[9] A. J. Taggart, C. L. Lin, B. Shrestha, C. Heintzelman, S. Kim, and W. G. Fairbrother, "Large-scale analysis of branchpoint usage across species and cell lines," *Genome Res*, vol. 27, no. 4, pp. 639–649, 04 2017.

[10] Q. Zhang, X. Fan, Y. Wang, M. A. Sun, J. Shao, and D. Guo, "BPP: a sequence-based algorithm for branch point prediction," *Bioinformatics*, vol. 33, no. 20, pp. 3166–3172, Oct 2017.

[11] B. Signal, B. S. Gloss, M. E. Dinger, and T. R. Mercer, "Machine learning annotation of human branchpoints," *Bioinformatics*, vol. 34, no. 6, pp. 920–927, 03 2018.

[12] J. M. Paggi and G. Bejerano, "A sequence-based, deep learning model accurately predicts RNA splicing branchpoints," *RNA*, vol. 24, no. 12, pp. 1647–1658, 12 2018.

[13] A. K. Gupta, T. Murthy, K. V. Paul, O. Ramirez, J. B. Fisher, S. Rao, A. B. Rosenberg, G. Seelig, A. C. Minella, and M. M. Pillai, "Degenerate minigene library analysis enables identification of altered branch point utilization by mutant splicing factor 3B1 (SF3B1)," *Nucleic Acids Res*, vol. 47, no. 2, pp. 970–980, 01 2019.

[14] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, p. 2692–2700.

[15] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Computation*, vol. 1, pp. 270–280, 1989.

[16] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, p. 1171–1179.

[17] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, Eds., vol. 27. Curran Associates, Inc., 2014. [Online]. Available: https://proceedings.neurips.cc/paper/2014/file/a14ac55a4f27472c5d894ec1c3c743d2-Paper.pdf

[18] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*,

Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1409.0473

[19] T. R. Mercer, M. B. Clark, S. B. Andersen, M. E. Brunck, W. Haerty, J. Crawford, R. J. Taft, L. K. Nielsen, M. E. Dinger, and J. S. Mattick, "Genome-wide discovery of human splicing branchpoints," *Genome research*, vol. 25, no. 2, pp. 290–303, 2015.