New Datasets and Improved Performance: Predicting Alternative Splicing Behaviour using Deep Learning

1044935

Department of Computer Science

University of Oxford

Submitted in partial completion of the

MSc in Computer Science

Trinity 2020

Abstract

Alternative splicing is a key means for genetic molecular diversity and its misregulation has been associated with up to 50% of all known pathogenic mutations, yet the regulatory mechanisms influencing it are still poorly understood. Better understanding of alternative splicing is in part driven by computational models which predict alternative splicing based on genomic sequence information.

In this work, we show that a dataset which was previously widely used for the quantification of alternative splicing is confounded such that 100-parameters models can match the performance of 20,000-parameter models, calling the meaningfulness of results using this dataset into question. To this end, we construct three new classes of datasets, each based on a different processing method, and show that only one of them provides high enough data quality for our task. We develop a new Deep Learning model which is the first to introduce an attention mechanism to the prediction of alternative splicing events. Evaluating our newly proposed model, we reimplement two models from the literature and find that it outperforms the previous state-of-the-art model for classifying alternative splicing events by 15%. Finally, we show that our model attends to biologically plausible regions of the input.

In summary, our work shows that previous results have been based on critically confounded data, provides better datasets upon which future work can build and introduces a new state-of-the-art model.

Contents

Glossary

Splicing . . . . . Conversion of pre-mRNA to mature mRNA through removal of introns and joining together of exons.

Exon . . . . . DNA sequence which is kept from the pre-mRNA during splicing.

Intron . . . . . DNA sequence which is removed from the pre-mRNA during splicing.

Non-coding sequence DNA sequence which does not encode protein sequences.

Junction . . . . . Site in a mature mRNA transcript where two exons border each other.

Cis-regulatory element Non-coding DNA region on the same DNA molecule as the gene ('close to the gene') it regulates.

PSI . . . . . Percent-spliced in, the proportion of transcripts containing a specific exon (or junction).

Motif . . . . . A widespread nucleotide sequence pattern conjectured to be biologically significant.

EST . . . . . Expressed sequence tag, a short cDNA sequence generated through Sanger sequencing.

GTEx . . . . . Genotype-Tissue Expression, a project providing a large repository of human-sequencing data.

iPSC . . . . . induced pluripotent stem cells, mature cells which have been reprogrammed to an immature pluripotent (undetermined) state.

HipSci . . . . . Human Induced Pluripotent Stem Cell Initiative, an initiative proving a large repository of iPSC-based sequencing data.

Splicing Code . . . . . A set of rules describing the regulatory mechanisms underlying splicing. Often implemented as a computational model.

DSC . . . . . Deep Splicing Code, a model used for constitutive exon classification.

Why genes in pieces?

— Walter Gilbert [1], in response to the discovery of splicing

1 Introduction

"One gene, one protein" was the prevailing dogma for over 30 years. Yet this belief was turned upside down by the stunning discovery in 1977 that genes contain long non-coding sequences [2] (see Figure 1.1). Instead, these non-coding sequences (introns) must be removed and only the coding sequences (exons) are joined together to form the blueprint of a protein. Different exons may be spliced together under different circumstances, allowing one gene to encode multiple proteins. Splicing, the removal of introns, and alternative splicing, the combination of different exons, were discovered.

But "Why genes in pieces?". This question was posed by Walter Gilbert one year later [1]. Gilbert answered his own question and proposed that splicing, in essence, provides a multiplier which speeds up the rate of evolutionary adaption: a single nucleotide mutation influencing splicing may now lead to the inclusion or exclusion of whole genomic regions and thus the generation of a different protein. Was Gilbert correct? 42 years later we are still uncertain, but some support for his ideas have been found [3].

What we have gained in the past four decades of research, is a greater appreciation of the importance and the complexity of splicing. Misregulated alternative splicing

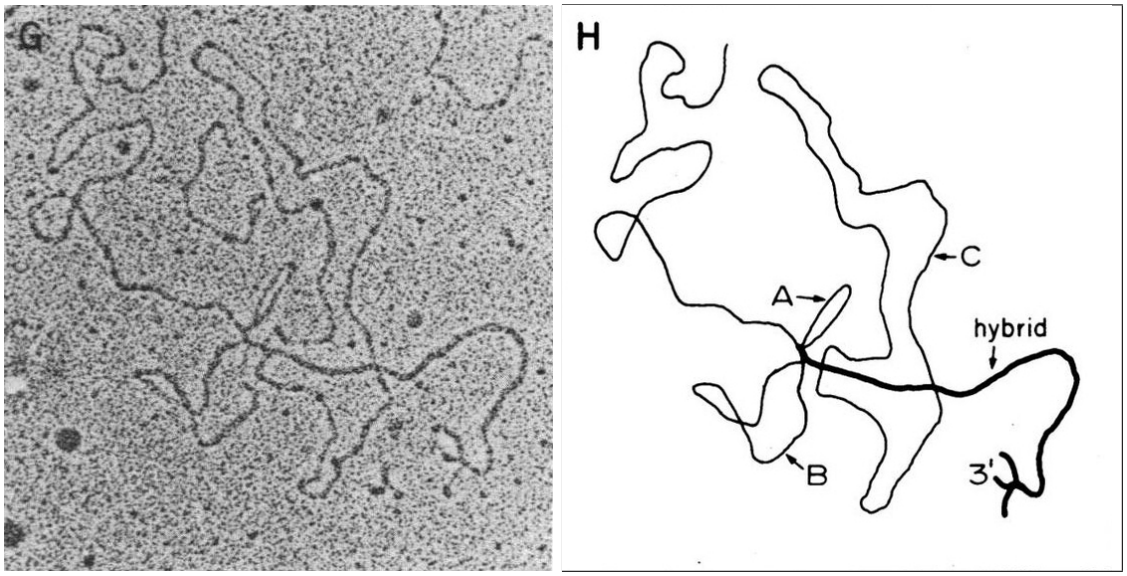

Figure 1.1: Original electron microspy leading to the discovery of splicing (left) and corresponding schematic (right) [2]. Introns (non-coding sequences) show up as the loops A, B and C in this illustration.

is a biomarker for several forms of cancer [4] [5] and up to 50% of known disease-causing mutations in humans have been shown to affect splicing [6]. Yet, our understanding of splicing is still limited: the core splicing signals which we have found are estimated to only account for approximately 50% of splicing behaviour even in limited circumstances [7]. The remaining 50% are likely accounted for by the complex interactions of numerous cis-regulatory elements.

An important long-term goal is the development of a splicing code which describes the regulatory mechanisms governing splicing and unifies them into a predictive model [8]. Due to splicing's complexity, attempts at these splicing codes have been increasingly based on computational models [9] and although many have been proposed, they are still too simplistic to explain the complex splicing behaviour we observe.

A parallel development has been the rise of deep learning fuelled by increasing data availability and the development of powerful GPUs [10]. Originating from Computer Vision [11], deep learning has brought about breakthroughs in tasks as disparate as the early detection of Alzheimer's disease [12] or estimating the demographic makeup of the US [13]. With the advent of next-generation sequencing, the amount

of genomic data available has increased by a multi-fold, making genomics a prime application domain for deep learning.

Our work lies at the intersection between the desire for better splicing codes and the revolution of many fields by deep learning. We develop a Recurrent Attentive Splicing Code (RASC) inspired by deep learning techniques known from Natural Language Processing (NLP). During evaluation, we find that we can replicate the performance of a 20,000-parameter model using a 100-parameter model which takes no sequence information as input when reusing a dataset from the literature. Motivated by this finding, we set out to develop better datasets and construct datasets based on three different processing methods. We demonstrate that only one of them provides appropriate data quality and quantity for the training of deep learning-based splicing codes. Comparing RASC against two reimplemented splicing codes, we show that it outperforms the previous state-of-the-art splicing code for classifying alternative splicing events by 15%.

The rest of the text is organized as follows:

- Chapter 2 succinctly introduces the biological background required to follow the rest of this work. It also gives more detail about the importance of splicing.
- Chapter 3 reviews the splicing codes which have already been developed.
- Chapter 4 describes the dataset construction and the evaluated models in detail. It is by far the longest and most theoretical chapter due to us developing new datasets and modifying the attention mechanism known from Transformer models.
- Chapter 5 presents the results of evaluating the models on a total of 16 different datasets. This includes the dataset which we show to be critically confounded and the datasets for whose future use we advocate.
- Chapter 6 summarizes our findings and draws more general conclusions. We also give suggestions for future work.

2

Biological background

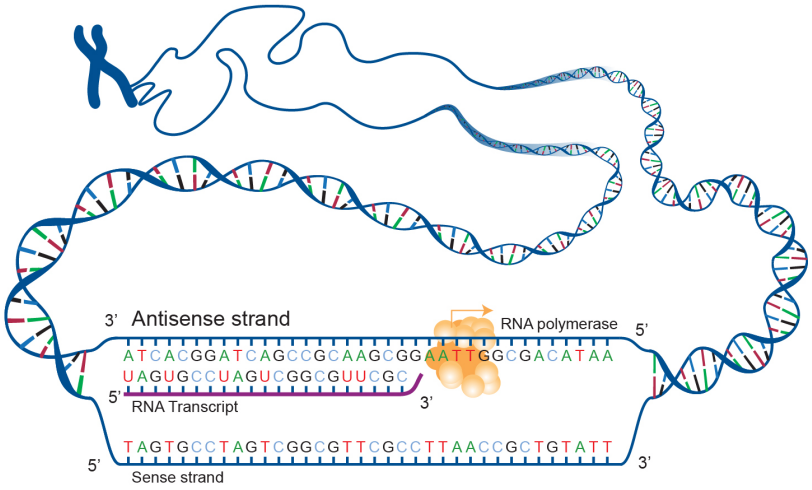

Figure 2.1: Transcription of DNA [14]. The enzyme RNA polymerase reads the DNA and constructs the complementary RNA transcript one nucleotide at a time.

Gene expression is fundamental to all life. It is the process whereby a segment of DNA, a gene, is used to direct the synthesis of a functional gene product (protein, functional RNA). Gene expression occurs in two steps: during transcription (see Figure 2.1), the DNA is transcribed into messenger RNA (mRNA) and during translation, the mRNA is decoded into gene products. A DNA strand consists of a sequence of the four nucleotides: adenine (A), thymine (T), cytosine (C), and

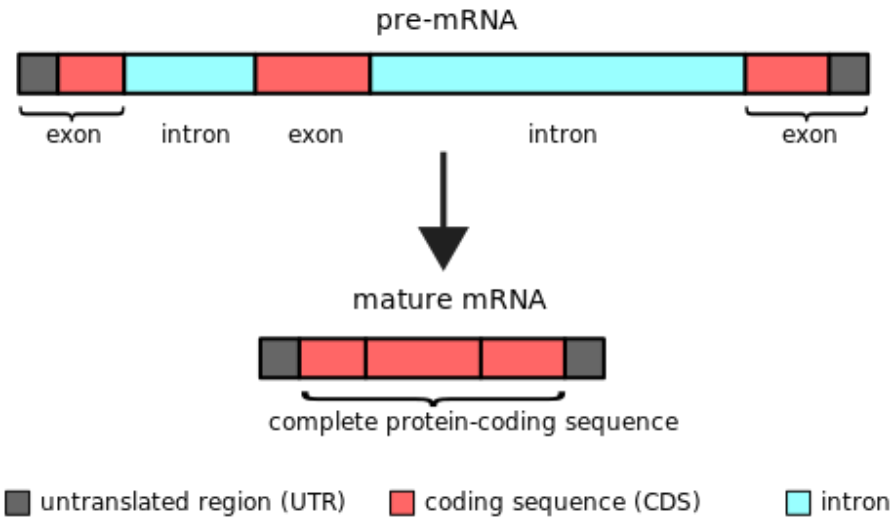

Figure 2.2: The process of splicing [16]. Introns are removed from the pre-mRNA to obtain the mature mRNA only consisting of exons. Apart from coding regions, exons may also consist of non-coding untranslated regions (UTRs). Like introns, UTRs influence gene expression.

guanine (G). The nucleotides adenine and thymine as well as cytosine and guanine form complementary base pairs, which are the building blocks of the double-stranded helix that makes up a DNA molecule. RNA is the same, except that thymine is replaced by uracil (U) and that most RNA molecules are single-stranded.

In more detail, during transcription, an initially transcribed precursor mRNA (pre-mRNA) is translated into a mature mRNA. The majority of sequence information (>90%) in the pre-mRNA is contained in long, non-coding regions, introns, which flank shorter, predominantly coding regions, exons. During splicing, the spliceosome removes the introns from the pre-mRNA and joins the exons together to form the mature mRNA (see Figure 2.2). The spliceosome is a complex molecular machine consisting of as many as 150 proteins [15]. The previous location of an intron in the transcript, where two exons adjoin each other, is called a splice junction or splice site.

Exons which are always included in the mRNA are called constitutive exons. However, 95% of human genes with multiple exons are alternatively spliced [9], that is, they may only sometimes be included, or may be included with different splice sites. Alternative splicing results in the generation of multiple transcripts (also transcript isoforms) from one gene.

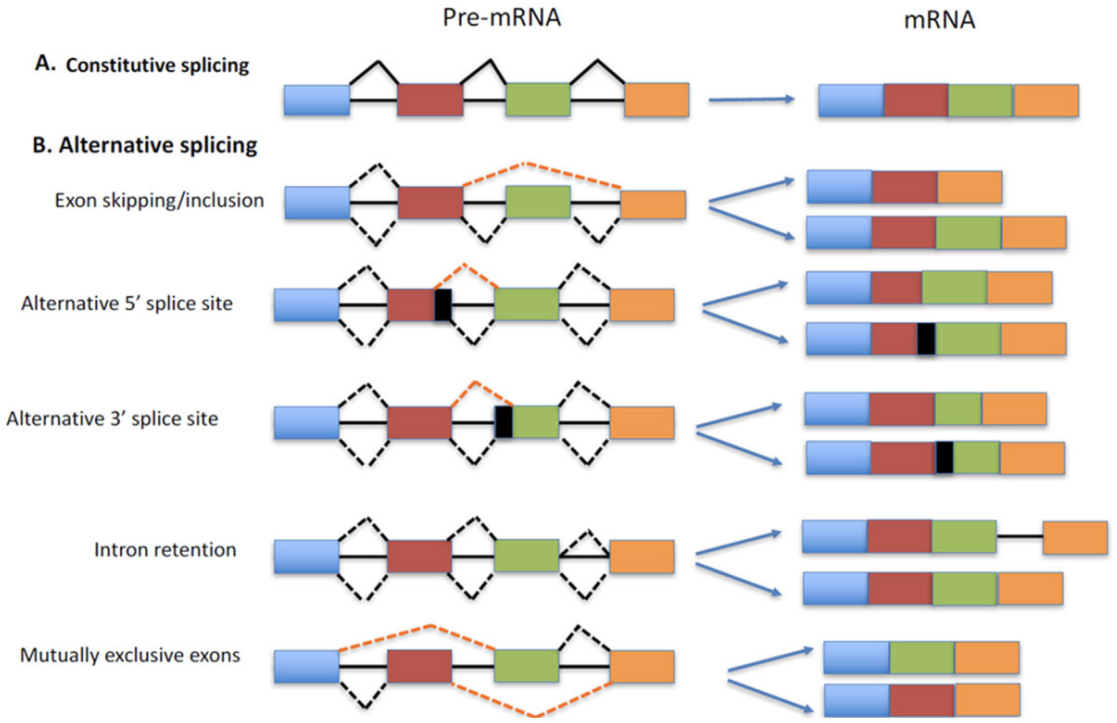

Figure 2.3: Different types of alternative splicing and the resulting different possible mature mRNAs [17].

The most common types of alternative splicing (visualized in Figure 2.3) in higher eukaryotes are [18][19]:

- Cassete exons are exons which are sometimes included in the mature mRNA and sometimes skipped. This is the most common form of alternative splicing in higher eukaryotes (so also humans), accounting for roughly 40% of all alternative splicing events [15].
- Exons with an alternative 3' or 5' splice-site. The 3' splice site is the end of the exon towards the 3' end of the RNA strand (typically visualized as falling to the right of the sequence). The 5' splice site is the end of the exon towards the 5' end of the RNA strand (typically visualized as falling to the left of the sequence). An alternative 3' or 5' splice-site may be located deeper inside the exon or outside the exon in a typically intronic region. Alternative 3' and 5' site splicing respectively constitute approximately 18% and 8% of all alternative splicing events in higher eukaryotes [15].

- intron retention, that is, when an intron between exons is not spliced out. It accounts for roughly 5% of alternative splicing activity in higher eukaryotes [15].

More complex forms of alternative splicing, such as mutually exclusive exons, also exist, but they are believed to be less common. Alternative splicing occurs in nearly all organisms that carry out pre-mRNA splicing such as plants or animals and its frequency varies across organisms [15].

Why does alternative splicing occur?

Alternative splicing enables a single gene to encode multiple protein variants. This massively contributes to proteomic diversity. For instance, the roughly 20,000 human protein-coding genes are estimated to encode over 100,000 different proteins [15].

Alternative splicing may also speed up the rate of evolutionary adaption [1] [3]. Due to alternative splicing, a single nucleotide mutation may affect the inclusion or exclusion of whole genomic regions. Additionally, without alternative splicing, a gene would first need to be duplicated before it could evolve to fulfil a different functionality if its previous functionality should be preserved. With alternative splicing, a gene can evolve to express different transcripts which fulfill both, old and new, functionalities.

How is alternative splicing regulated?

Alternative splicing was discovered 40 years ago [2], but the molecular mechanisms governing it are still poorly understood. It is known that the spliceosome recognizes exon-intron boundaries based on the 5' and 3' splice sites, the branch site located around the mid point of the exon, and the polypyrimidine tract located upstream of the 3' splice site. However, estimates suggest that these four factors only account for half of the information required to determine splicing behaviour. The rest is likely accounted for by intronic or exonic, cis-acting sequences of the pre-mRNA which bind to trans-acting factors. These cis-acting sequences are usually 4-18

nucleotide long and classified as exonic splicing enhancers or silencers [15]. However, we only have a limited understanding of the extremely complex, dynamic interaction between cis-acting and trans-acting factors [15], new factors are still being found [15] and thus a lot more work remains to be done.

What happens when splicing is misregulated?

Since alternative splicing is such a fundamental mechanism, its correct execution is crucial. Defects in splicing are typically caused by genomic sequence variations leading to misregulation of the splicing process [20]. An estimated 9%-30% of Mendelian disorders may act through disruption of splicing [21] and up to 50% of known disease-mutations in humans are associated with it [6]. Splice variants are biomarkers for multiple types of cancers [4] [5]. As a result, alternative splicing has also been suggested as a biomarker and potential target for drug discovery [22].

Application areas

Due to alternative splicing's role in disease-causing mutations, there is great interest in better understanding its mechanisms. One possible application of a better understanding would be the field of personalized medicine [23]: because of rapid advances in RNA sequencing technologies, it is now possible to sequence the genome of a patient within a day. However, the genomic variants (compared to a reference genome) observed in patients are often variants of unknown significance. That is, it is unknown whether these variants are pathogenic or benign. A better understanding of alternative splicing may improve the classification of genomic variants and help with the diagnosis of patients, especially those with rare genomic diseases.

Alles Gescheite ist schon gedacht worden.
Man muss nur versuchen, es noch einmal zu denken.

All intelligent thoughts have already been thought;
what is necessary is only to try to think them again.

— Johann Wolfgang von Goethe [24]

3 Related work

3.1 The first generation of splicing codes

Computational splicing codes are models that attempt to predict splicing behaviour based on putative regulatory features (such as sequence motifs). They were first introduced in the seminal papers by [9][25]. Their introduction was motivated by the recognition that splicing is highly condition-specific and regulated by the complex interaction of many factors in such a way that it is only feasible to model this behaviour computationally.

[9] focus on cassette exons and attempt to predict the change in splicing behaviour for a given exon between different tissues. They popularized the use of the quantitative measure PSI (sometimes denoted Ψ) to describe splicing behaviour: PSI is defined as the proportion of transcripts out of all transcripts that contain a given exon [26]. In other words, given a random transcript, PSI denotes the probability of a particular exon being included or excluded.

To quantify the change of splicing behaviour between conditions, these models predict the corresponding ΔPSI. Their input are over 1000 known and unknown motifs and higher-level features (such as exon/intron lengths and phylogenetic conversation scores) selected partially from previous studies and partially from de

novo searches. Successful application of these models lead to the discovery of novel regulators of key genes associated with diseases and to more accurate predictions about how genetic variants will affect splicing [27] [28].

3.2 The second generation of splicing codes

Improving upon these first models, the 'second generation' of computational splicing codes used several common and uncommon machine learning algorithms such as multinomial logistic regression, support vector machines (SVM) and Bayesian Neural Networks (BNN) to predict changes in alternative splicing behaviour [29]. Among these, BNNs were able to outperform the other methods when evaluated on a microarray dataset based on mouse data. In contrast to models from the first generation, BNN-based models only took in sequence information and very high-level features like tissue type which meant that the models were automatically able to learn relevant motifs from the data.

3.3 The third generation of splicing codes

3.3.1 Splicing codes for regressing PSI

However, BNNs often rely on expensive sampling methods like Markov Chain Monte Carlo (MCMC) to be able to sample models from a posterior distribution. It can be challenging to scale these methods to larger datasets and a large number of hidden variables. As a result, the 'third generation' of computational splicing codes relies on deep learning models which can effectively make use of the large amount of data available with the advent of high-throughput RNA-sequencing technologies. First forays into using deep learning-based models were made by [30]. Using a Deep Neural Network (DNN) with an autoencoder, they were able to improve upon the results achieved by BNN models. Albeit [30] initially used a different dataset and a different task formulation than [29], [31] were able to show that these improvements also lasted when directly comparing the models on the same dataset using the same

task formulation. Furthermore, [31] developed a framework for integrating further experimental data, like data from CLIP-seq based measurements of in vivo splice factors bindings, into the model developed by [30]. Adding these additional features improved the explained variance in splicing behaviour between tissues, as measured by the R^2 score, by roughly further 5% to an overall average value of 43.4%. Taking inspiration from advances in Natural Language Processing, [32] developed splicing codes based on the automated feature learning approach from word2vec [33] [34] and doc2vec [35] [36]. Developing two models, one based on doc2vec and a simple MLP, and one based on word2vec and the all-convolutional Inception architecture known from Computer Vision [37], they were able to achieve an average R^2 score of 69.2% significantly improving upon the predictive power of previous models.

3.3.2 Splicing codes for classifying splicing events

In contrast to these splicing codes which predict the (differential) inclusion frequency of an exon, a parallel strand of research focuses on splicing codes for distinguishing between constitutive and alternatively spliced exons. For the first task, the dataset the models are trained on only consists of alternatively spliced cassette exons and the models have to find features that are predictive of the exact inclusion rate of an exon. For the second task, the dataset consists of alternatively spliced as well as constitutive exons and the models have to find features predictive for distinguishing between constitutive and alternatively spliced exons.

While there is a large overlap between these features, there are also differences. For predicting the inclusion level of an exon, features from the cassette exon and the surrounding exons have reported to be required [27]. In contrast, for predicting whether an exon is constitutive or not, features around the cassette exon itself have been shown to be the most critical [38].

Originating from the second strand of research, [39] used 262 features extracted from an exon and its two flanking introns to train an SVM-based splicing code for distinguishing between constitutive exons, cassette exons and exons with an

alternative 5' or 3' splice site. The dataset used to train the model was based on roughly 4 million ESTs and known isoforms, as well as the alternative events, track (Alt Events) of the UCSC Genome Browser.

Their model achieved very impressive results with an AUC of roughly 0.94 when differentiating between rarely included and constitutive exons, but performance decreases to roughly 0.60 when distinguishing between frequently included and constitutive exons. [40] improved upon this work by using a deep learning model which was automatically able to learn relevant features from the raw sequence. Their model was based on a combination of convolutional blocks for feature extraction as well as an MLP for classification based on the extracted features. Training on a similar EST-based dataset, their model is significantly more robust when distinguishing between highly included cassette exons and constitutive exons with the AUC only dropping to 0.85. When distinguishing between rarely included cassette and constitutive exons, it was still able to achieve an impressive AUC of 0.92.

4 Methods

In this chapter, we define the exact tasks we are trying to solve, introduce the primary data sources and describe how these were used to obtain the training datasets. Additionally, we motivate and explain the evaluated models, and describe their implementation and training.

4.1 Task formulation

We consider a classification and regression task. For the classification task, the models predict whether an exon or junction is constitutive or not. Strictly speaking, all exons and junctions with a PSI of less than 100% are non-constitutive. However, to account for spurious reads resulting from low-quality alignments, we define all exons and junctions with PSI $\geq 99\%$ as constitutive and apply this definition to our training samples. Thus, the models are trained on the classified training samples and predict the classification of an exon or junction. For the regression task, the models are directly trained on, and also directly predict, the PSI value of an exon or junction. For either task, the output of the models is a prediction $\hat{y} \in [0, 1]$.

We focus on the easier classification task at first and only evaluate the models on the more challenging regression task once we have achieved a good predictive

power for the classification task.

Previous work indicates that advances in predicting one type of alternative splicing behaviour (e.g. cassette exons) via Machine Learning methods also translates to advances in prediction for other types (e.g. exons with an alternative 3' splice site) [40] [39]. Therefore, we only focus on one splicing type as this reduces the number of experiments and needed training time by two to three factors. In particular, we focus on cassette exons as they are the most common form of alternative splicing (accounting for 40% of all alternative splicing events [15]).

By a similar argument, the advances in predicting alternative splicing behaviour of junctions also translate to the same advances for exons. Junction-based datasets are interesting because they contain significantly more training samples than exon-based datasets (at least twice as many per definition). Deep learning algorithms are notorious for requiring large datasets to train and this enables us to test whether insufficient data quantity is a performance bottleneck when training on the exon-based datasets. Thus, we also evaluate the models on junction datasets.

4.2 Datasets

Three primary sources for genomic data were used in this study: the Human Exon splicing events (HEXEvent) database [41], data from the Genotype-Tissue Expression (GTEx) project [42] and data from the Human Induced Pluripotent Stem Cell Initiative (HipSci) [43]. Except for the HEXEvent database, none of these primary sources directly report the PSI values of exons or junctions. All of the data sources required further processing (e.g. extraction of corresponding nucleotide sequences) to obtain the final samples which are the input for the models. These further processing steps are described in Section 4.3.2.

4.2.1 HEXEvent database

The HEXEvent database contains genome-wide exon datasets of human internal exons which can be filtered for selected splicing events (e.g. constitutive or cassette exons). It was compiled based on known mRNA variants as defined by the UCSC Genome Browser for the human reference genome hg38 as well as their associated available expressed sequence tag (EST) information. ESTs are short cDNA sequences generated through Sanger sequencing (a first generation sequencing method) [44] with the primary aim to facilitate genome annotations. However, ESTs have largely been replaced by sequence reads generated from next generation sequencing technologies such as RNA-Seq which are able to generate vastly more reads at a fraction of the cost [45].

Two issues with ESTs are worth highlighting:

- ESTs are based on only a single sequencing pass and especially bases at the 3' and 5' end of the EST are known to be highly error-prone [46].
- ESTs underrepresent less frequent transcripts and as a result, often only capture 50 - 65% of an organism's genes [47].

Thus, the HEXEvent database should not be used without stringent filtering processes and only captures the splicing behaviour of the most frequently expressed genes.

4.2.2 GTEx

The GTEx project provides the most comprehensive database for tissue-specific gene expression and regulation available to-date; it contains over 17000 samples from nearly 1000 human donors. The sequencing data was obtained using mainly molecular assay-based techniques like Whole Genome Genotyping (WGS), Whole Exome Sequencing (WES) and RNA-Seq. Samples were taken from up to 54 different tissue sites. In particular, the tissue samples are also taken from less commonly seen tissue sites such as brain or heart as the GTEx project sources its

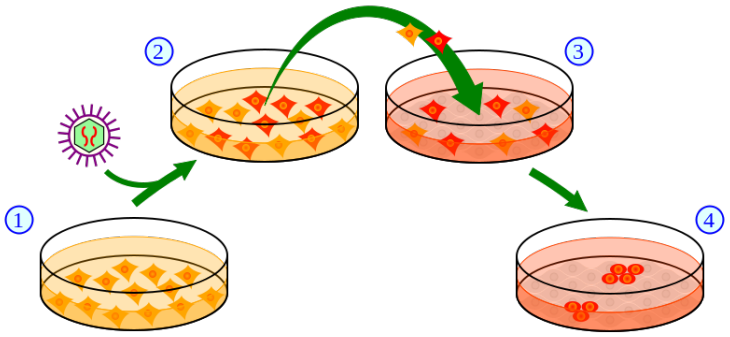

Figure 4.1: The making of iPSCs in four high-level steps [49]: (1) Grow a cell culture of donor cells, (2) transduce the genes associated with reprogramming into the cells via viral vectors, (3) isolate the cells expressing the transduced genes (in red) and culture them according to embryonic stem cell culture and, finally, (4) a small subset of the transfected cells become iPSCs.

samples from recently-deceased donors who have donated their body to science. Thus, the GTEx project, and by extension partially this thesis, was only made possible through the kindness and generosity of the donors.

Processed data which can not be used to identify the donors is publicly available on the GTEx portal. To access raw data (e.g. raw RNA-seq reads) and meta-information about the samples, one is required to undergo a data access request. It is intended that data access is requested by PIs or leader of research groups for their whole lab and approval of a data access request usually takes upwards of 3 months. Since the approval process would have finished very late into the project, if at all, we only use the publicly available part of the GTEx data. In particular, we use the files containing exon-exon junction read counts as accessible on the GTEx portal from GTEx Analysis V8 [48].

4.2.3 HipSci

HipSci provides a large repository of human induced pluripotent stem cell (iPSC) lines. iPSCs are mature cells which have been reprogrammed to again become immature pluripotent (undetermined) cells. This process is visualized in Figure 4.1. As iPSC lines are not directly obtained from human donors, but rather from

building a cell culture based on initial donor cells, accessing raw RNA-seq reads is not constrained by the same data privacy regulations which prevented access to the raw RNA-seq reads from the GTEx project.

Concretely, over 300 cell lines from over 300 donors along with sequencing information based on RNA-seq is publicly available from the HipSci portal. From these we selected two different groups of cell lines:

- The first group contains 25 biological replicates (that is, samples from different cell lines developed under the same conditions) of sensory neuron cell lines [50]. These cell lines were obtained by differentiating iPSCs to sensory neuron cells.
- The second group contains 20 biological replicates of iPSC lines which were not differentiated to any other cell type. Here care was taken to choose donors from whom at least two different iPSC lines were developed.

All used iPSCs were originally obtained from skin cells.

Selecting the samples in this way enables us to test cross-condition performance in three ways:

- On the same cell type from the same donor, but from a different cell line. Here we expect the best generalization performance.
- On the same cell type, but a different donor.
- On a different cell type and from a different donor. Here we expect the worst generalization performance.

The appendix contains the ENA Accession Numbers of the chosen samples in Section 7.1.

4.3 Data processing

4.3.1 Estimating PSI

Our primary data sources provide us access to short reads, that is, short inferred sequences of base-pairs of the mature mRNA transcript. Typical reads from our primary data sources are between 150 to 250 base pairs long [51]. We now discuss how to leverage these reads to obtain an accurate PSI estimate for exons and junctions.

Naive PSI estimation

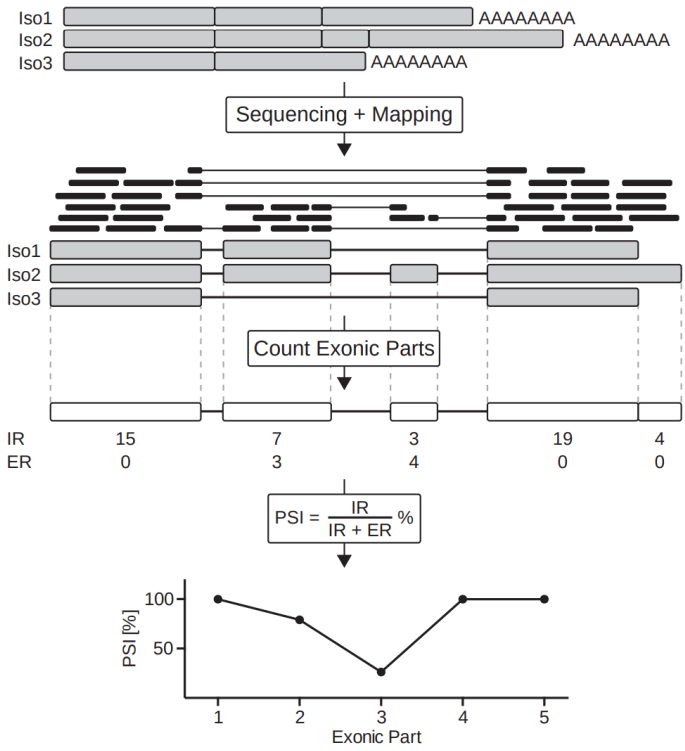

Figure 4.2: Process of estimating PSI based on reads [52]. First, the reads are mapped to the three known transcript isoforms Iso1, Iso2 and Iso3. Based on these, the reads overlapping exon/intron junctions are classified as including or excluding reads for a particular exon. Based on the observed IR and ER, the PSI for a given exon is then estimated.

Let #IR be the number of reads giving evidence for a particular exon being included. Let #ER be the numbers of reads giving evidence for a particular exon being excluded. A PSI value of 100% indicates a constitutive exon which

is always included, a score below 100% includes an alternatively spliced exon. PSI can then be estimated as:

$$PSI = \frac{\#IR}{(\#IR + \#ER)}$$

Figure 4.2 illustrates the process of estimating PSI based on reads. The advantage of this estimate lies in its simplicity and flexibility. It is quick and easy to implement (hopefully bug-free). It is independent of library size (where a library refers to the collection of all reads obtained from a particular biological sample). It can easily be adapted to estimate the PSI of a junction by redefining #IR and #ER to count the inclusion and exclusion read for that junction.

However, this estimate also has various issues:

(a) It does not account for uncertainty. A rarely expressed gene may only experience a few reads of a particular exon leading to a very uncertain estimate. If we observe 1 IR and 1 ER for exon E1, and 15 IR and 15 ER for exon E2, then the estimated PSI of both exons is 50%. The estimate of PSI_{E2} is likely to be more accurate, but this information is not contained in the estimate. Concretely, these two samples would be treated as equally important by the network even though they likely should not.

To account for this, it is possible to only count exons who experienced at least a certain number of reads. However, raw read counts are unreliable when not accounting for overall gene expression: 10 reads in a lowly expressed gene may be as significant as 20 reads in a highly expressed gene. There are several measures for the abundance of a gene: Reads Per Kilobase Million (RPKM), Fragments Per Kilobase Million (FPKM) and Transcripts Per Kilobase Million (TPM). Of these, TPM is usually the most common choice. Therefore, a better solution is likely to only count reads from genes whose TPM is above a certain threshold.

Note that there is not a principled way to choose this threshold. The choice between a threshold which, e.g., filters 5% or 20% of the training samples is a

trade-off between data quality and data quantity. In practice, the 'optimal' cut-off threshold will be unknown and vary from dataset to dataset.

(b) Reads aligning purely to the flanking constitutive exons are ignored when estimating the PSI for a cassette exon. While these reads could have occurred in both transcript isoforms, those containing and those excluding the cassette exon, they still provide latent evidence. In isoforms where the cassette exon is included, the total length of the isoform is longer which means that reads are distributed over a larger area. This leads to a comparatively reduced proportion of reads across the flanking exons when the exon was included and vice versa. The estimate neglects to take this information into account.

(c) Related to (a), typically multiple samples or biological replicates of a given experiment are available. It is desirable that reads across multiple samples can be integrated to give a more well-adjusted estimate. How to best achieve this is not obvious. Read depths between multiple libraries may vary and this should be accounted for.

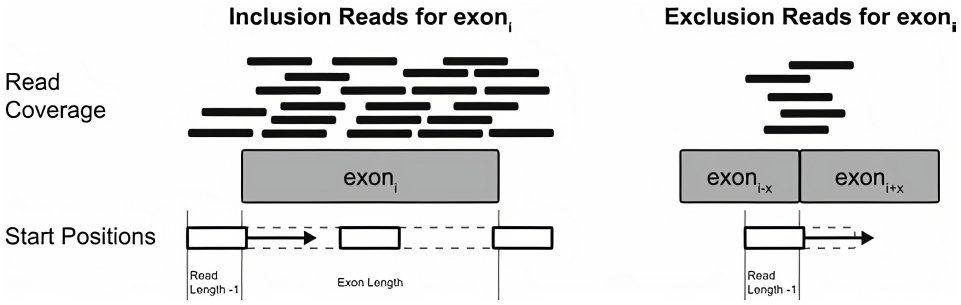

Figure 4.3: The need for read length normalization when estimating PSI [52]. An IR (left) can be located anywhere over the exon and around the junction. An ER (right) can only be located where it crosses the exon-exon junction. Thus, any PSI estimate which does not account for this will be biased towards estimating higher PSI values than accurate.

(d) IRs and ERs must be normalized for exon length to obtain meaningful results. For a long exon, the majority of reads will be IRs because they can be located over a much larger area than the ERs who must overlap with the 0-length feature of the splicing junction (see Figure 4.3). This can be accounted for by

normalizing for the possible number of start positions of each read population [52]:

$$\#IR_{norm} = \frac{\#IR}{exon\_length + read\_length - 1}$$

$$\#ER_{norm} = \frac{\#ER}{read\_length - 1}$$

(e) Reads may be misattributed. The motivation for analyzing splicing at a level of alternative splicing events is that full gene isoforms can not reliably be quantified given the currently available short RNA-seq read lengths. However, this issue may still occur at the splicing event level when two splicing variants of an exon share a junction. This is illustrated in Figure 4.4. This misattribution of a read can only be corrected by considering circumstantial evidence, similar to (b); for instance, read counts on the non-shared junction may give evidence regarding which splicing event the read at the shared junction belongs to.

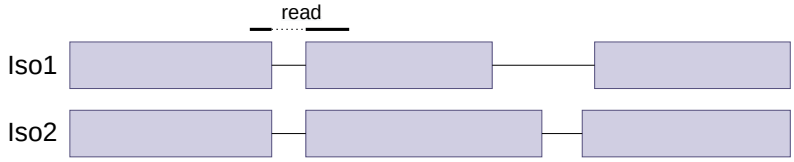

Figure 4.4: The read across the exon-exon junction may be attributed to either isoform 1 or isoform 2. Consequently, it may also be attributed to the first or second splicing variant of the middle exon.

Any estimate of PSI that does not account for all, or at least the majority of these issues, will likely be unreliable.

A word of caution

However, even if all of these issues are addressed optimally, the resulting PSI estimate would still be imperfect. For instance, the 'solutions' for (a) and (d) implicitly rely on the assumption that reads across a transcript are evenly distributed. It is well-known that sequence reads across a transcript are not equally distributed due to biases relating to GC-content [53], gene length and dinucleotide frequencies [54]. These biases are not easily quantifiable and can never be perfectly corrected for. Thus, the PSI estimate will always be affected by these biases.

In more general terms, the data on which the estimate is based on is biased in multiple complex, dynamically interacting ways and this will always lead to downstream effects independent of the preprocessing method used. The question is not whether the resulting estimate is still biased, but rather if the effect of systematic biases influencing it can be alleviated enough for the estimate to be useful. In this study, we answer this question in an empirical way by testing datasets resulting from four different PSI estimation methods. We now introduce the exact datasets we use and which PSI estimation method was used to obtain them.

Dataset based on HEXEvent

The HEXEvent database provides EST read-based PSI estimates for exons and facilitates standard filtering steps (e.g. only select cassette exons). The baseline paper [40] takes HEXEvent as starting point and generates three different datasets respectively only containing cassette exons, exons with an alternative 3' splice and exons with an alternative 5' splice site. Each of these datasets also contains constitutive exons. To account for the biases inherent in EST data, it filters training samples for a minimum number of supporting ESTs and only uses exons which display a single type of alternative splicing. They make the resulting datasets publicly available and we use the dataset containing cassette exons for direct comparability.

However, it should be noted that HEXEvent uses the criticized naive estimation method for PSI, i.e., they estimate PSI as $PSI = \frac{\#IR}{(\#IR+\#ER)}$. Except for requiring a minimum number of supporting ESTs, none of the five main critique points from Section 4.3.1 are explicitly accounted for by [40]. Additionally, the EST data available from UCSC is based on many different tissues which may lead to the derived alternative splicing behaviour being an average of the splicing behaviour of many tissues which never occurs in any singular tissue. Therefore, care should be taken when drawing conclusions based on the PSI estimates of this dataset.

Our PSI estimation implementation

To obtain a PSI estimate based on the publicly accessible exon-exon junction read counts of the GTEx project, we implement our own PSI estimation method. We are not able to use the PSI estimation methods from the literature we introduce below, as they require access to the raw RNA-seq reads which we do not have access to due to data privacy regulations.

GTEx publicly gives access to anonymized information about which sample types were taken from which donor. We use this information to find a donor from whom brain cortex tissue, cerebellum tissue and heart tissue samples were taken. For each tissue sample, we derive two datasets: one exon-centric dataset in which we estimate PSI for cassette exons and one intron-centric dataset in which we estimate PSI for junctions. We obtain six datasets in total.

Fundamentally, our PSI estimation relies on the formula $PSI = \frac{\#IR}{(\#IR+\#ER)}$. However, before computing PSI, we also apply the following filtering and normalization steps:

- Cassette exons which are shorter than 25 nucleotides and introns which are shorter than 80 nucleotides are excluded. Exons and introns of these lengths constitute less than 1% of exons and introns in the genome and are usually an artefact of sequencing errors [40] [55].
- GTEx provides access to TPM values for each gene from a sample. Addressing point (a) from Section 4.3.1, we obtain the TPM-adjusted read counts via $\#IR^{TPM} = \frac{\#IR}{TPM}$ and $\#ER^{TPM} = \frac{\#ER}{TPM}$. To address spurious reads from lowly-transcribed genes having a disproportionate impact, we only count reads from genes whose TPM is larger than 10. This filters around 84% of genes which in turn filters around 56% of the junctions. This is the most stringent filtering step we apply.
- Addressing point (d) about IRs naturally being more common than ER, we estimate the read length of the RNA-seq reads to be 150 on average [56] and

apply the proposed normalization method: $\#IR^{TPM}_{norm} = \frac{\#IR^{TPM}}{(exonlength+149)}$ and $\#ER^{TPM}_{norm} = \frac{\#ER^{TPM}}{149}$.

Using the normalized #IR and #ER of all exons and junctions remaining after filtering, we then compute PSI: $PSI = \frac{\#IR^{TPM}_{norm}}{(\#IR^{TPM}_{norm}+\#ER^{TPM}_{norm})}$.

Comparing our estimation method to the standards set out by 4.3.1, this estimate fails to take into account the information contained in non-junction reads, as we simply do not have access to them. This estimate also does not aggregate the information from multiple biological samples, although we took care to choose the donor with tissue samples from brain cortex, cerebellum and heart, whose total read count was the highest. Reads may also be misattributed in our cassette exon PSI estimation as described in (e). Note that misattribution is not an issue when estimating the PSI of junctions.

More sophisticated PSI estimation approaches

Several approaches have been developed which try to address the issues in estimating PSI. However, many of these focus purely on the differential splicing changes between conditions (in the form of Δ PSI) and do not directly report the PSI in a given condition which is what we are initially interested in. Of the methods directly reporting PSI, we chose two fast methods which represent different approaches to PSI estimation: SUPPA [57] and MAJIQ [58]. SUPPA is primarily based on quantification of transcripts, while MAJIQ is based on building up a splicing graph. MAJIQ has already been used for the construction of similar datasets before [31].

SUPPA

SUPPA estimates the PSI value for each alternative splicing event based on transcript abundance. It operates in 2 steps for quantifying the PSI of an alternative splicing event:

- Given an input annotation file in GTF format, it generates the transcript isoforms which count as IR or ER for a given alternative splicing event.

- Given the information which transcript count as IR or ER from the first step and how frequently each transcript occurs, it estimates the PSI value via $PSI = \frac{TPM_{IR}}{TPM_{IR}+TPM_{ER}}$. TPM_{IR} and TPM_{ER} respectively refer to the total TPM of transcripts supporting a certain exon being included or excluded.

SUPPA can integrate the TPM values from multiple samples.

Using SUPPA: SUPPA requires a GTF annotation file and a file quantifying the abundance of each transcript. A GENCODE version 34 annotation file obtained from Ensembl was used as an annotation file. Salmon [59] was used to quantify the relative abundance of each transcript in TPM. Salmon's quantification is based on the raw RNA-seq reads in FASTQ format, takes into account experimental attributes and corrects for biases commonly observed in RNA-seq data. After extracting the TPM values provided by Salmon into the format required by SUPPA (a TSV file with one column for the transcript ID and one column for the TPM value), SUPPA estimates the PSI for each alternative splicing event.

SUPPA's main competitive advantage for PSI estimation lies in its speed and low memory overhead compared to other methods [60] which it achieves through leveraging fast transcript quantification methods such as Salmon, not necessarily that it achieves the best accuracy. This becomes apparent when judging how well SUPPA addresses the issues we laid out in Section 4.3.1: of these addresses it only (c) — the integration of evidence from multiple samples. SUPPA focuses exclusively on alternatively spliced exons and thus we are not able to derive a list of constitutive exons from SUPPA. Since SUPPA operates at the transcript level, it implicitly relies on the assumption that all transcripts of a given gene are known — this is often not the case [61] [62]. Therefore, it remains to be seen whether the estimation of SUPPA is accurate enough for our purposes.

MAJIQ

Modeling Alternative Junction Inclusion Quantification (MAJIQ) builds up a splice graph which contains exon as vertices and junctions as edges. An edge is added

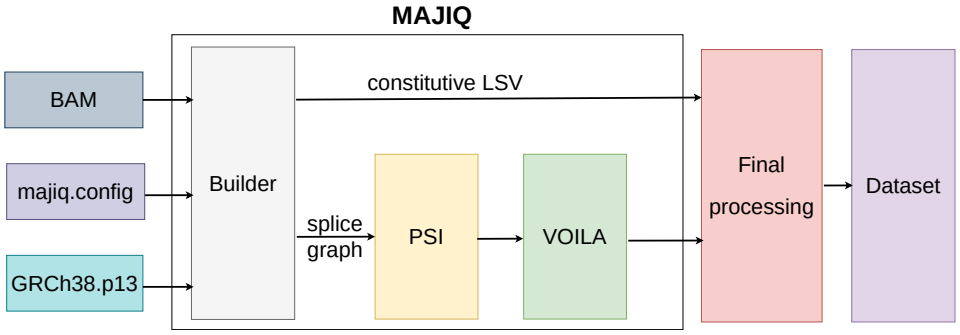

Figure 4.5: Data flow when using MAJIQ to create a training dataset.

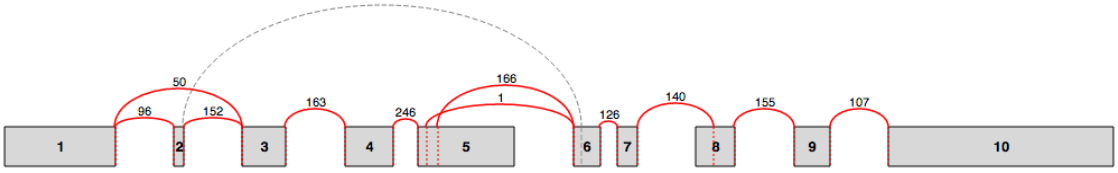

Figure 4.6: An example splice graph as displayed in VOILA [58]. Exons, represented by grey rectangles, are connected with another exon via an edge, if a junction between the exons (or a variant of them) is observed. The number above an edge displays the number of observed raw reads spanning a particular exon-exon junction.

between two vertices if a transcript isoform is found in which they share a junction (that is if the exons are neighbours and everything between them has been spliced out). Constitutive exons are vertices with two incoming edges. Vertices which have more than two incoming edges are denoted as local splicing variations (LSVs) and they represent exons involved in at least one alternative splicing event. This problem formulation allows MAJIQ to model more complex splicing events (rather than only the splicing events from Figure 2.3), although we will not leverage this extra capability here. Figure 4.6 shows an example splice graph. Importantly, apart from estimating Δ PSI between different conditions, MAJIQ is also able to directly estimate PSI for a given condition. The estimation is done using a combination of read rate modelling, Bayesian PSI modelling and bootstrapping. The only issue from Section 4.3.1 MAJIQ does not account for is (b): the integration of the information from non-junction reads. Thus, we expect MAJIQ's PSI estimates to be the most accurate of all methods we are testing.

Using MAJIQ: MAJIQ requires sequence files in BAM format (along with their

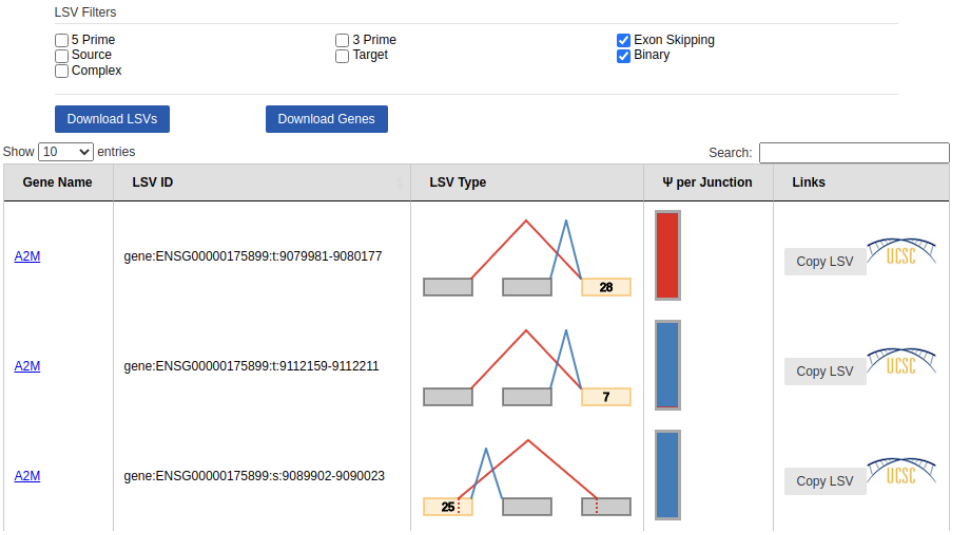

Figure 4.7: Example output of VOILA while filtering for LSV which only experience exon skipping and no other alternative splicing event.

respective index files) and an annotation DB (we used human GRCh38 release 13) in GFF3 format. The BAM format is a format for aligned RNA-seq reads. To this end, the raw reads from each sample were uploaded to the bioinformatics data processing platform Galaxy [63] from the European Nucleotide Archive (ENA). For each sample, the raw reads in FASTQ format were then mapped to the reference genome GRCh38 [64] using STAR [65]. STAR produces the required BAM and BAI format files as output.

MAJIQ use then proceeds in three stages. First MAJIQ Builder takes a configuration file, the GFF3 annotation and the BAM and BAI files of all samples as input and builds up the splicing graph. At this stage, MAJIQ also identifies constitutive exons and optionally saves them to a file. This option was used. Secondly, MAJIQ PSI estimates the PSI of the LSV candidates obtained through MAJIQ Builder. MAJIQ PSI improves the accuracy of its PSI estimate by integrating evidence across multiple samples. Finally, the obtained LSVs can be visualized using VOILA. See Figure 4.7 for an example output of VOILA. VOILA also allows filtering of LSV according to type and we used this to obtain the estimated PSI values of all exon skipping LSVs. The filtered LSVs, along with the constitutive exons obtained from MAJIQ Builder, were then used for processing as described in Section 4.3.2.

We refer to the datasets resulting from processing GTEx data with our PSI estimation method, from processing HipSci data with SUPPA and from processing HipSci data with MAJIQ as GTEx dataset, HipSci SUPPA dataset and HipSci MAJIQ dataset.

4.3.2 Final sample processing

At this point, the datasets have been preprocessed such that we have the following information for each training sample that we want to create:

- chromosome and strand information of the to-be-used exon or junction,
- start and end coordinates of the exon or junction within a chromosome,
- the lengths of neighbouring introns or exons[1].

Using this information, now either exon-centric or intron-centric training samples are created. Exon-centric training samples are created when the task of the network is to classify an exon or regress its PSI. Intron-centric training samples are created when the task of the network is to classify a junction or regress its PSI.

Note that for intron-centric training samples we do not classify the intron or regress its PSI because the corresponding PSI is estimated with respect to the inclusion or exclusion of a particular junction. To estimate the PSI of an intron, we would need to also account for all other junctions which correspond to a particular intronic region being spliced in or not.

Using the chromosome information, and the start and end coordinates, two 140 nucleotide windows around the start and end coordinates are extracted from the human reference genome GRCh38 (see Figure 4.8). The size of the nucleotide window is motivated by previous work [40]. We refer to this input as the sequence input or sequence features. Additionally, we abbreviate the sequence around the exon (or junction) start as start sequence. We abbreviate the end sequence analogously.

[1]Due to data limitations, we do not know the length of neighbouring exons when creating intron-centric training samples from GTEx data. In this case, the length of the neighbouring exons is set to 0.

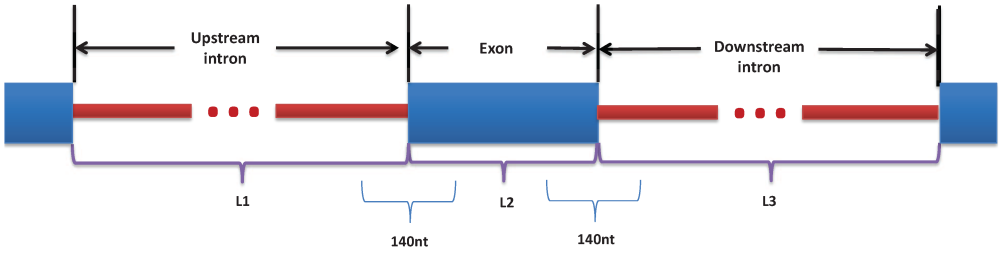

Figure 4.8: Schematic of model inputs for an exon-centric training sample [40]. The input are the 140 nucleotide windows extracted around the exon start and exon end, as well as the normalized length of the exon and its flanking introns. The inputs for intron-centric training samples (not shown) similarly consist of two 140 nucleotide windows around intron start and end, as well as the length features.

Furthermore, if an exon or junction is located on the negative strand, the extracted start and end windows are switched, the order of the nucleotides within the start and end windows are reversed, and each extracted nucleotide is converted to its reverse complement. This mirrors biology, as the spliceosome also observes the exon start and end sites as switched and reverse complemented between the positive and negative strand.

Samples from the chromosomes X, Y and M are excluded due to the fundamental functional differences between them and the autosomes (e.g., there only exists one functional copy of the sex chromosomes).

The extracted nucleotides are one-hot encoded as four dimensional vectors. Specifically, adenine 'A' is encoded as [1 0 0 0], cytosine 'C' as [0 1 0 0], guanine 'G' as [0 0 1 0] and thymine 'T' as [0 0 0 1]. Thus, each 140 nucleotide long window is converted into an 140x4 matrix containing one-hot encoded nucleotides.

It is commonplace that repetitive sequences are soft-masked as lower case letters in the reference genome. As this has no known bearing on alternative splicing, this information is ignored during one-hot encoding.

As in previous work [40] [66], we include the lengths of flanking exons or intron and the length of the exon or intron itself as additional input features. For brevity, we often refer to these exon and intron lengths as the length features. The inclusion of the length features is motivated by the observation that exon and intron lengths are

correlated with splicing behaviour [67] [68]. In particular, alternatively spliced exons tend to be flanked by longer introns. This may be due to long intronic sequences making it more challenging for the spliceosome to locate the exon or it plainly being more likely that novel exons originate within long introns [69].

Introns are on average one to two orders of magnitudes larger than exons and their relative standard deviation is three times as large as that of exons. To avoid giving features to the network whose magnitude might differ by several orders of magnitude, the intron and exon lengths features were normalized by the respective mean length and standard deviations of internal exons ($\mu = 145, \sigma = 198$) and introns ($\mu = 5340, \sigma = 17000$) in the human genome [70] [71].

Some of the datasets contain significantly more constitutive, than non-constitutive training samples. For instance, the HexEvent dataset contains roughly three times as many constitutive as alternatively spliced training samples. In these cases, we rebalanced the datasets by including each alternatively spliced training sample multiple times until the class imbalance was less than two-to-one.

The end result is that a single training sample for the model contains two 140x4 one-hot encoded matrixes and three normalized length values. This is the model input. The associated training label for the classification task is the scalar 1 if the respective exon or junction is constitutive, and 0 if not. The associated training label for the regression task is the PSI estimate.

4.3.3 Dataset statistics

Table 4.1 shows the number of training samples in each dataset. Among the exon-centric datasets, the HEXEvent and HipSci MAJIQ datasets contain comparatively more training samples than the HipSci SUPPA and GTEx datasets because they also contain constitutive non-cassette exons. In contrast to the sensory neuron and iPSC-based HipSci datasets, the magnitudes of the GTEx-based datasets vary significantly between tissues: the cerebellum tissue dataset contains roughly twice as many training samples as the heart tissue dataset. On closer investigation, this

Name	Tissue	Type	Misc	Number of samples
HEXEvent	mixed	exon	/	50,918
GTEx	brain cortex	exon	/	30,466
GTEx	cerebellum	exon	/	37,095
GTEx	heart	exon	/	19,257
GTEx	brain cortex	intron	/	127,908
GTEx	cerebellum	intron	/	161,310
GTEx	heart	intron	/	81,659
HipSci SUPPA	neuron	exon	/	17,863
HipSci MAJIQ	neuron	exon	/	44,746
HipSci MAJIQ	iPSC	exon	/	48,652
HipSci MAJIQ	iPSC	exon	diff lib	49,275
HipSci MAJIQ	iPSC	exon	diff indv	48,489

Table 4.1: Type, tissue and number of training samples per dataset. Type refers to whether the dataset is intron- or exon-centric. 'diff lib' denotes that the dataset was taken from the same individual, but a different library and 'diff indv' denotes that the dataset was taken from a different individual than the first shown HipSci MAJIQ iPSC-based dataset.

is because the cerebellum tissue contains 10,000 genes whereas the heart sample only contains 5,000. Surprisingly, the heart tissue is based on 17 million exon-exon junction reads compared to only 8 million exon-exon junction reads from the cerebellum tissue sample. However, the additional reads from the heart sample are concentrated in a few extremely highly expressed genes (e.g. gene ENSG00000198886 is expressed 1000 times as often as the median highly expressed gene) and while this concentration of reads also appear in the cerebellum tissue, it is less extreme. The same cross-tissue observations also hold for the intron-centric GTEx-based datasets.

In general, the intron-centric datasets contain roughly four times more training samples than the exon-centric datasets. This is because they do not only contain junctions of cassette exons (in which case they would contain roughly twice as many training samples as the exon-centric datasets) but contain all non-filtered junctions.

Figure 4.9 shows histograms for the exon-centric version of the four primary datasets versions. The histograms show that in each dataset the distribution of PSI values is bimodal with modes around very rarely included ($< 5\%$) and very frequently included ($>95\%$) exons. This aligns well with previous observations that alternatively spliced

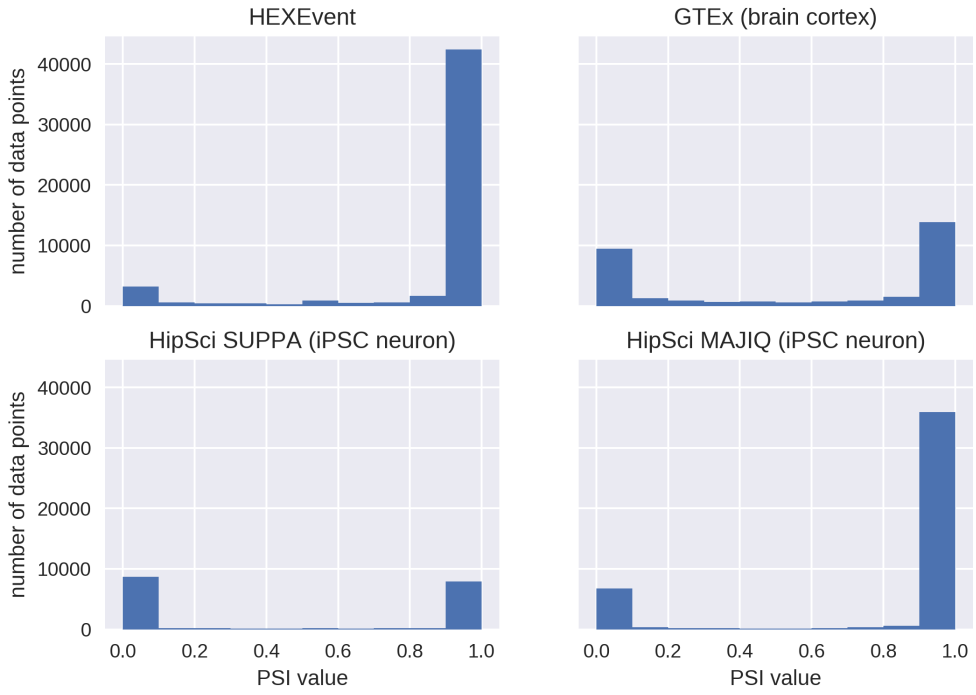

Figure 4.9: Histograms comparing the exon-centric versions of the four different training dataset classes. The shown GTEx dataset is based on samples from brain cortex tissue and the shown HipSci datasets are based on samples from iPSCs differentiated to sensory neuron cells.

junctions and exons tend to be either very frequently or very rarely included [39] [72] [73].

The lack of non-cassette constitutive exons is visible in the GTEx and HipSci SUPPA datasets containing significantly fewer constitutive samples; the other two primary datasets contain roughly three times as many constitutive as non-constitutive samples. In particular, the HipSci SUPPA dataset contains an extremely low number of non-rarely or frequently included exons (only $\sim$1%) which is likely an artefact of its coarse, transcription-level estimation method.

4.4 Models

We now introduce the three models which we evaluate in this work. They are all fundamentally split into two components:

1. The first component extracts the most relevant features from the two input sequences of nucleotides. Depending on the exact model, the feature extraction is based either on Convolutional Neural Networks (CNNs), word embeddings or Bidirectional Long-Short Term Memory Units (BiLSTMs) and an attention layer. In this way, the three evaluated models represent three different approaches to feature extraction from a nucleotide sequence. The extracted features are concatenated along with the length features and fed as input to the second component.

2. The second component uses the extracted features and length features to perform binary classification or regression. This component is implemented as a shallow multilayer perceptron (MLP) for all models. It consists of one or two fully connected layers followed by dropout [74] layers (to reduce overfitting) and ReLU [75] activation layers (to enable the network to model non-linear behaviour). A sigmoid activation function is used after the last layer to obtain an output between 0 and 1.

4.4.1 DSC: CNN-based

This model is a reimplementation of the Deep Splicing Code (DSC) from [40] and the current state-of-the-art model for the classification of alternative splicing events.

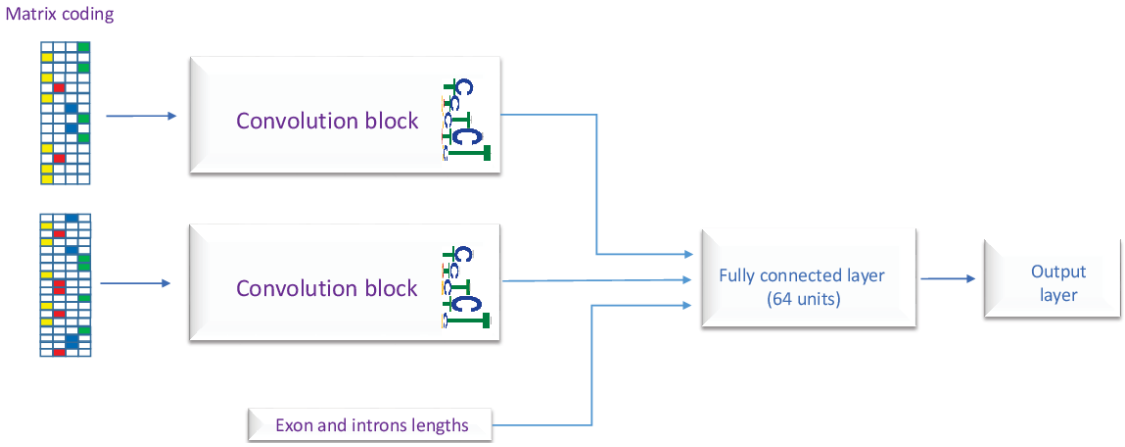

Figure 4.10: Architecture of DSC [40].

The motivation behind using CNNs

DSC uses CNNs for feature extraction. CNNs are useful when the input data contains spatially invariant patterns. Since images are fundamentally made up of small, spatially invariant patterns which can be combined into more complex patterns, CNNs have been extremely successful in Computer Vision. CNNs are also promising for the application to genomic data:

- Nucleotide sequences contain motifs (nucleotide sequence patterns conjectured to be biologically significant). Motifs are frequently represented as position weight matrixes (PWMs) which encode information about the significance of the occurrence of a particular nucleotide at a particular position. The kernels used in CNNs can be interpreted as PWMs with learnable weights. Thus, the inductive bias of CNNs, that the learning task can be solved via optimizing matrix weights, aligns well with our initial knowledge about the task of detecting motifs within a sequence.
- Motifs are found at different positions in the input sequence. This is not a natural problem setting for some Machine Learning algorithms since they would need to learn to recognize the same motif in different positions. However, CNNs are inherently spatially invariant and thus their inductive bias is again well-suited for detecting motifs within a sequence.

Model architecture

Keeping this motivation in mind, DSC extracts features from the two input sequences using CNN-based blocks. A CNN feature extraction block with the same architecture but independently learned weights is used for each input sequence. This is to accommodate the model learning to extract different features from the exon start and exon end sequence. The extracted features are concatenated with the lengths and used by a shallow MLP for predicting the output. The architecture of DSC is visualized in Figure 4.10.

Concretely, each CNN block consists of three convolutional layers. The first convolutional layer has a window size of 7 units and 32 filters, the second has a window size of 4 units and 8 filters and the third has a window size of 3 units and 8 filters. Each convolutional layer is followed by a dropout layer with dropout probability 0.2 to reduce overfitting, a ReLU activation layer and a max-pooling layer with stride and window size 2 to extract the most salient features. The hyperparameters for this model were chosen with grid search by [40]. 1D convolution layers are used.

Applying NLP techniques to genomic data

Like text, genomic data is fundamentally a sequence of characters. This makes it possible to apply techniques known from NLP to genomic data. This is an especially promising area of cross-pollination because of the large strides NLP has been able to make in recent years using deep learning. Some of the most important milestones in NLP have been efficient embeddings via word2vec [33] [34], applying Recurrent Neural Networkss (RNNs) to text as in Sequence-to-Sequence (seq2seq) learning [76] and leveraging attention as in the extremely powerful and deep Transformer models [77] [78].

In this work, we evaluate the application of doc2vec (a derivative of word2vec), seq2seq models and the attention mechanism for the classification of alternative splicing events and the regression of their frequency. We do not evaluate the current state-of-the-art Transformer models. Albeit very potent, they usually require huge datasets in the order of millions of samples which are not available for this task (this decision is discussed in more detail in Section 4.4.3). Nonetheless, we do adapt the attention mechanism as known from Transformers for our task. In this way, we feel the evaluated models make use of the most important deep learning innovations originating from NLP.

4.4.2 D2V: MLP-based

This model is a reimplementation of the model used for PSI regression on the mouse genome in [32] and based on the doc2vec word embedding algorithm. The only architectural difference is that we possibly use a differently-sized MLP in the second part of the network, as architectural details on the MLP are not given and no public implementation is available.

Introduction to word embeddings

Word2vec is a neural network-based algorithm which provides a continuously distributed (vector) representation for each word within a sentence [33][34]. Trained word2vec models have been shown to recover semantic and syntactic relationships. Given that each word is represented as a numerical vector, the following equation usually holds on trained word2vec models: $\vec{King} - \vec{Man} + \vec{Woman} \sim= \vec{Queen}$ where $\vec{x}$ represents the vector representation or embedding of word x. The key idea through which word2vec achieves this behaviour is by representing words based on the context they appear in. In particular, a shallow neural network is trained on a large corpus of unlabelled text with a loss which incentivizes the network to encode words by the context they occur in.

Doc2vec [35] [36] is an extension of word2vec which uses the same principles, but can also handle variable-length texts (such as paragraphs and complete documents). It returns a fixed-length representation independent of input size.

Training

To train word2vec or doc2vec, a large corpus of unlabelled data is needed. As we are training on genomic data, the largest corpus is the complete genome itself. To this end, we obtained the complete human reference genome GRCh38 from UCSC [79].

During training on text data, the corpus is split into documents which are ideally semantically meaningful (like paragraphs). Due to memory limitations and due

to corresponding to the average gene size [71], we split the genome into sequences of length 28,000. Tests with splitting the genome into sequences of length 10,000 showed that the effect of this choice on model performance is imperceptible.

Using k-mers

Naively applying word2vec or doc2vec to genomic data would mean treating each nucleotide as a word. Drawing the parallel to natural language, this would mean wanting to embed each character in a word. However, a single character or nucleotide (which comes from an alphabet of only four characters) is not unique enough and occurs in too many contexts to obtain meaningful representations. Therefore, it is desirable to base the embedding on multiple characters or multiple nucleotides, that is on words or k-mers. While a sentence can easily be split into words, the split of a nucleotide sequence into k-mers is not as clear-cut: we do not know the functionality of most nucleotides and therefore can not easily split a sequence into semantically meaningful units.

Previous studies have explored this issue [80] [81], and found that the compromise of splitting each nucleotide sequence into overlapping 3-mers is a performant choice. Overlapping means that given, for example, the sequence 'AACGAT', the resulting overlapping 3-mers are 'AAC', 'ACG', 'CGA' and 'GAT'. Based on these findings, we split each sequence of nucleotides into overlapping 3-mers and obtain 27,998 overlapping 3-mers from the 28,000 nucleotides long pre-training sequences.

Pre-training of word embeddings

The pre-training of word2vec and doc2vec uses a shallow neural network with one hidden layer. One of two techniques is typically used for pre-training word2vec: either continuous bag-of-words (CBOW) or skip-gram. If CBOW is used, all words in a context window except the current word are given as input and the target output is the current word. If the skip-gram technique is used, only the current word is given as input and the task is to predict the surrounding words in the window.

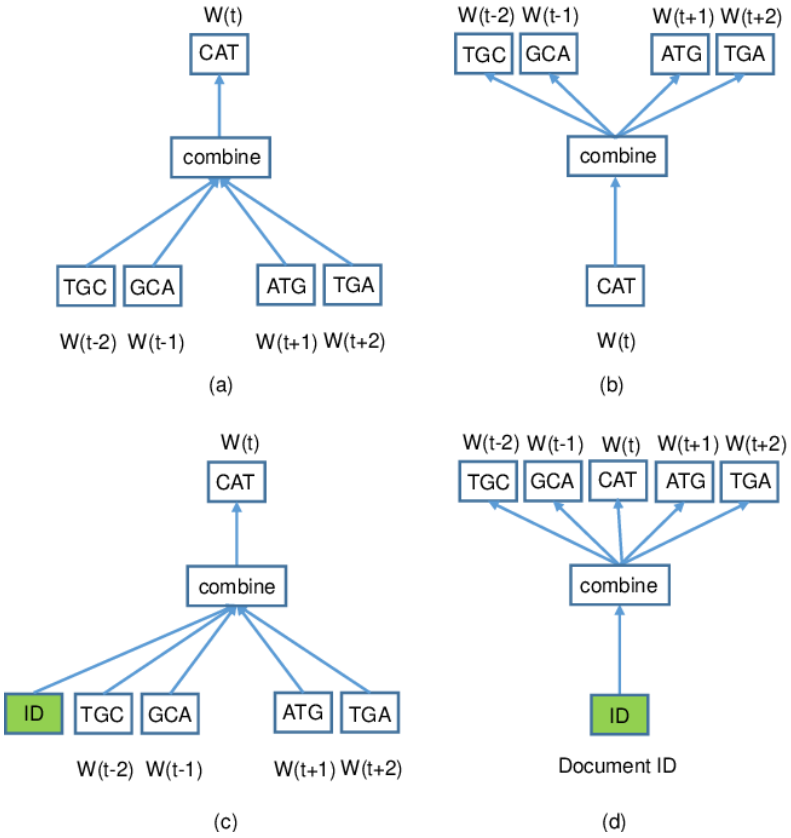

Figure 4.11: The four possible training algorithms for word2vec and doc2vec models [32]: (a) CBOW, and (b) skip-gram for word2vec, and (c) DM and (d) DBOW for doc2vec.

The context or sliding window around a word contains the words immediately adjacent to it. A typical window size is 5 words, meaning 5 words behind and after the current word will be used.

The pre-training of doc2vec works similarly. In the CBOW equivalent called distributed memory (DM), a document identifier (commonly simply an integer enumerating all documents) is given as additional input. In the skip-gram equivalent distributed bag of words (DBOW), the current word is replaced by the document identifier and the task is to predict all words in the window. All four different training methods are visualized in Figure 4.11.

There is no clear guideline for when which training method should be used for word2vec or doc2vec as they tend to perform similarly. However, DM (and also CBOW) preserve the order of words and as the order of words (3-mers) is likely significant, we chose DM as pre-training method.

Pre-processing the human genome like described above and using the DM training method, we trained a doc2vec model to output a 100-dimensional embedding for each document.

Obtaing the embeddings

After pre-training, the embedding for a given word or document is the value of the hidden cells of the shallow pre-training MLP[2]. In the case of 100-dimensional embeddings, this means that the hidden layer contains 100 neurons. The intuition behind this is that the models learned a representation which is helpful for the prediction task the model was trained on. Since the prediction task was either to reconstruct a word or document given its context or vice-versa, the representation for a given word or document should encode its context. Thus, the main idea behind word2vec and doc2vec, that a word or document is defined by the context in which it occurs (and its representation should encode this context), is achieved.

Analyzing the obtained embeddings

We visualize the embeddings for all 64 possible 3-mers using Stochastic Neighbor Embedding (t-SNE) [82] in Figure 4.12.

Doc2vec is trained to map words (3-mers) which occur in a similar context to similar representations. We observe that this often translates to mapping the same amino acids to similar representations. There are some instances when different amino acids are mapped together like the quartet Alanine (Ala), Threonine (Thr), Proline (Pro) and Serine (Ser) in the top-left corner. Why are exactly these different amino acids mapped together? The associated 3-mers reveal that these different amino acids share two nucleotides at the same position. This probably leads to these amino acids appearing in similar contexts. Overall, we take these

[2]In practice, the words in a corpus are one-hot encoded and the value of the hidden layers does not need to be computed. Recalling that the weights of the hidden layer of an MLP are represented as a $N \times d$ matrix, only the d-dimensional column which belongs to the respective one-hot encoded word needs to be retrieved. All other columns will be multiplied by 0. N represents the size of the vocabulary (64 for us) and d the number of embedding dimensions (100 for us).

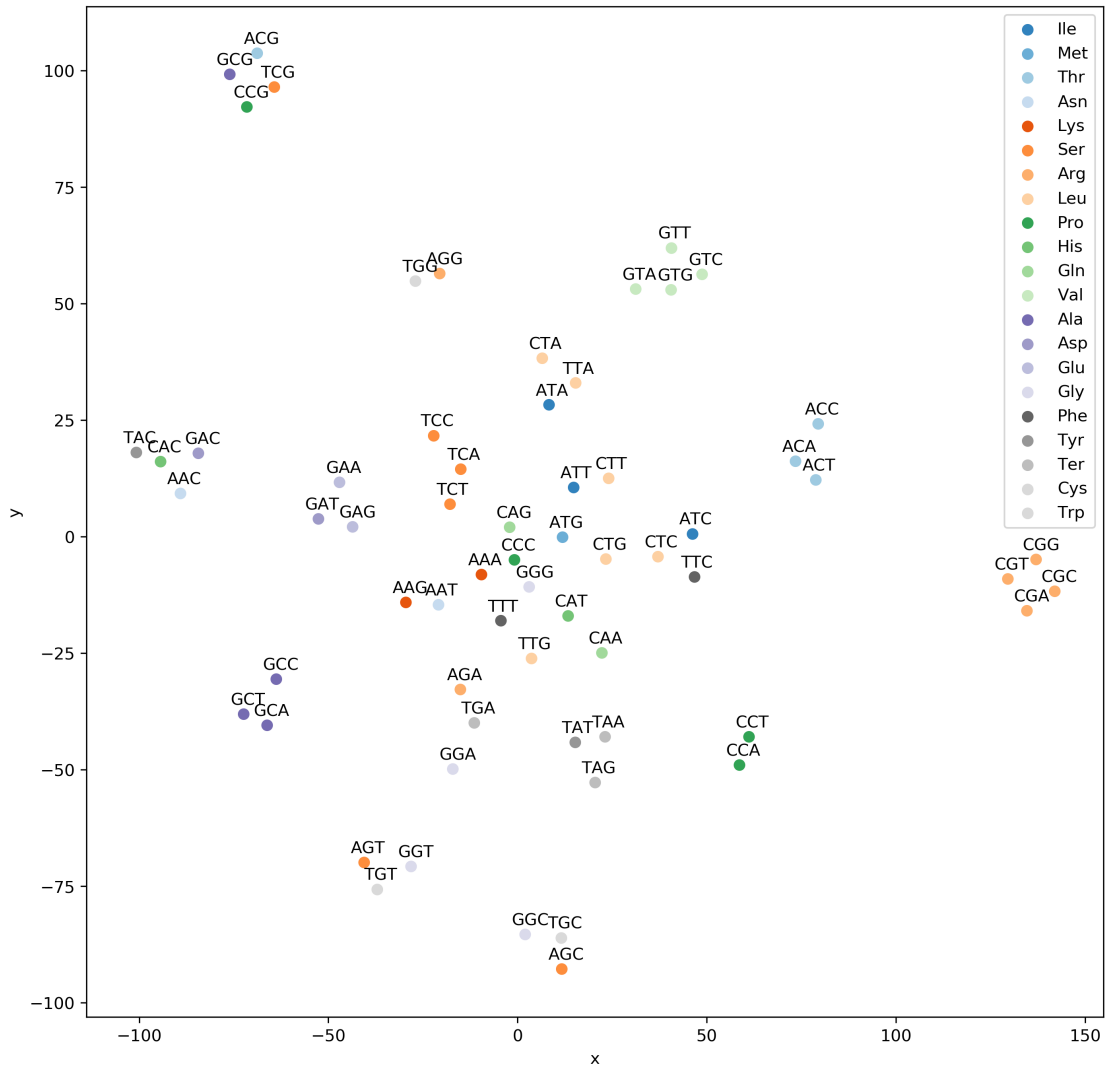

Figure 4.12: The 2-dimensional embeddings obtained by applying t-SNE to the 100-dimensional learned representations of all 64 possible 3-mers. The corresponding amino acids are color-coded.

visualizations as an indication that doc2vec was able to learn biologically useful embeddings for the overlapping 3-mers.

Putting it all together

The resulting model uses the pre-trained doc2vec network to obtain 100-dimensional embedding of the extracted nucleotide sequences around the exon start and exon end. These two embeddings, along with the length features, are then concatenated and given to a shallow MLP to obtain the final prediction. We refer to this model

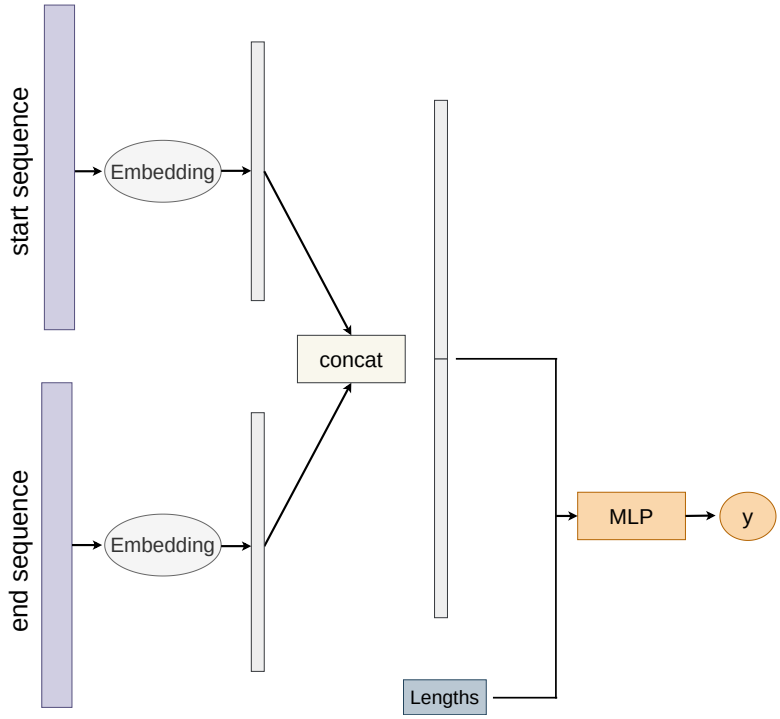

Figure 4.13: Architecture of D2V.

as doc2vec (D2V) and visualize it in Figure 4.13.

4.4.3 RASC: BiLSTM and Attention-based

Seq2seq

Sequence-to-sequence learning (seq2seq) [76] is a general deep learning-based framework for mapping one sequence to another. The first part of the framework is an encoder which processes the input symbol-by-symbol and produces a vector representation for each input symbol. The second part is a decoder which predicts the output sequence symbol-by-symbol based on the representation computed by the decoder. In the first timestep, the decoder uses the last output of the encoder and in all other timesteps, it uses its own output from the previous time step as input. The encoder and decoder are typically RNN-based networks such as LSTMs [83] or GRUs [84]. Encoder and decoder are jointly trained to maximize the probability of a correct output. This framework has been particularly successful in machine translation (MT): for instance, Google announced in 2016 that it started using seq2seq models

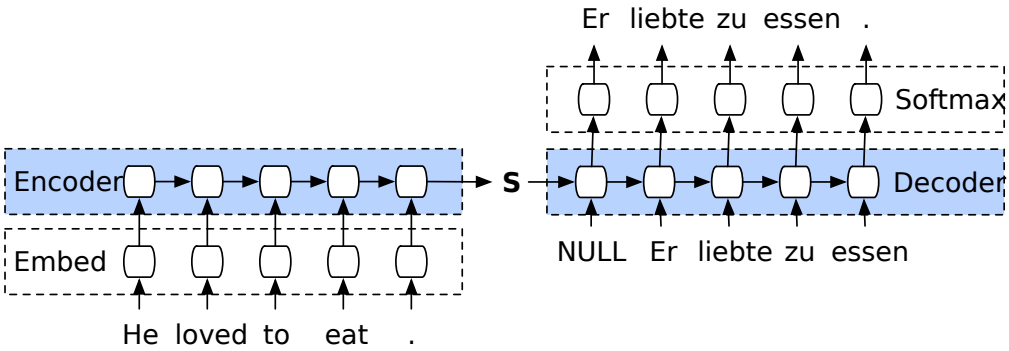

Figure 4.14: An encoder-decoder model [86] translates the English sentence 'He loved to eat' to the German sentence 'Er liebte zu essen'. The decoder takes the last hidden cell state of the encoder as initial input and generates the output token-by-token. The decoder takes its own previous output as prompt for the next output and stops generating output tokens when it generates a '.'-token.

for Google's MT [85]. Figure 4.14 visualizes an encoder-decoder model.

Are you paying attention?

In the original seq2seq framework, only the last state of the encoder is passed to the decoder. This means that the encoder needs to encode all the information observed in the input sequence into its last state. While theoretically possible, [87] hypothesized that this informational bottleneck hampers performance in practice. They proposed removing this bottleneck by introducing the attention mechanism. Attention is a mechanism whereby the decoder can access all the representation generated by the encoder at once and can 'choose' to focus on (attend to) the most important ones. Introducing attention lead to large performance gains across seq2seq-based models and quickly became standard practice [88].

The attention mechanism also adds interpretability to a model since it indicates which parts of the input sequence are most crucial for a model's prediction. Figure 4.15 visualizes what parts of an input sequence a model is paying attention to during an MT task. Attention is one of the most important innovations in deep learning research in recent years [89]. It can theoretically be useful for any task where a model needs to make a decision based on certain parts of an input. For instance, the current state-of-the-art Transformer models [77] [78] [90] in NLP forego

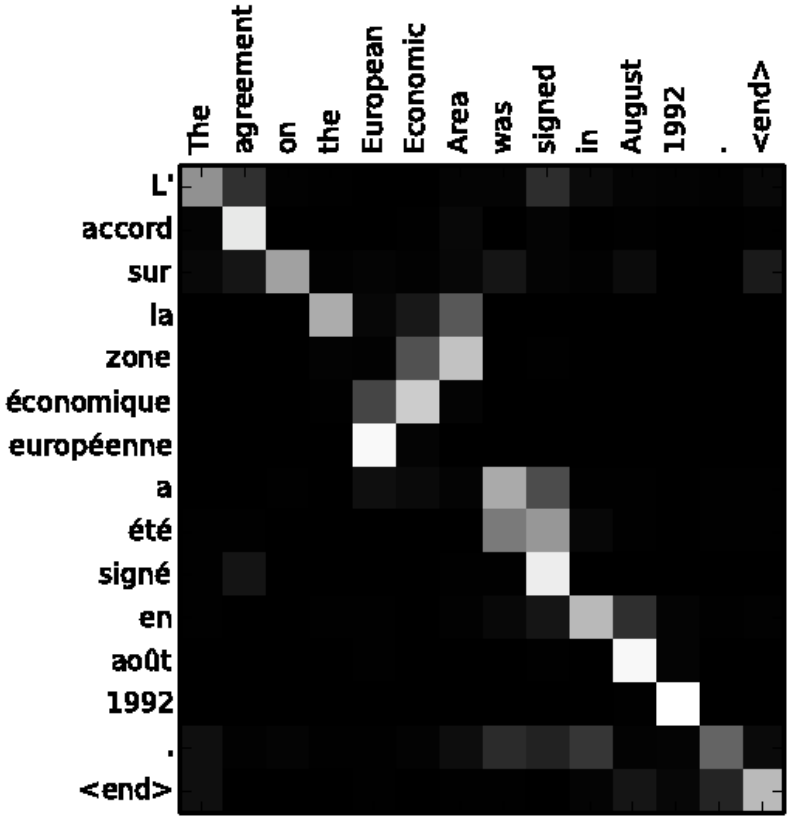

Figure 4.15: Attention heatmap when translating a sentence from French to English [87]. A brighter color indicates that the model pays more attention to a particular word. The order of the constituent words in 'zone économique européen' and 'European Economic Area' is different between French and English, but the model learned to pay attention to the semantically corresponding word during translation.

recurrent networks completely and solely use attention within the encoder and decoder components. Attention has also been successfully used in other domains such as recommender systems [91] or speech recognition [92].

Despite these successes, not many attention-based models have been applied to genomic data yet and in particular, none have been applied to the prediction of splicing behaviour. Aiming to close this gap, we develop an attention-based model for the prediction of splicing behaviour and now describe our deliberations when doing so.

Why a LSTM-based architecure was chosen over Transformers

Generally speaking, the standard architectures using attention are either seq2seq models which incorporate LSTMs (or other RNNs) or Transformers. However, in recent years, LSTM-based models have largely been replaced by Transformers [86]. Thus, it is perhaps surprising, that we chose to not base our model on the Transformer network architecture:

In contrast to LSTMs who need to process all symbols in their input sequentially, Transformers can process them concurrently. This is the main advantage of Transformers over LSTMs and enables them to scale to enormous datasets and hundreds of millions (or billions [90]) of parameters [77].

However, we have no huge datasets to scale to: our largest exon dataset consists of roughly 50,000 training samples and this already includes all the known cassette exon in the human genome. Creative use of data augmentation might increase dataset size by two or three factors, but even 150,000 training samples are still far fewer than the millions of training samples used in NLP. Scaling down Transformers models would mean they lose their main advantage over LSTMs: on small NLP datasets LSTMs and small (not-distilled [93]) Transformers perform similarly [94][95] and there is no clear consensus which approach is superior.

Additionally, the autoregressive bias inherent to LSTMs aligns well with our knowledge about the biological reality: the spliceosome also processes information in the sequences piecewise. Opposed to LSTMs and the spliceosome, Transformers process their complete input concurrently.

Thus, we base the encoder part of our model on the simpler LSTMs.

Modifying seq2seq

We now describe the modifications we made to the seq2seq and attention set-up known from MT for alternative splicing classification and regression. In

particular, we describe the attention mechanism in more detail as necessary for our second modification.

Modification 1: MLP instead of a recurrent decoder

As the name implies, models based on the seq2seq framework take a sequence as input and give a sequence as output. This is achieved via using a recurrent decoder which outputs symbols until it outputs an <EOS> (end-of-sequence, e.g. a '.') token. However, we only expect a single output: a scalar signifying the classification or frequency of the alternative splicing behaviour of an exon or junction. Thus, we do not require a recurrent decoder and use a shallow MLP in its place.

Modification 2: Attention with a learned query vector

The goal of using the attention layer in our network is to attend to the most important nucleotide representations from the start and end sequences. We achieve this goal through starting from the most common form of attention, multiplicative or dot-product self-attention [77], and then modifying it for our task.

Dot-product self-attention makes uses of conceptual queries and key-value pairs. A query, key and value matrix is assigned to each symbol in the input sequence. A given query is compared to all keys by computing a similarity score between the query and each key via the dot-product. The similarity scores are normalized through a softmax layer. The normalized similarity scores are also commonly called the attention weights; keys similar to the query are weighted more. The output of the attention layer for a given query and input sequence is then the weighted vector sum obtained by multiplying each value by the attention weight of its respective key. Putting the above intuition into formulas, the (dot-product self-) attention mechanism can be described more formally:

Initially, the query, key and value matrices Q, K and V are computed by multiplying the input I with the query, key and value parameter matrices W^Q, W^K and W^V:

$$Q, K, V = IW^Q, IW^K, IW^V$$

where $I \in \mathbb{R}^{l \times in}$, and $W^Q, W^K, W^V \in \mathbb{R}^{in \times out}$, and $Q, K, V \in \mathbb{R}^{l \times out}$. *in* corresponds to the number of features of each element in the input sequence to the attention layer, *out* corresponds to the number of features in the output of the attention layer. l corresponds to the number of elements in the input sequence. The input I is a concatenation (along the first dimension) of the representations given by the encoder for the nucleotide window around the exon start and exon end.

Attention can be then computed as:

$$Z = attention(I) = softmax(QK^T)V$$

where $Z \in \mathbb{R}^{l \times out}$. Note that this makes use of the fact that $Q \cdot K = QK^T$ where $\cdot$ denotes the dot product.

As shown, multiplicative self-attention computes a separate query for each input token. It is called self-attention because it allows a sequence to pay attention to 'itself'. (Unmasked) self-attention is mainly used in the encoder part of Transformer architectures to improve the representation of each word.
However, our objective in using attention is not to improve the representation of each symbol in a sequence (finding a good representation is the task of the encoder), but to select the most important features from an input sequence. The conceptual query for our task is also always the same independent of input: is this exon constitutively spliced or not? Thus, we adapt the attention mechanism so that we learn a single, input-independent query $Q^* \in \mathbb{R}^{1 \times out}$. The output of the attention layer is now computed as:

$$Z^* = attention^*(I) = softmax(Q^*K^T)V$$

where $Z^* \in \mathbb{R}^{1 \times out}$. $attention^*$ denotes the so modified attention mechanism. Note that the query matrix Q^* is directly learned and not multiplied with the input, giving rise to its input independence. The output Z is now a weighted sum of the nucleotide representation in the input sequences which the model deemed most crucial to pay attention to.

4.4.4 Further modifications of the attention mechanism

In the following, we introduce three further modifications of the attention mechanism which we evaluated separately from the modifications described so far.

No query matrix

As the query matrix Q^* is the same independent of the exact input sequences, it could be possible to remove the query matrix and compute the attention weights sorely based on the key matrix K. This is possible by learning a key weight matrix $W^{K_{no\_query}} \in \mathbb{R}^{in \times 1}$ and computing attention as:

$$K_{no\_query} = IW^{K_{no\_query}}$$

$$Z_{no\_query} = attention^{**}(I) = softmax(K^*)V$$

where $K_{no\_query} \in \mathbb{R}^{l \times 1}$ and $Z_{no\_query} \in \mathbb{R}^{1 \times out}$. This has the advantage of reducing the number of trainable weights as the adapted key weight matrix $W^{K_{no\_query}} \in \mathbb{R}^{in \times 1}$ is smaller than the initial key weight matrix $W^K \in \mathbb{R}^{in \times l}$ and the query matrix $Q^* \in \mathbb{R}^{1 \times out}$ is removed.

Multiple attention heads

Having one set of query, key and value matrices limits the model to one representational subspace (one way of 'interpreting' the input). It might be useful for the model to have multiple representational subspaces. This is usually achieved [77] via introducing multiple attention heads: instead of learning one set query, key and weight matrices, h sets of query, key and weight matrices $W^{Q_1^*}, W^{K_1}, W^{V_1}, \ldots W^{Q_h^*}, W^{K_h}, W^{V_h}$ are learned to produce h outputs $Z_1, \ldots, Z_h$. Applying this idea to the modified $attention^*$ mechanism engenders the following formulas:

$$K_i, V_i = IW_i^K, IW_i^V$$

$$Z_i^* = attention_i^*(I) = softmax(Q_i^* K_i^T)V_i$$

where $i \in \{1, \ldots, h\}$. Note that the query weight matrices $W^{Q_1^*}, \ldots, W^{Q_h^*}$ are again directly learned, like in the single head case.

The final output matrix $Z_{final} \in \mathbb{R}^{l \times out}$ is then computed as:

$$Z_{final} = concat(Z_1^*, \ldots, Z_h^*)W^O$$

where the concatenation occurs along the second dimension and $W^O \in \mathbb{R}^{out*h \times out_{new}}$. out_{new} represents the new output dimension of the attention layer, which integrates the outputs of the h attention heads.

While these extra representational subspaces might make the model more powerful and also more robust (in the sense that multiple attention heads act like an ensemble model [96]), it also increases the number of trainable weights potentially leading to overfitting.

Convolution in attention heads

The information contained in a single nucleotide is very low. This motivated [97] to integrate an additional 1D convolution into the computation of the query, key and value matrices Q, K and V which is applied after the multiplication with the weight matrices W^Q, W^K and W^V. This convolution allows the model to more effectively integrate information from multiple neighbouring nucleotides. This extension can also be understood as an alternative to splitting the input nucleotide sequence into k-mers (similar to the split into overlapping 3-mers for the input of the D2V model described in Section 4.4.2) and provided better results than splitting the input nucleotide sequence into k-mers when using a Transformer architecture for a genome annotation task in [97]. In formulas, we can describe the adapted attention mechanism as:

$$K, V = IW^K, IW^V$$

$$K_{conv}, V_{conv} = conv(K), conv(V)$$

$$Z_{conv}^* = attention_{conv}^*(I) = softmax(Q^* K_{conv}^T)V_{conv}$$

where K and V are both fed through the same convolutional layer *conv* to arrive at K_{conv} and V_{conv}. The weight matrix corresponding to the convolutional layer *conv* is chosen so it has the same number of feature dimensions as K and V, that is, $W_{conv} \in \mathbb{R}^{out \times kernel \times out}$. $kernel$ denotes the size of the convolutional kernel. The padding is chosen so that the first dimension of K and V stays the same which leads to $K_{conv}, V_{conv} \in \mathbb{R}^{l \times out}$. We now describe two further implementational details:

To incorporate the domain knowledge that the input sequences to our model come from two different places in the genome, the convolution is applied in such a way that it does not mix the information between the sequences around the exon start and around the exon end. To achieve this, we implemented the following computations:

$$K^{start}, K^{end} = I^{start}W^K, I^{end}W^K$$

$$K^{start}_{conv}, K^{end}_{conv} = conv(K^{start}), conv(K^{end})$$

$$K_{conv} = concat(K^{start}_{conv}, K^{end}_{conv})$$

where the *start* and *end* superscript respectively refer to the sequence around the start and end of the exon. The concatenation is along the first dimension (so the length of the sequences). The analogue computation was implemented for V_{conv}.

The query, key and value parameter matrices are implemented via a fully-connected layer with the appropriate dimensions. By default, fully-connected layers also learn bias weights which are added to the output. While in theory the bias weights should be turned off to implement exactly the shown formulas, in practice other attention mechanism implementations leave these turned on [98] and so we follow suit. Thus, to be precise, the query, key and value matrix computation should be:

$$K, V = IW^K + B^K, IW^V + B^V$$

$$Z^* = attention^*(I) = softmax((Q^*K^T) + B^{Q^*})V$$

where the query, key and value bias matrices B^{Q^*}, B^K and $B^V \in \mathbb{R}^{1 \times out}$.

Putting it all together:

Like in a seq2seq model, an RNN-based Encoder is used to capture a representation of the input sequence. We attend to the most important representations of the encoded input using the modified attention mechanism we introduced. The resulting feature vector is concatenated with the lengths and given to a shallow MLP.

The input to the encoder blocks is a dense 4-dimensional embedding of the one-hot encoded sequences. This dense embedding is the same for each sequence and jointly learned during training. The extra embedding layer is included as previous work [99] has shown that otherwise, the model might suffer from limited generalization performance, caused by the sparsity of the one-hot encoded representations.

LSTM units opposed to unmodified RNNs are used as encoder as LSTM units have been shown to be better at retaining information over long sequences and avoiding the vanishing and exploding gradient problems [83]. In particular, we use Bidirectional LSTM (BiLSTM) units [100] to model that the splicing of an exon likely depends on bi-directionally interacting factors. A separate BiLSTM unit is used for encoding the exon start and exon end sequences.

As this lead to empirically better results, we also introduce a 1D batch normalization layer [101] before the attention layer. A dropout layer in the attention head is used for the same reason.

We pre-empt the results chapter by noting that we only use the extension of multiple attention heads of the three additional modifications of the attention mechanisms we considered. The architecture of our model is visualized in Figure 4.16. We refer to this new model as Recursive Attentive Splicing Code (RASC). In the experiments section, we validate the choice of introducing the attention mechanism to RASC, by removing it and evaluating the model which only uses the hidden state of the BiLSTM units. We denote this model as Recursive Splicing Code (RSC).

For RASC, the hyperparameter space was a lot larger than that of previous models and model performance was very sensitive to the hyperparameter choice. The

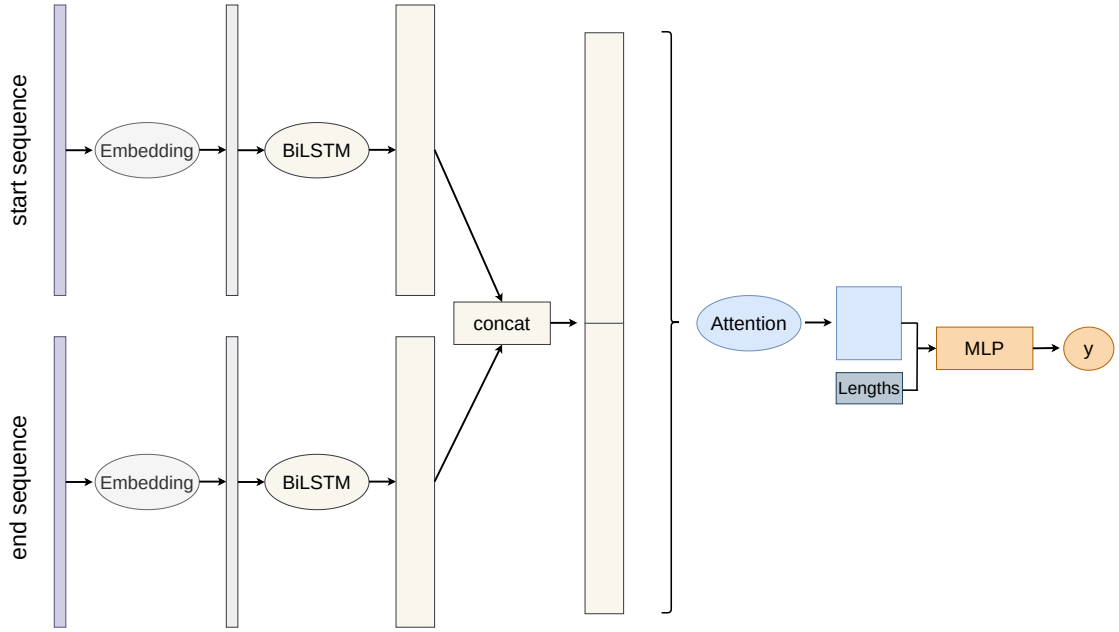

Figure 4.16: Architecture of RASC.

most crucial hyperparameters were the number of encoder dimensions and the number of dimensions of the attention layer. Hyperparameters were optimized via a grid search on the HipSci MAJIQ dataset.

4.5 Implementation, training and evaluation

4.5.1 Implementation

All models were implemented using the PyTorch library (version 1.5.0) [102]. The doc2vec algorithm was implemented using the gensim library (version 3.8.3) [103]. PyTorch and gensim are both standard choices when implementing deep learning or text embedding algorithms. The data processing was done using a mixture of Python and Bash scripts as well as the software tools MAJIQ and SUPPA described in Sections 4.3.1 and 4.3.1. The latest versions of SUPPA (version 2.3) and MAJIQ (version 2.0) available as of July 2020 were used. The complexity induced by the use of a large number of different datasets and data processing methods was handled by standardizing the shape of the final training samples:

even though 10 different datasets and 3 different models are used, the codebase contains only 2 different data loader and trainer classes.

The structure of the repository was based on the PyTorch Deep Learning Project template available at `github.com/victoresque/pytorch-template`. This template provides abstract classes for data loaders, trainers and the models themselves. These abstract classes already contain a lot of the implementation-independent code to log results, and train and save models. The abstract base classes were then extended with implementation-specific code, such as, exact data formats. Using this structure avoids the duplication of a lot of boilerplate code.
To run an experiment, the experimental settings are defined in a separate JSON file. This allows for easy and unambiguous reproduction of results. All experiments for whom results in this study are shown are available as JSON files that define the exact parameters used. The code repository itself contains multiple README which give more details about the code structure, how to run experiments and what naming conventions are used.

4.5.2 Training

The models were trained on one NVIDIA GeForce GTX 1080 Ti GPU. This proved to be sufficient as initial exploratory experiments showed no advantage of using deep networks and the networks we evaluate are all relatively shallow. For training and testing, we randomly split each dataset into 10 folds: in a given run, 8 folds were used for training, 1 fold for validation and 1 fold for testing. Each model was trained with different folds 9 times and we compute the mean test set performance. The models were trained with early stopping on the validation fold until they stopped improving for 15 epochs. This typically occurred after 70 - 150 training epochs.

DSC, D2V and RASC respectively contain 20,001, 6,801, and 66,861 trainable weights. In absolute terms, all of these models are still small compared to models known from Computer Vision and Natural Language Processing [10]. The number of trainable weights is also reflected in the training times with the models respectively

DSC	
Kernel filters	32, 8, 8
Kernel size	7, 5, 3
Dropout (after fully-connected layer)	0.5
Neurons fully-connected layer	64
Dropout (after convolution)	0.2, 0.2, 0.2
D2V	
Embedding dimension	100
Neurons (fully-connected layers)	32, 8
Dropout (fully-connected layers)	0.2, 0.2
RASC	
Dense embedding dimensions	4
BiLSTM dimension	50
BiLSTM layers	1
Attention heads	4
Attention head dimension	50
Dropout (attention head)	0.4
Attention layer output dimension	100
Neurons fully-connected layer	128
Dropout (after fully-connected layer)	0.5

Table 4.2: Hyperparameters of the three models.

taking between 5 to 30 minutes, 1 to 5 minutes and 15 to 75 minutes to train once (depending on dataset). The comparatively long training time of RASC is likely also influenced by LSTMs processing the input sequentially. The pre-training of the doc2vec model on the human genome took 3 hours.

The hyperparameters used for training the models are given in Table 4.2. More details on the doc2vec pre-training procedure and hyperparameters values are given in the appendix in Section 7.2.

Loss

All models were trained to minimize the binary cross-entropy loss:

$$\mathbb{L} = -(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y}))$$

where the ground truth $y \in \{0, 1\}$ and the prediction of the model $\hat{y} \in [0, 1]$. The binary cross-entropy loss is the standard loss for training deep learning-based (binary) classifiers.

4.5.3 Evaluation

To evaluate the models, we report the mean test set performance and standard deviation across (cross-validation) runs. Where we established in a base experiment that the variance for a given model was low between runs (lower than ∼1%), we only trained the model once in follow-up experiments for practical reasons.

Previous studies have shown that model performance varies significantly between lowly and highly included alternatively spliced exons [40] [39]. Thus, following previous work, we split our test set into a set of highly (PSI > 80%) and a set of lowly to moderately included, alternatively spliced exons (PSI <=80%). Both sets contain all constitutive exons. We denote the first set as high PSI dataset (HighPSI), the second as low PSI dataset (LowPSI) and the complete test as all PSI dataset (AllPSI). We evaluate the models on all three datasets.

Metrics

As in previous work [39] [40], we use the Area under the receiver operating characteristic curve (AUC) as the primary evaluation metric. The AUC provides an aggregate measure of the classification performance across all possible decision thresholds.

However, the AUC does not provide information about model performance at a particular decision threshold. To evaluate whether a model is fit for practical use, decision threshold-specific metrics like sensitivity and specificity are crucial to know. To evaluate these, we analytically compute the decision threshold which maximizes the F1 score. The F1 score provides an aggregate measure which treats precision and recall equally and is defined as the harmonic mean of precision and recall. It is a common measure to evaluate the performance of classifiers and we use it as a starting point for our further analysis.

Again following previous work [31] [32], we use the explained variance R^2 to evaluate model performance on the regression task.

5 Results

This section presents and discusses the results of our experiments based on the three models and the four primary datasets introduced in Chapter 4. It also describes the rationale behind deciding which experiments should be run.

Results are given as obtained on the test set. Across all datasets and experiments, validation and test set metrics differ by less than 1% on average. We observe no undue overfitting and unless explicitly mentioned, AUCs on the training set are generally higher by 0.02 to 0.06.

5.1 HEXEvent dataset

To obtain a baseline, we evaluate all three models on the HEXEvent dataset as used in [40] and introduced in 4.2.1. The results of these experiments are shown in Figure 5.1.

Analysis of main findings

Generally all models perform extremely well with AUCs nearing 90%. While differences are small, RASC seems to perform slightly better than the other models.

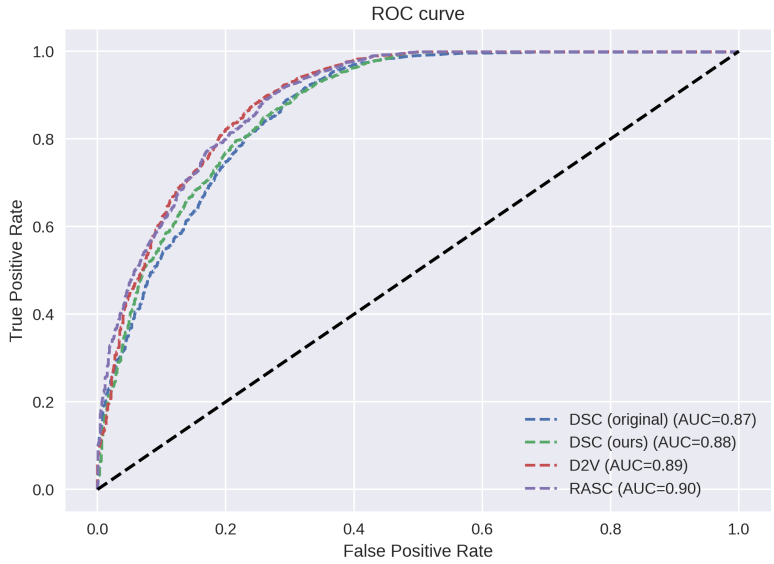

Figure 5.1: Comparison of the ROC curves of the three models as well as the original implementation of DSC on the HEXEvent dataset. The values of the original implementation of DSC were obtained by rerunning the training on the publicly available implementation.

Assessing our reimplementation, we observe a small difference between the AUC value reported in [40] (mean 0.899, standard deviation of 0.18) and the ones we observe (mean 0.873, standard deviation of 0.06). This is likely the result of random statistical noise influenced by different random seeds between runs and differences between TensorFlow/Keras (their implementation) and PyTorch (our implementation). Notably, when reevaluating the publically available original implementation in a single run, we also obtain results (0.872 AUC) closer to the results of our reimplementation. Thus, we conclude that, albeit with minor caveats, the reimplementation and replication of [40] was successful.

Length features are necessary

Reimplementation was completed piece-wise, that is, first the sequence features were added as input, then the networks were trained to ensure that this was done correctly and then the length features were added. This lead to an interesting observation: model performance is significantly worse when no length features are given to the models. Quantitative results for this observation are given in Table 5.1.

Model name	Only sequences	Sequences + lengths	Relative gain
DSC	0.618	0.873	68.4%
D2V	0.614	0.896	73.7%
RASC	**0.636**	**0.904**	66.3%

Table 5.1: Performance of the main models on the HEXEvent dataset with and without length features given as AUC. The relative performance gain (from adding the length features) was computed as $\frac{AUC - AUC_{no\_lengths}}{AUC - 0.5}$. Computing it this way accounts for the baseline AUC of random guessing being 0.5.

This observation is true across models and leads to an average relative performance drop of over 66%. The information about the secondary structure obtained in the lengths seems to be necessary for the models to obtain good predictive power.

Stress testing HEXEvent

To further investigate, we also test three other models: MLP100, MLP20 and MLPLinear which respectively contain 100, 20 and 20 trainable parameters. The models are MLPs with one hidden layer which only take the length features as inputs. MLPLinear does not use a non-linear activation function after its hidden layer.

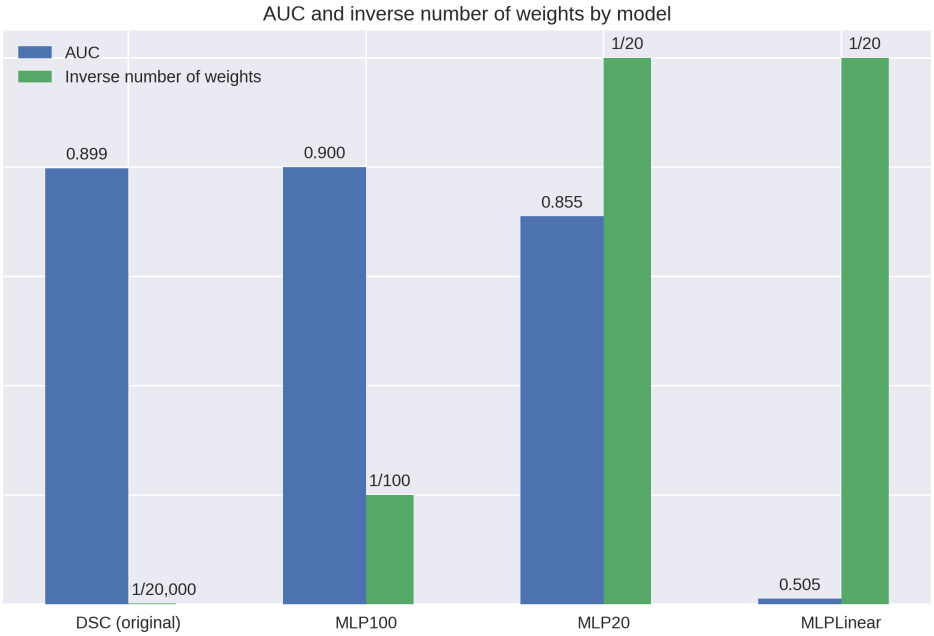

Figure 5.2: Comparing DSC against three simple MLPs on the HEXEvent dataset from [40]. The inverse number of trainable weights are shown to emphasize the different model sizes.

Surprisingly, the results in Figure 5.2 show that it is possible to replicate the results of [40] with these very simple models using two to three orders of magnitude fewer parameters (note that DSC has 20,001 parameters). Model performance is improved by adding further parameters and breaks down when no non-linearities are used in the network. This indicates that the models capture a relatively simple, but non-linear relationship between the lengths and the classification of an exon in the dataset. There are multiple possible explanations for why we observe this behaviour:

1. There are confounders in the dataset learned by the model. As discussed in Section 4.2.1, EST-based data is inherently biased. The biases inherent in EST-based data could be captured by the exon and intron structure, and the models are learning to make their prediction based on this bias.
2. Exon splicing is extremely well predictable based on the lengths of neighbouring introns and exons. This is very unlikely given that research into splicing has been ongoing for over four decades years.
3. There are bugs (e.g. mixing of testing and validation data) in our reimplementation. In the first instance, this is unlikely given that we were able to replicate the original results of [40]. To reduce complexity and further reduce the likelihood of a bug leading to these observations, we extracted the complete code for replicating the results of the simple MLP models from Figure 5.2 into a single file. Additionally, in case of a data leakage bug, the performance of the linear model likely would not break down either. Therefore, we judge bugs in our reimplementation unlikely.

Overall, we conclude that the HEXEvent-based dataset is most likely fundamentally flawed and suffers from confounders. These findings have significant implications. It calls into questions the meaningfulness of [40]'s results, showing the competitiveness of their model. These findings likely also warrant a critical investigation of any other conclusions drawn from papers based on the HEXEvent database. At the time of writing, the HEXEvent paper is cited 34 times. While most of these citations are in passing, multiple papers use a HEXEvent-derived dataset for the training of

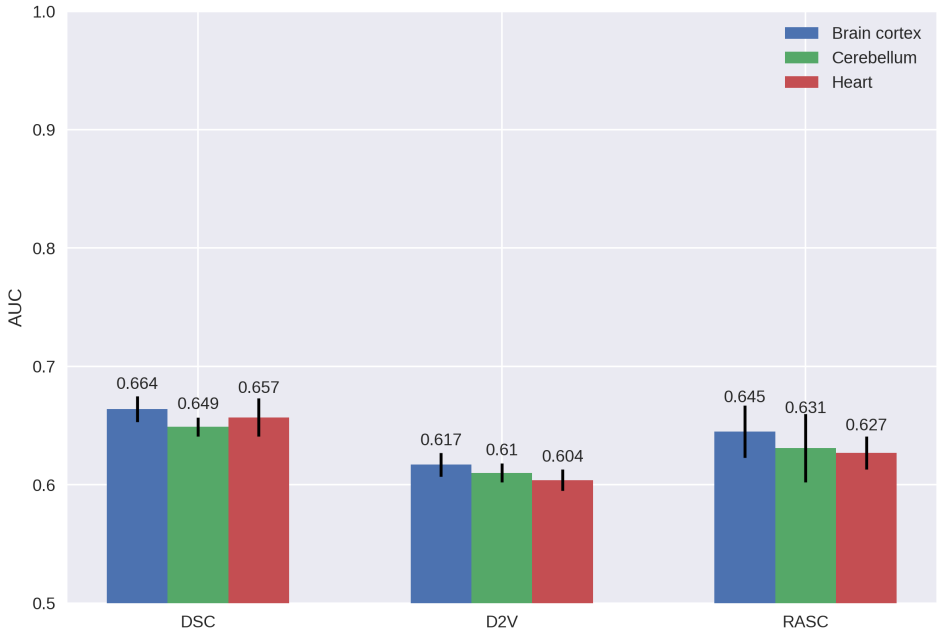

Figure 5.3: Performance on the GTEx-based exon-centric datasets across different tissues. The error bars give the standard deviation across all cross-validation runs.

Machine Learning models such as SVMs [39], Random Forests [66] [104], Decision Trees [105] or AdaBoost-based algorithms [106]. The validity of these papers results' are called into question by these findings.

These data quality issues motivated us to construct alternative, better, datasets. We now give their results.

5.2 GTEx-based datasets

5.2.1 Exon-centric datasets

Main analysis

Results are given in Figure 5.3. Across all models, performance is poor: the highest mean performance we observe is 0.664 AUC. Surprisingly, performance is also very similar across tissues: the mean cross-tissue performance difference of 0.007 AUC is smaller than the mean cross-run difference of 0.014 AUC. This is surprising from a biological perspective as cross-tissue splicing differences are well-documented [107]. This could indicate that, while splicing across tissues varies, the relative

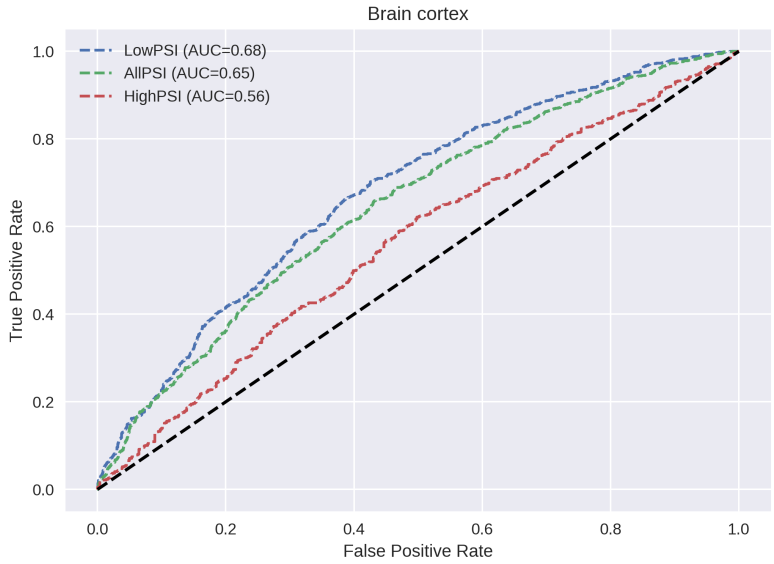

Figure 5.4: ROC curve of RASC on the GTEx-based exon-centric brain cortex dataset.

complexity of cassette exon splicing behaviour across the evaluated tissues is similar. However, this would be overinterpreting models with low predictive power. Most likely, the models are already struggling to learn the baseline splicing behaviour invariant across tissues and do not learn finer cross-tissue differences. The low cross-tissue performance differences are also surprising from a machine learning perspective, as the cerebellum-based dataset contains almost twice as many training samples as the heart-based dataset. This indicates that either 1) the models have already reached a point of diminishing returns for adding more data or 2) the models generally require significantly more training samples. All models perform best on the dataset based on a brain cortex sample.

Cross-model performance differences are small (mostly between 0.02-0.03 AUC), yet constant between tissues: DSC tends to perform best, followed by RASC and the D2V model. The variance of RASC between runs is on average twice as twice as high as the respective variance of DSC and D2V. As a result, RASC is the best performing model, as measured by the maximum rather than mean AUC value, on the brain cortex tissue-based and cerebellum tissue-based datasets. This indicates that RASC needed more regularization on this dataset and dataset specific fine-tuning could have lead to it performing the best across cross-validation runs.

Differences between highly and rarely included exons

Figure 5.4 gives more insight into model performance. The performance on highly included, alternatively spliced exons ($80\% \leq \text{PSI} < 99\%$) is significantly worse than on more rarely included, alternatively spliced exons ($\text{PSI} < 80\%$). Per definition of the AUC, the AUC on the AllPSI dataset lies in-between these two extremes. In this case, only ∼28% of alternatively spliced exons belong to HighPSI and therefore the combined AUC is heavily weighted towards the AUC on LowPSI. These observations mirror analogue observations made on the HEXEvent dataset [40].

5.2.2 Intron-centric datasets

Figure 5.5: Performance on the GTEx-based junction datasets across different tissues.

Main analysis

Figure 5.5 shows that the AUC of all models increases by at least 0.05 to 0.06 compared to the exon classification task. This is likely a result of the intron-centric

datasets containing on average four times more training samples than the exon-centric datasets. The performance of RASC is the most promising and experiences the largest growth (by 0.15 AUC) demonstrating its capacity to capitalize on the increased number of training samples. The corresponding classification accuracy of RASC, when choosing the decision threshold which maximizes the F1 score, is roughly 70% and demonstrates that there is still room for further improvement.

Similar to the exon-centric datasets from Section 5.2.1, we observe very low cross-tissue performance differences. We again deem model performance generally too poor to warrant speculation about the underlying biological realities. However, we come to a different conclusion regarding the observation that the different dataset sizes between tissues do not affect model performance: for the GTEx-based exon-centric datasets we hypothesized that the models have hit diminishing returns for adding more data or that they need significantly more data. Since the number of training samples has quadrupled in the move from exon- to intron-centric datasets and cross-tissue performance differences are still very small, we conclude that the models have hit a point of diminishing returns for adding more data.

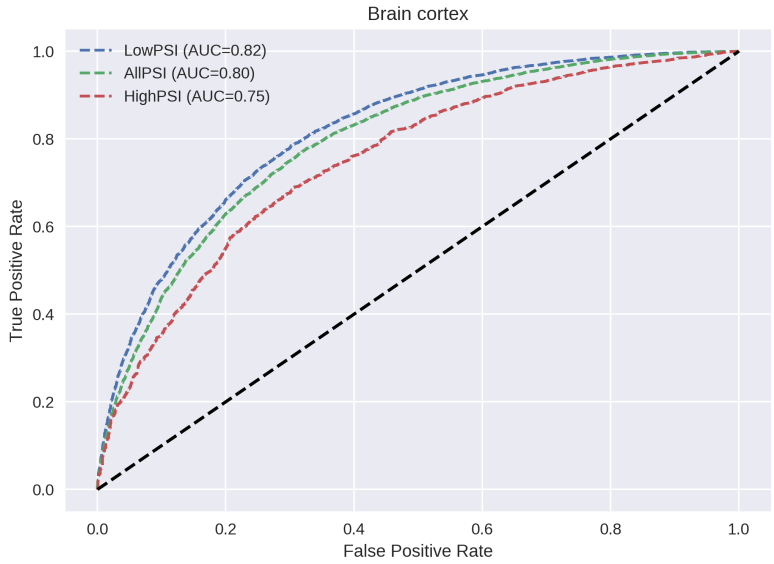

Figure 5.6: ROC curve of RASC on the GTEx-based intron-centric brain cortex dataset.

The performance across rarely and highly included junctions is less disparate than on the exon-centric datasets (see Figure 5.6). This indicates that the models are

better able to learn to recognize motifs which separate constitutive and highly alternatively spliced junctions than for exons.

Assessing model performance without length features

Model name	Only sequences	Sequences + lengths	Relative gain
DSC	0.661	0.704	0.211
D2V	0.629	0.673	0.254
RASC	**0.776**	**0.808**	0.104

Table 5.2: Performance on the GTEx-based intron-centric cerebellum dataset with and without length features.

As model performance is more promising than on the GTEx-based exon-centric datasets, we validate that this isn't due to an overreliance on the length features as on the HEXEvent dataset. Table 5.2 shows that the models do not completely rely on the length features, albeit they still make up a large part of model performance. From a biological standpoint, the length features are not a factor that explicitly affects the spliceosome, but are correlated with multiple factors such as exon and intron nucleotide composition [108] or splice site strength [68] that do. Ideally, the models would learn to identify these factors directly and would not rely on the length features. Thus we conclude that the dataset is an improvement over the HEXEvent dataset, but generating a dataset where model performance is less dependent on the length features is still desirable.

Evaluation of the GTEx-based datasets

Overall, we observe poor to promising model performance, biologically surprising low cross-tissue performance differences and still undesirably high reliance on the length features. In Section 4.3.1, we mentioned issues when trying to estimate PSI naively. Although we partially alleviated these in pre-processing, we do not account for the information contained in non-junction reads and do not integrate information from multiple samples. As a result, data quality is likely still an issue distorting the results in this section. Thus we conclude that the GTEx-based datasets improve upon

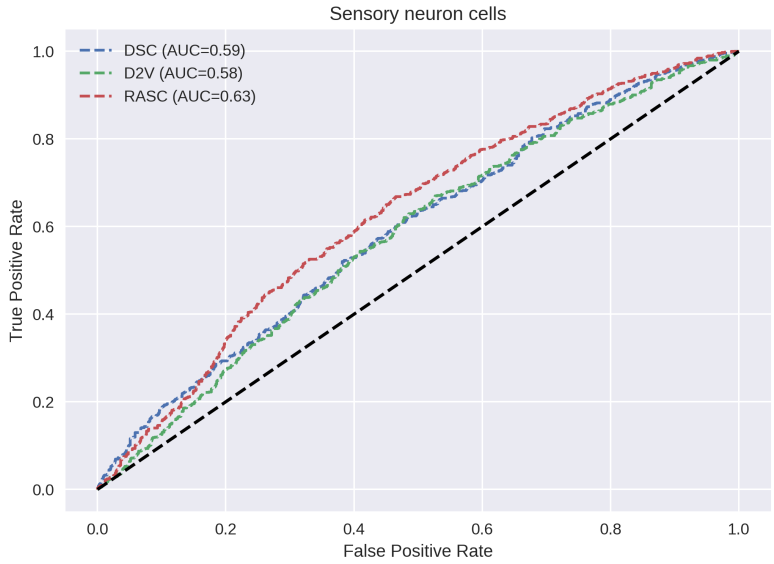

Figure 5.7: Comparison of the ROC curves of the three models on the HipSci SUPPA dataset. The dataset is based on iPSC lines differentiated to sensory neuron cell lines.

the HEXEvent dataset, but using datasets based on more accurate PSI estimation methods is desirable and would lead to more meaningful results.

5.3 HipSci SUPPA datasets

5.3.1 Dataset based on iPSC-derived sensory neuron cells

Main analysis

Figure 5.7 shows how the model perform very poorly on the HipSci SUPPA dataset. The performance is worse than on the GTEx-based exon-centric dataset and roughly equal to the performance on the HEXEvent dataset without length features. This indicates that

1. either we arrive at a strong dataset here and that constitutive exon classification is more challenging than so far anticipated,
2. or this dataset suffers from data quality and quantity issues which make constitutive exon classification very challenging for the models.

Among these two, 2. is more likely because SUPPA does not address various challenges in PSI estimation and because it does not generate constitutive training

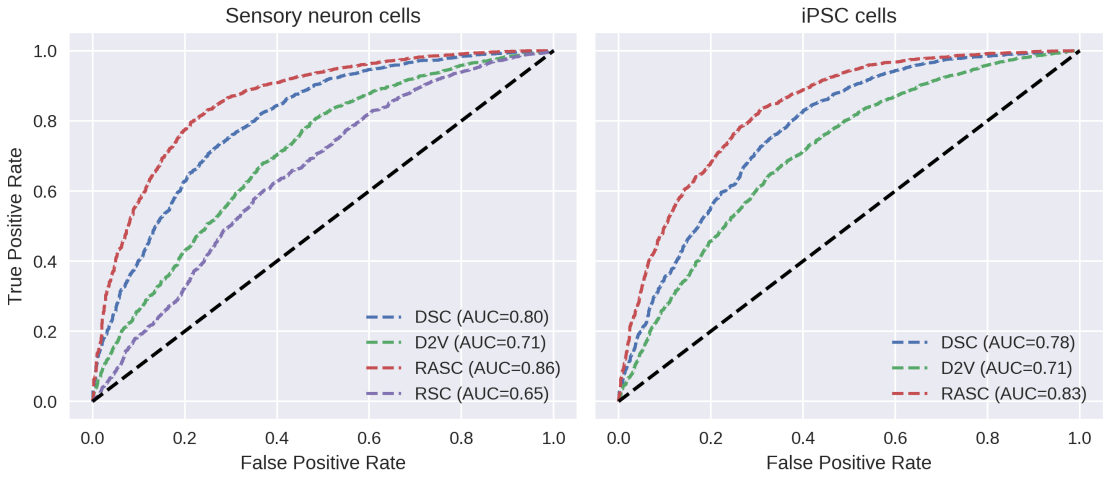

Figure 5.8: Left: ROC curves on HipSci MAJIQ dataset derived from iPSCs differentiated to neurons. Right: ROC curves on HipSci MAJIQ dataset derived from undifferentiated iPSCs.

samples (see also Section 4.3.1). SUPPA does not appear to be suited for our purposes. Therefore, we skip the evaluation of the models on the HipSci SUPPA dataset based on undifferentiated iPSC lines.

5.4 HipSci MAJIQ datasets

5.4.1 Dataset based on sensory neuron cells and iPSCs

Main analysis

All models perform significantly better than on previous datasets and we observe stark disparities between the models (see Figure 5.8). In particular, RASC outperforms D2V and DSC by respectively $\sim 44\%$ and $\sim 15\%$. Adding attention to RSC promotes it from the worst to the best performing model, validating our choice. Performance on the sensory neuron cell dataset is generally higher by roughly 2%, indicating that the captured splicing behaviour is easier to predict for the models. To explain this behaviour, we observe that the models are again worse at classifying highly-included than lowly-included non-constitutive exons (see Figure 5.9) and note that highly-included exons constitute a roughly 2% larger proportion in the iPSC-based dataset.

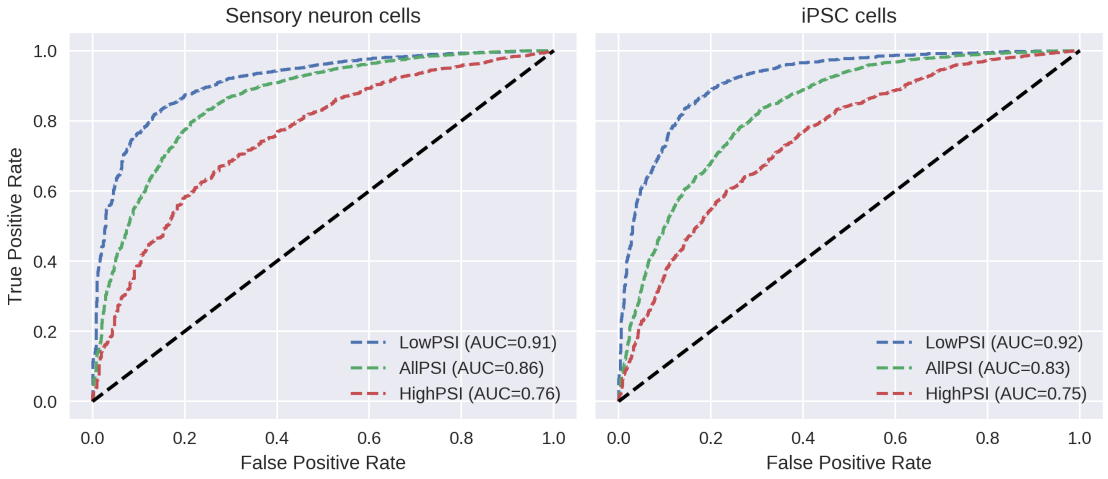

Figure 5.9: Left: ROC curves on HipSCi MAJIQ dataset derived from iPSCs differentiated to neurons. Right: ROC curves on HipSCi MAJIQ dataset derived from undifferentiated iPSCs.

Impact of architectural choices

The poor performance of D2V is likely a result of its architecture: the 100-dimensional embeddings used for classification were not optimized jointly during training and likely do not contain fine-grained information, such as the specific positons of motifs, required for accurate splicing prediction. We conjecture that D2V would benefit from receiving further split input sequences, e.g., eight input sequences each containing 35 nucleotides. The best solution, while keeping the word embeddings, would likely be to embed the overlapping 3-mers in the sequence via word2vec and combine this with an efficient convolutional architecture, as done in [32].

Assessing the dataset quality

Model name	Only sequences	Sequences + lengths	Relative gain
DSC	0.811	0.808	-1.0%
D2V	0.665	0.715	23.3%
RASC	**0.853**	**0.865**	3.3%

Table 5.3: Performance on the sensory neuron cell-based HipSci MAJIQ dataset with and without length features given.

As litmus test for dataset quality, we repeat the experiment of removing the length features as model input. We document the results in Table 5.3. They reveal that

a large part of D2V's performance relies on information contained in the length features. However, since removing the length features only marginally affects the performance of DSC and RASC, we conclude that this observation is symptomatic of D2V being unable to effectively incorporate information from the sequences (see discussion in Section 5.4.1), rather than the dataset being biased in the same way as the HEXEvent dataset.

Given a) the strong model performance independent of the length features and b) the prior expectation for MAJIQ to produce the highest quality PSI estimates, we conclude that it does. This indicates that the comparatively poor performance on the GTEx-based and HipSci SUPPA datasets are due to poor data quality, and not due to the inherent difficulty of the splicing classification task. Thus, using the HipSci MAJIQ datasets is preferable to using any of the other datasets we have evaluated.

Motivated by this observation, we use the HipSci MAJIQ dataset to validate our architectural choices for RASC and to carry out further analysis omitted on the other datasets.

5.4.2 Evaluation of the additional attention modifications

We evaluate the three additional attention modifications described in Section 4.4.4 by testing them sequentially. The results of these tests are illustrated in Figure 5.10.

Using multiple attention heads

Using multiple attention heads leads to significantly improved performance and to significantly reduced variance between runs. Thus, the additional representational subspaces appear to improve performance and the ensemble model effect of multiple attention heads appears to reduce variance. No overfitting is observed, despite the number of model weights increasing from roughly 36,000 to 67,000 compared to the single-head attention model. Due to the uniformly improved performance, we elect to incorporate multiple attention heads into RASC.

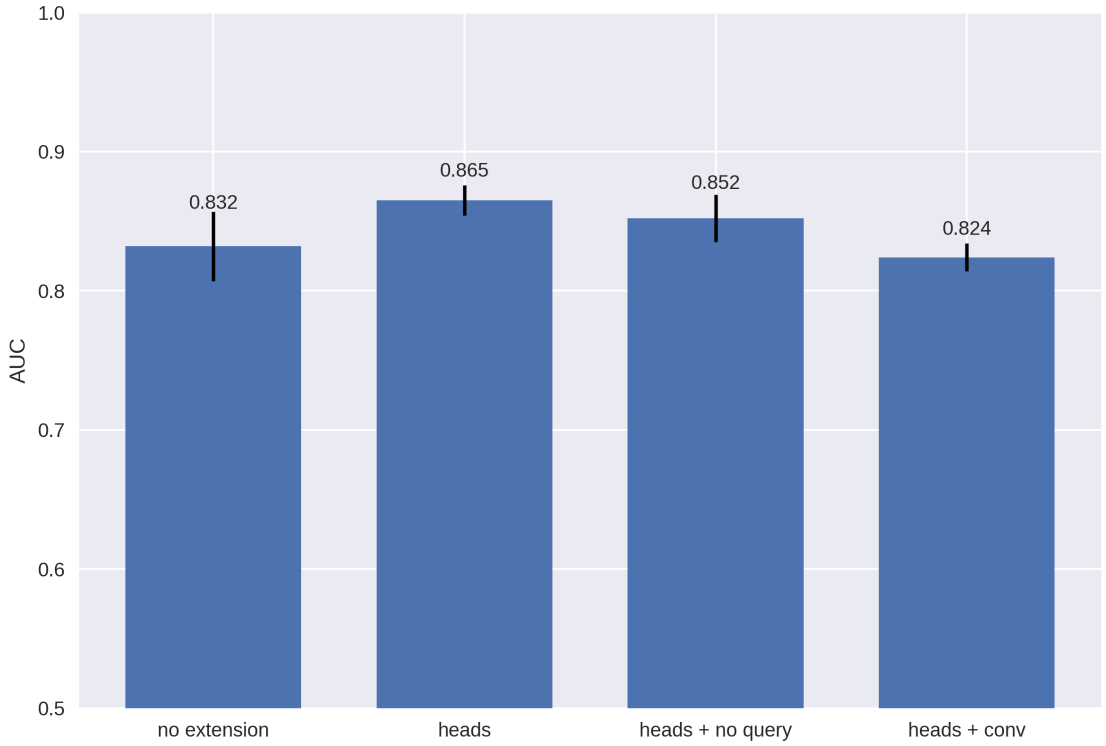

Figure 5.10: Performance evaluation of the three extensions to the attention mechanism. We denote the use of multiple attention heads as 'heads', the removal of the query matrix as 'no query' and the introduction of an additional convolution as 'conv'.

Removing the query matrix

We observe slightly diminished performance upon removing the query matrix. The extra representational capacity afforded to RASC by having query matrices appears to improve its predictive power. Despite decreasing model size by roughly 10,000 parameters when removing the query matrices, we decide to not incorporate this adaption of the attention mechanism.

Adding an extra convolution

Adding an additional convolution operation to the attention mechanism, surprisingly leads to decreased performance, even comparing unfavorably to the baseline single-head attention model. This in contrast to [97] where this extension lead to very significant relative performance improvements. There are likely multiple factors contributing to these varying results:

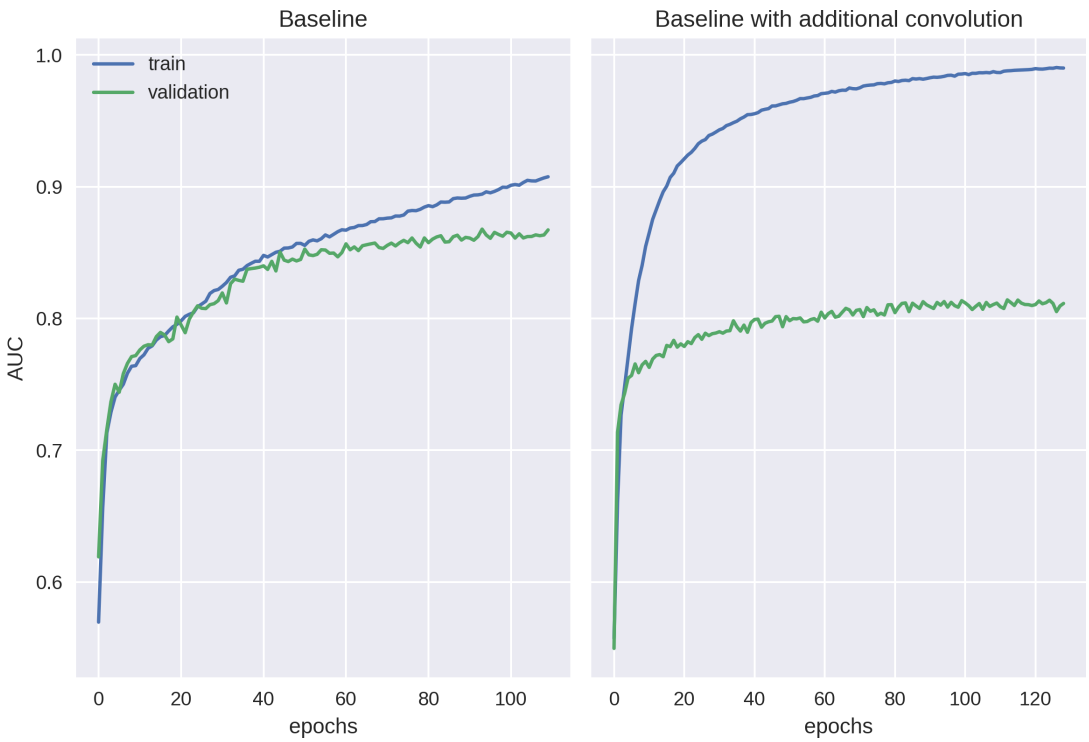

Figure 5.11: RASC overfits when the additional convolution operation is introduced into the attention heads. The baseline refers to RASC with multiple attention heads.

1. Although we process the input at single-nucleotide resolution, the use of a BiLSTM unit means that the encoder is aware of the nucleotides which precede and follow a given nucleotide. Therefore, the model can already encode information about the neighbouring nucleotides in the representation for a given nucleotide. This is in contrast to a Transformer model, as used in [97], where the layers are non-recurrent and only work at single-nucleotide resolution. Thus, the convolutional layers do not give RASC modelling capacity it does not already theoretically has.

2. In [97], a 3000 nucleotide context window is processed through 6 attention layers (so 500 nucleotides per attention layer). In our work, a 280 nucleotide context window is processed through 1 attention layer. Thus, one attention layer in RASC needs to integrate information from fewer nucleotides and RASC might need less additional capacity to integrate evidence from multiple nucleotides than [97]'s model.

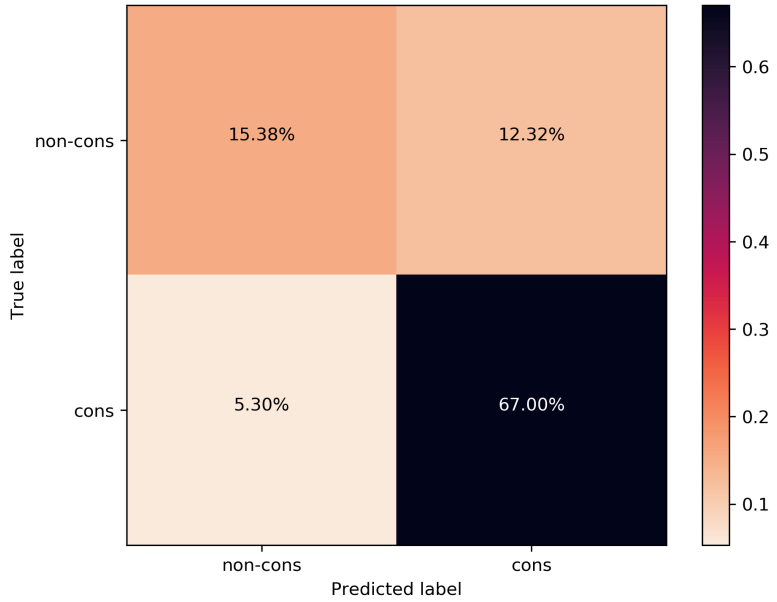

Figure 5.12: Confusion matrix from RASC on the neuron HipSci MAJIQ dataset. 'cons' and 'non-cons' respectively denote constitutive and non-constitutive exons.

3. Furthermore, we observe overfitting when using this extension. The training curve in Figure 5.11 showcases this behaviour. We observe overfitting despite using the same convolution across multiple heads, increasing the dropout probability in the attention heads, introducing additional batch normalization layers and limiting the convolution kernel size to 3.

In summary, we only choose to extend RASC with multiple attention heads. The results of RASC we report on the other datasets, also use the variant of RASC with multiple attention heads.

5.4.3 Sensitivity and specificity analysis

We present the confusion matrix for the binary classifier based on RASC in Figure 5.12. The decision threshold which maximizes the F1 score was chosen, resulting in a F1 score of 0.88 (for the negative class of constitutive exons) and classification accuracy of 82%. We observe comparatively few false positives, but many false negatives and correspondingly obtain a low sensitivity score of 0.56, but a high specificity score of 0.93. Thus, the classifier performs well at ruling in that an exon

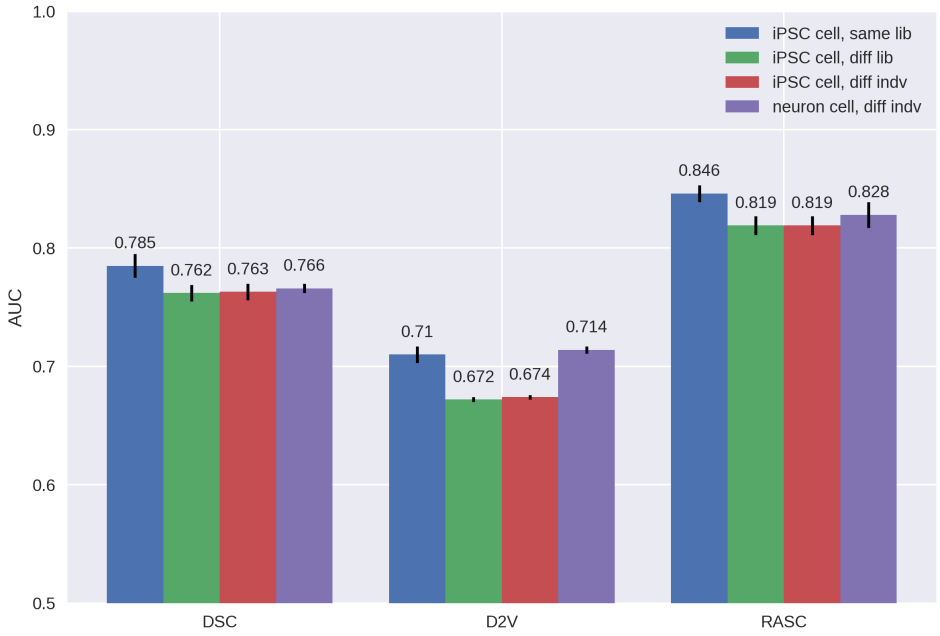

Figure 5.13: Performance when training models on one HipSci dataset and testing on the same dataset as well as three others. Within the bars of one model, going further right means testing on a dataset which is less similar to the dataset the model was trained on (the left-most bar display performance when testing the model on the dataset it was trained on). 'lib' denotes the library a sample is based on, 'indv' denotes the individual the sample was taken from, 'diff' denotes different. 'iPSC' refers to a dataset based on RNA-seq data from undifferentiated iPSCs while 'neuron cell' refers to iPSCs differentiated to sensory neuron cells.

is alternatively spliced, but poorly at ruling that it isn't. This means the classifier is attractive for guiding the investigation of exons, where it wasn't previously known that they are alternatively spliced. This would not be possible with a less specific classifier because further investigation relies on expensive wet-labs experiments for which the number of false positives need to be minimized. When employing RASC in practice, the decision threshold should be chosen such that the specificity is even higher (at the cost of further reduced sensitivity): a false positive rate of 5.3% is still too high for wet lab experiments. Thus, we conclude that our classifier shows attractive properties, but would require further tuning before practical use.

5.4.4 Cross-condition performance

We evaluate how well the models generalize when trained on a dataset derived from one individual and then tested on a dataset derived[1]

- from the same individual, but a different library.
- from a different individual, but the same cell type.
- from a different individual from a different cell type.

We visualize the results in Figure 5.13. Unsurprisingly, performance generally drops when a model is evaluated on a dataset different from the one it was trained on. Surprisingly, this performance drop does not directly correlate with the similarity to the original dataset: the least similar dataset from a different tissue performs best. This is because the baseline performance on the sensory neuron dataset is roughly 2% higher and it experiences only a roughly 1% higher performance drop when training was done on a different dataset. Thus, the models generalize extremely well across tissues and conditions. As previously suggested, this is biologically surprising due to known splicing differences between tissues [107]. A possible explanation is that the datasets do not capture the splicing behaviour which is different across tissues.

However, to more seriously investigate how splicing differs between conditions and tissues, the capability of MAJIQ to quantify differential splicing should be taken advantage of. The best performing model, RASC, could be adapted such that it takes in an exon as well as two different conditions (e.g. via one-hot encoded tissues) and then predicts the change in splicing behaviour between the two conditions.

[1]When choosing the samples to evaluate on from the different datasets, care needs to be taken to avoid a subtle data leak: despite the different primary data sources, many of the same cassette exons appear as training samples in multipe datasets. This initially lead to the surprising observation that it is better to train on a dataset different from the one the model is evaluated on. We avoid this issue by efficiently filtering all samples that are in the original training dataset from the three additional datasets we evaluate on via a hash set.

Model	AllPSI	LowPSI	HighPSI
DSC	0.244 ± 0.036	0.340 ± 0.039	-77.980 ± 13.631
D2V	0.120 ± 0.009	0.155 ± 0.010	-57.523 ± 2.771
RASC	**0.486** ± 0.067	**0.595** ± 0.074	**-25.470** ± 5.717

Table 5.4: Mean R^2 for PSI regression task on sensory neuron cell HipSci MAJIQ dataset. ± denotes standard deviation.

5.4.5 PSI Regression

After having achieved a good level of performance on the classification task, we turn to the regression task. We train the models on the sensory neuron cell HipSci MAJIQ dataset and give the results in Table 5.4.

The results show that the models still fail to explain a lot of the variance of the PSI value: no model is able to explain even half of it. As on the classification tasks, RASC outperforms the other models by a wide margin and all models are significantly worse on explaining the variance for highly included, rather than rarely to moderately included, non-constitutive exons. The latter observation is particularly pronounced: the R^2 score of all models on HighPSI is below 0, showing that they fail to explain the variance even as well as a baseline model whom consistently predicts the mean. Thus, we take these results as indication that the current splicing codes can still be significantly improved.

Model	Performance
BNN-UDC [31]	0.220
BNN-LMH [31]	0.368
DNN-PSI [31]	0.434
D2V [32]	0.594
W2V [32]	**0.680**

Table 5.5: Mean R^2 over 5 tissues for PSI regression task on mouse dataset as reported by [32].

To further compare our model to other work, we repeat the main results from [32] in Table 5.5. [32] used a dataset derived from mouse sequencing data, originally from [31]. On this dataset, D2V is the second-best performing model, only moderately outperformed by a more finer-grained word2vec model (denoted W2V).

Nonetheless, D2V significantly outperforms all other models from the literature it was compared to. However, D2V is significantly outperformed by RASC on our dataset. This indicates that RASC would compare very favorably against all five models also evaluated in [32].

We also observe that the R^2 values reported by [32] are significantly higher than on our dataset: the best performing model W2V is able to explain nearly 70% of the variance. We consider multiple possible explanations for this:

- Splicing behaviour on the mouse dataset is easier to predict because of data processing differences. Like our work, [32] use MAJIQ for data processing. While they do not explicitly state the version of MAJIQ they used, MAJIQ has received various updates since their publication. These updates include improvements to the splicing detection algorithm which could affect the outcome of data processing.
- Splicing behaviour on the mouse dataset is easier to predict because of dataset composition differences. The mouse dataset only includes cassette exons, while our dataset also includes constitutive non-cassette exons. Since splicing motifs between cassette and non-cassette exons likely differ, the learning task of models on our dataset could be more challenging since they have to learn to recognize both. As a counter point, the mouse dataset only contains 10,000 training samples which might be too few samples to train RASC and DSC.
- Splicing behaviour on the mouse dataset is easier to predict because of different model inputs. [32]'s models D2V and W2V take 600 nucleotide windows, instead of 140 nucleotide windows, around the exon start and end sites as input. In addition, they are also given two analogue 600 nucleotide windows from the exon end immediately to 'the left' and the exon end immediately to 'the right' from the cassette exon. In total their models are given 2400 nucleotides as input, while we give our models a total of 280 nucleotides as input. These additional nucleotides could contain information about splicing behaviour which is not contained in the nucleotide windows given to the

models when we evaluated them and may help to equalize the performance between the mouse dataset and our dataset.

- Splicing behaviour on the mouse dataset is easier to predict because it is less complex. Increased relative prevalence in exon skipping has been observed for more complex eukaryotic organisms [15], giving plausibility to this explanation. On the other hand, we observed nearly no cross-tissue performance differences between tissues, when cross-tissue splicing behaviour is also known to vary [107]. This hypothesis is incredibly challenging to evaluate, as there are many dimensions to splicing and there is no universal measure for splicing complexity (nor likely ever will be). However, due to the other possible explanations we explored and due to the low cross-tissue performance differences we observed, we evaluate it as unlikely.

While most of these possible explanations are difficult to falsify, increasing the amount of sequence information given to the models is a simple to implement and promising avenue of research, which we leave to future work.

5.4.6 Interpreting RASC

We now analyze which regions of the input sequence RASC attends to.

Heads focus on a single sequence

When using a single attention head, the head needs to split its conceptual attention over the start and end input sequence to derive information from both. This is not the case for multiple attention heads: single heads may focus on only one sequence, as long as at least one other head attends to the other sequence. We observe exactly this behaviour in Figure 4.15: three heads attend soley to the start sequence and one head attends solely to the end sequence. Across the 9 cross-validation runs, single attention heads only split their conceptual attention 7 out of 36 times. Additionally, all trained versions of RASC always attend to

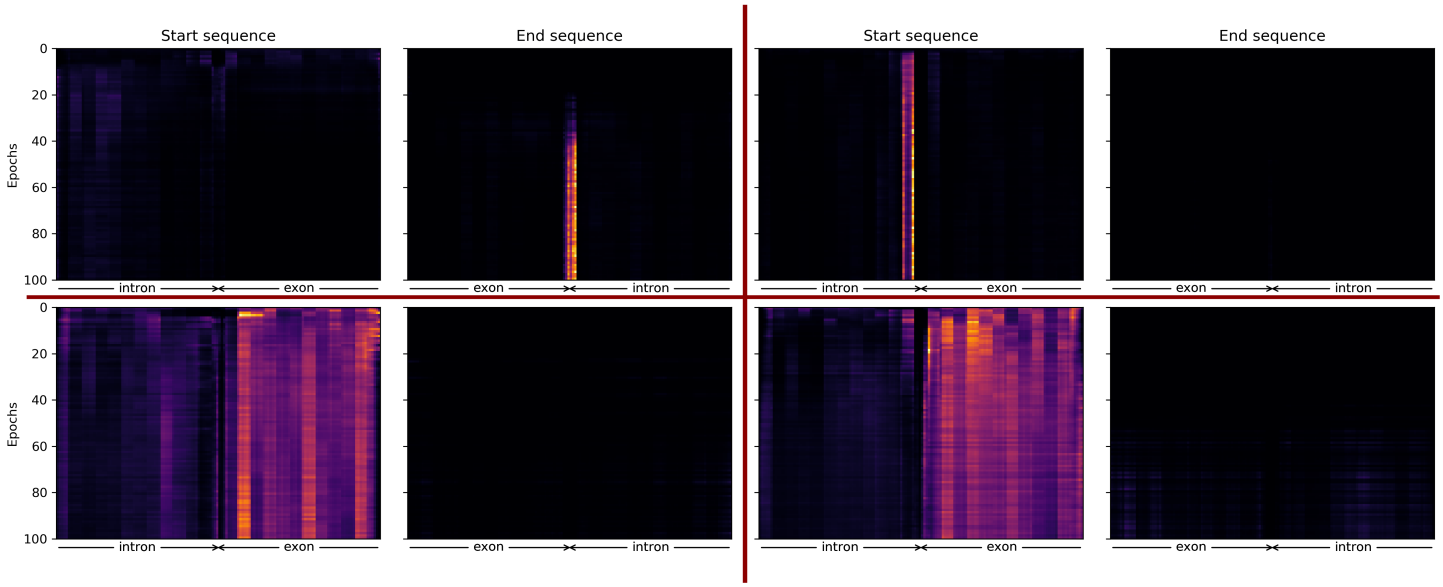

Figure 5.14: Each of the four quadrants (marked by the red grid) shows the distribution of the attention weights for a single attention head. For each head, one heatmap for the start sequence and one heatmap for the end sequence is given. Each heapmap illustrates how the distribution of attention weights develops during training. The mean distribution of the attention weights over the test set is displayed.

both sequences, reassuring us that the dataset isn't biased such that RASC only needs to take information from one sequence into account.

We empirically observe that the sequence which an attention head focuses on generally does not change after the first 5 epochs. Similar observations, that the role of attention heads is decided very early on in training, have also been made in the context of NLP [96].

What does the model pay attention to?

Averaging the attention weights over all heads and over all cross-validation runs at the end of training, we visualize the attention weights assigned to the 20 positions around the exon and intron boundaries in Figure 5.15. We chose to visualize the attention weights for these positions because they are the most attended to (we provide the attention distribution over the full sequence in the appendix in Figure 7.1).

The focus of the heads on regions around the boundaries align well with studies showing that the alternative splicing of exons is primarily affected by its immediate

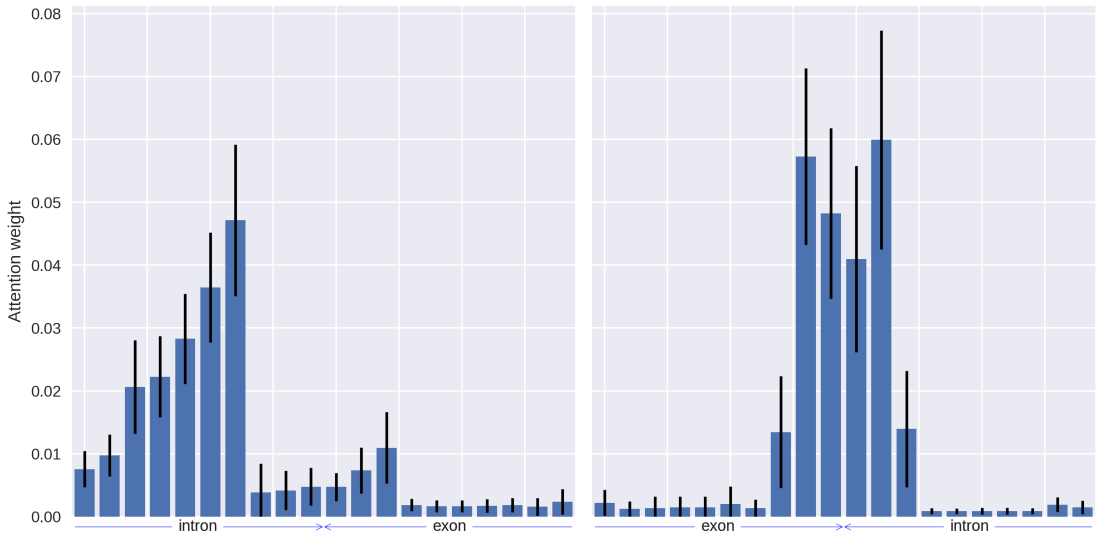

Figure 5.15: Mean attention weights averaged over all test samples, all four attention heads and all cross-validation runs. Each bar represents the attention given to one element from the 280 elements given to the attention layer. The 20 elements closest to the two boundaries are shown. The error bars represent the standard deviation between cross-validation runs.

sequence context [40] [109] [110]. However, care should be taken when interpreting these attention distributions. The attention weights do not directly correspond to the relative importance of each position for multiple reasons:

(a) A position, as displayed in Figure 5.15, may contain the information from multiple neighbouring positions in the sequence. This is because the representation at a given position is computed by a BiLSTM unit which may encode information about the preceding and following positions into the representation for the given position as well.

(b) The shown attention weights are a mean between all test set samples and all training runs. Therefore, they do not correspond to the attention weights used by any particular model for any particular training sample.

(c) The variance between runs is large: generally 20% of the mean value. A contributing factor to the large variance is that the attention distribution of the model likely shifts based on which information is encoded in which position (and this may differ between runs). Thus this observation is also

related to (a).

(d) The displayed attention weights are a mean between the four attention heads. The results from Figure 5.10 show that RASC's performance is similar with multiple and only one attention head. Thus, there is likely some redundant overlap between the attended to regions of different heads, e.g., two heads may simultaneously attend to the region close to the exon/intron boundary even though only one head would need to attend to it. Due to this redundancy, the overlap is free to shift between runs likely introducing further noise into the attention weights we observe. Additionally, while we averaged the attention weights between heads, RASC weights them based on the learned matrix W^O which might not apply an uniform weighting, e.g., it might assign more weight to heads if they are the only head attending to a particular region.

Thus, while the results in Figure 5.15 can serve to identify a general trend regarding which sequence regions contain the information most relevant to splicing behaviour, they should not be the basis for specific conclusions about the relative importance of specific nucleotide positions.

If the goal is working towards such an interpretation (based on a deep learning model), a BiLSTM unit should not be used to find representations for the nucleotides. A fully connected-model which does not mix information between positions until the attention layer would be better suited. Additionally, only one attention head should be used to avoid issues when averaging the attention of multiple heads.

Further modifications, which do not require architectural changes, could include computing the mean attention weights over more cross-validation runs to make the attention distribution estimate more robust, and only interpreting the attention weights for one training sample. The application of methods visualizing the most important input features for a specific prediction such as DeepLIFT [111] is another possibility.

Oooooh, the weary traveller draws close to the end of the path.

— Izaro, Emperor of the Eternal Empire

6 Conclusion

6.1 Summary

This work approached the task of predicting alternative splicing behaviour from a deep learning perspective. We discussed key challenges which should be addressed when estimating PSI, implemented a method for PSI estimation and constructed splicing quantification datasets based on processing with our PSI estimation method, SUPPA and MAJIQ. We proposed RASC: a novel computational splicing code which is the first to introduce an attention mechanism to splicing quantification and reimplemented the two baselines models DSC and D2V.

Evaluating the models on HEXEvent, we showed that HEXEvent is confounded and that the performance of DSC on HEXEvent can be matched by extremely simple MLPs with two to three orders of magnitudes fewer parameters which use no sequence information. Moving to the datasets we constructed, we found that only the HipSci MAJIQ dataset provides data of appropriate quality and quantity. Evaluating RASC on HipSci MAJIQ, RASC outperforms DSC and D2V by at least 15% each. In further experiments, we showed that RASC generalizes extremely well to different conditions, demonstrated that it has high specificity and gave evidence that RASC also compares favourably to other methods from the literature

when regressing PSI. Finally, we showed that the nucleotides around exon and intron boundaries are most attended to by RASC.

6.2 Discussion and Future Work

6.2.1 The need for better datasets

Our work further corroborates the lack of publically available, standardized datasets suitable for the training of Machine Learning-based splicing codes that was already lamented during the introduction of HEXEvent in 2013 [41]. The most commonly used dataset to train splicing codes is based on mouse, instead of human, data and not publically made available in an accessible format [31]. As a result, many papers introducing new splicing codes attempt to reconstruct this dataset [32], use HEXEvent [40] or construct their own dataset ad hoc [112]. This places the additional burden of dataset construction on each author and makes comparisons only indirectly possible or flawed, when implementation differences lead to different versions of the same dataset. In contrast, the wide use of standardized, publically available datasets would allow a quicker iteration of ideas and a fair comparison between them. For these reasons, we believe that a concerted effort to construct high-quality, standardized datasets for the training of machine learning, and particular deep learning, splicing codes should be undertaken. We plan to make our HipSci MAJIQ datasets, as well as our code base, publically available as a stepping stone towards this goal.

6.2.2 Possible improvements to RASC

While RASC improves upon previous models by a wide margin, there is still scope to improve its prediction accuracy and there are many avenues for future work.

RASC's predictive power could further be improved by incorporating additional information sources known to affect splicing, like the tissue of an exon or chromatin states [113]. Increasing the amount of sequence information given to RASC is

another simple, yet promising idea. Futhermore, even hyperparameter tuning may lead to performance improvements since our hyperparameter tuning was limited due to computational constraints. Experiments maxing out the batch size (and stabilizing training) or evaluating different learning rate schedules would be particularly interesting.

Like all deep learning models, RASC suffers from poor interpretability. While we have taken some first steps towards interpreting RASC by analyzing what parts of a sequence it attends to, a lot more work remains to be done. We discussed changes to RASC's model architecture to make its interpretation more reliable, such as replacing the BiLSTM units and using only one attention head. Further inspiration could be taken from methods exploring the inner workings of Transformer models in NLP [114]. General neural network visualization techniques such as DeepLIFT [111] which visualize the relative importance of the inputs for a specific prediction could also be used.

With a view to its practical application, adapting RASC for differential splicing prediction is a promising avenue of future research. Here the existing tool MAJIQ could be exploited to generate a dataset for splicing changes between conditions. A model capable of predicting the impact of genetic variations on splicing would be extremely valuable for the emerging field of personalized medicine.

7 Appendix

7.1 Accession Numbers of HipSci data

The ENA Accession Numbers of the 25 biological replicates belonging to sensory neuron cell lines [50] are ERR177-: 5544, 5551, 5552, 5554, 5594. 5595, 5596, 5598, 5600, 5601, 5631, 5634, 5637, 5638, 5640, 5641, 5643, 5644, 5684, 5685, 5686, 5687, 5688, 5689, 5693. While all of these were used in MAJIQ Builder process (and thus contributed to the constitutive exons), only the sample with Accession Number ERR1775544 was used with the MAJIQ PSI step (and thus determined the alternatively spliced exons).

The ENA Accession Numbers of the 20 biological replicates belonging to undifferentiated iPSC cell lines [43] are ERR-: 914342, 946968, 946976, 946983, 946984, 946990, 946992, 946994, 947011, 1203463, 1243454, 1274914, 1274917, 1724696, 1724699, 1743789, 2039345, 2039336, 2278244 2278245. Similarly, all of the above biological replicates were used in the MAJIQ Builder process. Only the samples with Accession Numbers ERR946992, ERR946984 (same cell type and donor as ERR946984, but different cell line), and ERR946968 (same cell type, but different donor) were used with MAJIQ PSI.

7.2 Additional Doc2Vec training details

Training method	DM
Embedding dimensions	100
Corpus	Human Genome GRCh38
Window size	5
Minimum count	5
Negative sampling	5
Epochs	5

Table 7.1: Exact hyperparameters used for training Doc2Vec model.

The hyperparameters used during pre-training are given in Table 7.1. Except for the number of epochs, these are the same as in the baseline paper [32]. We reduced the number of epochs from 20 to 5, as initial tests showed no performance difference between these two values. However, while [32] don't mention what Doc2Vec implementation they use, almost all of these parameters are the same as the default parameters from the gensim library. Therefore, we believe that these parameters weren't fine-tuned very intensively.

Two hyperparameters, not yet introduced, are mentioned in Table 7.1. Although these hyperparameters aren't very impactful when training on genomic data (as the vocabulary only consists of 64 words), we mention them for completeness:

- The minimum count parameters eliminates all words which occur fewer than the minumum amount of times from the corpus. Infrequent words don't have enough examples to allow the model to learn a good representation of them. Additionally, while the individual words themselves are uncommon, the total amount of uncommong words might still be large, making it computationally expensive to keep them.
- Negative sampling [34] is a technique to reduce the computational cost of backpropagating the gradient updates. The size of the weights in the Word2Vec or Doc2Vec can easily reach millions of learnable weights with a medium sized vocabulary: for 10,000 words in the vocabulary and 300-dimensional embedding, the matrix representing the hidden weights already has 3 million

weights. This makes backpropagation expensive as all of these weights need to be updated in every step.

When negative sampling is enabled, by default only the weights connected to the word the network should predict will be updated via backpropagation. Additionally, a certain number of negative samples, words which the network shouldn't predict, are randomly chosen and their weights are updated too. This dramatically reduces the computational cost of backpropagating the gradient updates, since only the weights of very few words in the vocabulary are updated.

7.3 Distribution of attention weights

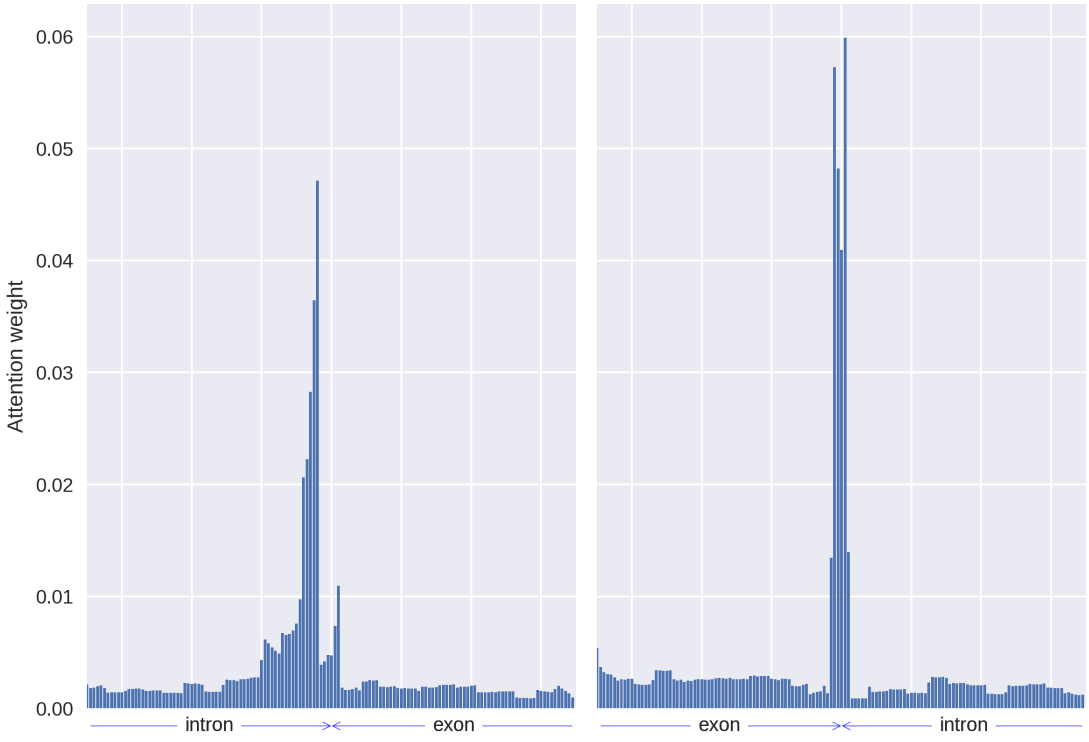

Figure 7.1: Mean attention weights averaged over all test samples, all cross-validation runs and all four attention heads at the end of training.

References

[1] W. Gilbert. "Why genes in pieces?" In: *Nature* 271.5645 (Feb. 1978), p. 501 (cit. on pp. 1, 7).

[2] S. M. Berget, C. Moore, and P. A. Sharp. "Spliced segments at the 5' terminus of adenovirus 2 late mRNA". In: *Proc. Natl. Acad. Sci. U.S.A.* 74.8 (Aug. 1977), pp. 3171–3175 (cit. on pp. 1, 2, 7).

[3] Ben Smithers, Matt Oates, and Julian Gough. "'Why genes in pieces?'—revisited". In: *Nucleic Acids Research* 47.10 (2019), pp. 4970–4973 (cit. on pp. 1, 7).

[4] B. M. Brinkman. "Splice variants as cancer biomarkers". In: *Clin. Biochem.* 37.7 (July 2004), pp. 584–594 (cit. on pp. 2, 8).

[5] Qiang Xu and Christopher Lee. "Discovery of novel splice forms and functional analysis of cancer-specific alternative splicing in human expressed sequences". In: *Nucleic acids research* 31 (Oct. 2003), pp. 5635–43 (cit. on pp. 2, 8).

[6] Núria López-Bigas et al. "Are splicing mutations the most frequent cause of hereditary disease?" In: *FEBS letters* 579.9 (2005), pp. 1900–1903 (cit. on pp. 2, 8).

[7] Lee P Lim and Christopher B Burge. "A computational analysis of sequence features involved in recognition of short introns". In: *Proceedings of the National Academy of Sciences* 98.20 (2001), pp. 11193–11198 (cit. on p. 2).

[8] 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 (cit. on p. 2).

[9] Y. Barash et al. "Deciphering the splicing code". In: *Nature* 465.7294 (May 2010), pp. 53–59 (cit. on pp. 2, 5, 9).

[10] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *Nature* 521.7553 (2015), pp. 436–444 (cit. on pp. 2, 52).

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105 (cit. on p. 2).

[12] Taeho Jo, Kwangsik Nho, and Andrew J. Saykin. "Deep Learning in Alzheimer's Disease: Diagnostic Classification and Prognostic Prediction Using Neuroimaging Data". In: *Frontiers in Aging Neuroscience* 11 (2019), p. 220 (cit. on p. 2).

[13] Timnit Gebru et al. "Using deep learning and Google Street View to estimate the demographic makeup of neighborhoods across the United States". In: *Proceedings of the National Academy of Sciences* 114.50 (2017), pp. 13108–13113 (cit. on p. 2).

[14] National Human Genome Research Institute. *Antisense*. [Online; accessed 11-September-2020]. n.d. URL: genome.gov/genetics-glossary/antisense (cit. on p. 4).

[15] E. Kim, A. Goren, and G. Ast. "Alternative splicing: current perspectives". In: *Bioessays* 30.1 (Jan. 2008), pp. 38–47 (cit. on pp. 5–8, 14, 75).

[16] Mutundis. *File:Pre-mRNA to mRNA MH.svg*. [Online; accessed 23-August-2020]. 2015. URL: commons.wikimedia.org/wiki/File:Pre-mRNA_to_mRNA_MH.svg (cit. on p. 5).

[17] L. Liu et al. "Aberrant regulation of alternative pre-mRNA splicing in hepatocellular carcinoma". In: *Crit. Rev. Eukaryot. Gene Expr.* 24.2 (2014), pp. 133–149 (cit. on p. 6).

[18] N. L. Barbosa-Morais et al. "The evolutionary landscape of alternative splicing in vertebrate species". In: *Science* 338.6114 (Dec. 2012), pp. 1587–1593 (cit. on p. 6).

[19] C. W. Sugnet et al. "Transcriptome and genome conservation of alternative splicing events in humans and mice". In: *Pac Symp Biocomput* (2004), pp. 66–77 (cit. on p. 6).

[20] Jamal Tazi, Nadia Bakkour, and Stefan Stamm. "Alternative splicing and disease". In: *Biochimica et Biophysica Acta (BBA)-Molecular Basis of Disease* 1792.1 (2009), pp. 14–26 (cit. on p. 8).

[21] C. F. Rowlands, D. Baralle, and J. M. Ellingford. "Machine Learning Approaches for the Prioritization of Genomic Variants Impacting Pre-mRNA Splicing". In: *Cells* 8.12 (Nov. 2019) (cit. on p. 8).

[22] K. Q. Le et al. "Alternative splicing as a biomarker and potential target for drug discovery". In: *Acta Pharmacol. Sin.* 36.10 (Oct. 2015), pp. 1212–1218 (cit. on p. 8).

[23] Charlie F Rowlands, Diana Baralle, and Jamie M Ellingford. "Machine learning approaches for the prioritization of genomic variants impacting pre-mrna splicing". In: *Cells* 8.12 (2019), p. 1513 (cit. on p. 8).

[24] Johann Wolfgang von Goethe. *Wilhelm Meisters Wanderjahre oder die Entsagenden*. de. Cotta, 1829 (cit. on p. 9).

[25] Y. Barash, B. J. Blencowe, and B. J. Frey. "Model-based detection of alternative splicing signals". In: *Bioinformatics* 26.12 (June 2010), pp. i325–333 (cit. on p. 9).

[26] J. P. Venables et al. "Identification of alternative splicing markers for breast cancer". In: *Cancer Res.* 68.22 (Nov. 2008), pp. 9525–9531 (cit. on p. 9).

[27] M. R. Gazzara et al. "In silico to in vivo splicing analysis using splicing code models". In: *Methods* 67.1 (May 2014), pp. 3–12 (cit. on pp. 10, 11).

[28] H. Y. Xiong et al. "RNA splicing. The human splicing code reveals new insights into the genetic determinants of disease". In: *Science* 347.6218 (Jan. 2015), p. 1254806 (cit. on p. 10).

[29] H. Y. Xiong, Y. Barash, and B. J. Frey. "Bayesian prediction of tissue-regulated splicing using RNA sequence and cellular context". In: *Bioinformatics* 27.18 (Sept. 2011), pp. 2554–2562 (cit. on p. 10).

[30] M. K. Leung et al. “Deep learning of the tissue-regulated splicing code”. In: *Bioinformatics* 30.12 (June 2014), pp. i121–129 (cit. on pp. 10, 11).

[31] A. Jha, M. R. Gazzara, and Y. Barash. “Integrative deep models for alternative splicing”. In: *Bioinformatics* 33.14 (July 2017), pp. i274–i282 (cit. on pp. 10, 11, 24, 54, 73, 80).

[32] M. Oubounyt et al. “Deep Learning Models Based on Distributed Feature Representations for Alternative Splicing Prediction”. In: *IEEE Access* 6 (2018), pp. 58826–58834 (cit. on pp. 11, 36, 38, 54, 66, 73, 74, 80, 83).

[33] Tomas Mikolov et al. “Efficient Estimation of Word Representations in Vector Space”. In: *arXiv e-prints*, arXiv:1301.3781 (Jan. 2013), arXiv:1301.3781. arXiv: `1301.3781 [cs.CL]` (cit. on pp. 11, 35, 36).

[34] Tomas Mikolov et al. “Distributed Representations of Words and Phrases and their Compositionality”. In: *arXiv e-prints*, arXiv:1310.4546 (Oct. 2013), arXiv:1310.4546. arXiv: `1310.4546 [cs.CL]` (cit. on pp. 11, 35, 36, 83).

[35] Quoc V. Le and Tomas Mikolov. “Distributed Representations of Sentences and Documents”. In: *arXiv e-prints*, arXiv:1405.4053 (May 2014), arXiv:1405.4053. arXiv: `1405.4053 [cs.CL]` (cit. on pp. 11, 36).

[36] Ryan Kiros et al. “Skip-Thought Vectors”. In: *arXiv e-prints*, arXiv:1506.06726 (June 2015), arXiv:1506.06726. arXiv: `1506.06726 [cs.CL]` (cit. on pp. 11, 36).

[37] C. Szegedy et al. “Rethinking the Inception Architecture for Computer Vision”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 2818–2826 (cit. on p. 11).

[38] P. J. Shepard et al. “Efficient internal exon recognition depends on near equal contributions from the 3’ and 5’ splice sites”. In: *Nucleic Acids Res.* 39.20 (Nov. 2011), pp. 8928–8937 (cit. on p. 11).

[39] A. Busch and K. J. Hertel. “Splicing predictions reliably classify different types of alternative splicing”. In: *RNA* 21.5 (May 2015), pp. 813–823 (cit. on pp. 11, 14, 32, 54, 59).

[40] Z. Louadi et al. “Deep Splicing Code: Classifying Alternative Splicing Events Using Deep Learning”. In: *Genes (Basel)* 10.8 (Aug. 2019) (cit. on pp. 12, 14, 22, 23, 28, 29, 33, 35, 54–58, 61, 77, 80).

[41] A. Busch and K. J. Hertel. “HEXEvent: a database of Human EXon splicing Events”. In: *Nucleic Acids Res.* 41.Database issue (Jan. 2013), pp. D118–124 (cit. on pp. 14, 80).

[42] L. J. Carithers et al. “A Novel Approach to High-Quality Postmortem Tissue Procurement: The GTEx Project”. In: *Biopreserv Biobank* 13.5 (Oct. 2015), pp. 311–319 (cit. on p. 14).

[43] I. Streeter et al. “The human-induced pluripotent stem cell initiative-data resources for cellular genetics”. In: *Nucleic Acids Res.* 45.D1 (Jan. 2017), pp. D691–D697 (cit. on pp. 14, 82).

[44] Frederick Sanger, Steven Nicklen, and Alan R Coulson. “DNA sequencing with chain-terminating inhibitors”. In: *Proceedings of the national academy of sciences* 74.12 (1977), pp. 5463–5467 (cit. on p. 15).

[45] Jerzy K Kulski. "Next-generation sequencing—an overview of the history, tools, and "Omic" applications". In: *Next Generation Sequencing–Advances, Applications and Challenges* (2016), pp. 3–60 (cit. on p. 15).

[46] Shivashankar H. Nagaraj, Robin B. Gasser, and Shoba Ranganathan. "A hitchhiker's guide to expressed sequence tag (EST) analysis". In: *Briefings in Bioinformatics* 8.1 (May 2006), pp. 6–21 (cit. on p. 15).

[47] M Bonaldo, Greg Lennon, and Marcelo Soares. "Normalization and Subtraction: Two Approaches to Facilitate Gene Discovery". In: *Genome research* 6 (Oct. 1996), pp. 791–806 (cit. on p. 15).

[48] GTEx Consortium. *The GTEx Project: Datasets.* [Online; accessed 26-June-2020]. 2020. URL: `gtexportal.org/home/datasets` (cit. on p. 16).

[49] Y tambe. *File:Induction of iPS cells.svg.* [Online; accessed 23-August-2020]. 2007. URL: `commons.wikimedia.org/wiki/File:Pre-mRNA_to_mRNA_MH.svg` (cit. on p. 16).

[50] Jeremy Schwartzentruber et al. "Molecular and functional variation in iPSC-derived sensory neurons". In: *bioRxiv* (2017). eprint: `https://www.biorxiv.org/content/early/2017/01/06/095943.full.pdf` (cit. on pp. 17, 82).

[51] Koen Van den Berge et al. "RNA Sequencing Data: Hitchhiker's Guide to Expression Analysis". In: *Annual Review of Biomedical Data Science* 2.1 (July 2019), pp. 139–173 (cit. on p. 18).

[52] S. Schafer et al. "Alternative Splicing Signatures in RNA-seq Data: Percent Spliced in (PSI)". In: *Curr Protoc Hum Genet* 87 (Oct. 2015), pp. 1–11 (cit. on pp. 18, 20, 21).

[53] Yuval Benjamini and Terence P. Speed. "Summarizing and correcting the GC content bias in high-throughput sequencing". In: *Nucleic Acids Research* 40.10 (Feb. 2012), e72–e72 (cit. on p. 21).

[54] Wei Zheng, Lisa Chung, and Hongyu Zhao. "Bias detection and correction in RNA-Sequencing data". In: *BMC bioinformatics* 12 (July 2011), p. 290 (cit. on p. 21).

[55] Allison Piovesan et al. "Identification of minimal eukaryotic introns through GeneBase, a user-friendly tool for parsing the NCBI Gene databank". In: *DNA Research* 22 (Nov. 2015) (cit. on p. 23).

[56] GTEx Consortium. *The GTEx Project: Documentation.* [Online; accessed 26-August-2020]. 2020. URL: `gtexportal.org/home/documentationPage` (cit. on p. 23).

[57] J. L. Trincado et al. "SUPPA2: fast, accurate, and uncertainty-aware differential splicing analysis across multiple conditions". In: *Genome Biol.* 19.1 (Mar. 2018), p. 40 (cit. on p. 24).

[58] J. Vaquero-Garcia et al. "A new view of transcriptome complexity and regulation through the lens of local splicing variations". In: *Elife* 5 (Feb. 2016), e11752 (cit. on pp. 24, 26).

[59] Rob Patro et al. "Salmon provides fast and bias-aware quantification of transcript expression". In: *Nature methods* 14.4 (2017), pp. 417–419 (cit. on p. 25).

[60] Gael P Alamancos et al. “Leveraging transcript quantification for fast computation of alternative splicing profiles”. In: *Rna* 21.9 (2015), pp. 1521–1531 (cit. on p. 25).

[61] Ana Conesa et al. “A survey of best practices for RNA-seq data analysis”. In: *Genome biology* 17.1 (2016), p. 13 (cit. on p. 25).

[62] Michael Clark et al. “Long-read sequencing reveals the complex splicing profile of the psychiatric risk gene CACNA1C in human brain”. In: *Molecular Psychiatry* 25 (Nov. 2019), pp. 1–11 (cit. on p. 25).

[63] E. Afgan et al. “The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2016 update”. In: *Nucleic Acids Res.* 44.W1 (July 2016), W3–W10 (cit. on p. 27).

[64] Eric Lander et al. “Initial sequencing and analysis of the human genome”. In: *Nature* 409 (Mar. 2001), pp. 860–921 (cit. on p. 27).

[65] A. Dobin et al. “STAR: ultrafast universal RNA-seq aligner”. In: *Bioinformatics* 29.1 (Jan. 2013), pp. 15–21 (cit. on p. 27).

[66] Ying Cui, Meng Cai, and H. Stanley. “Comparative Analysis and Classification of Cassette Exons and Constitutive Exons”. In: *BioMed Research International* 2017 (Dec. 2017), pp. 1–8 (cit. on pp. 29, 59).

[67] Seung Gu Park and Sridhar Hannenhalli. “First intron length in mammals is associated with 5’ exon skipping rate”. In: *bioRxiv* (2015) (cit. on p. 30).

[68] Kristi L. Fox-Walsh et al. “The architecture of pre-mRNAs affects mechanisms of splice-site pairing”. In: *Proceedings of the National Academy of Sciences* 102.45 (2005), pp. 16176–16181 (cit. on pp. 30, 63).

[69] Sahar Gelfman et al. “Changes in exon–intron structure during vertebrate evolution affect the splicing pattern of exons”. In: *Genome research* 22 (Jan. 2012), pp. 35–50 (cit. on p. 30).

[70] M. K. Sakharkar, V. T. Chow, and P. Kangueane. “Distributions of exons and introns in the human genome”. In: *In Silico Biol. (Gedrukt)* 4.4 (2004), pp. 387–393 (cit. on p. 30).

[71] R. Milo et al. “BioNumbers–the database of key numbers in molecular and cell biology”. In: *Nucleic Acids Res.* 38.Database issue (Jan. 2010). `bionumbers.hms.harvard.edu/bionumber.aspx?id=105336&ver=6`, pp. D750–753 (cit. on pp. 30, 37).

[72] Jie Wu et al. “Splice Trap: A method to quantify alternative splicing under single cellular conditions”. In: *Bioinformatics (Oxford, England)* 27 (Sept. 2011), pp. 3010–6 (cit. on p. 32).

[73] Shihao Shen et al. “MATS: a Bayesian framework for flexible detection of differential alternative splicing from RNA-Seq data”. In: *Nucleic Acids Research* 40.8 (Jan. 2012), e61–e61 (cit. on p. 32).

[74] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting”. In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958 (cit. on p. 33).

[75] Vinod Nair and Geoffrey E Hinton. “Rectified linear units improve restricted boltzmann machines”. In: *ICML*. 2010 (cit. on p. 33).

[76] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. "Sequence to Sequence Learning with Neural Networks". In: *arXiv e-prints*, arXiv:1409.3215 (Sept. 2014), arXiv:1409.3215. arXiv: 1409.3215 [cs.CL] (cit. on pp. 35, 41).

[77] Ashish Vaswani et al. "Attention Is All You Need". In: *arXiv e-prints*, arXiv:1706.03762 (June 2017), arXiv:1706.03762. arXiv: 1706.03762 [cs.CL] (cit. on pp. 35, 42, 44, 45, 47).

[78] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *arXiv e-prints*, arXiv:1810.04805 (Oct. 2018), arXiv:1810.04805. arXiv: 1810.04805 [cs.CL] (cit. on pp. 35, 42).

[79] W Kent et al. "The human genome browser at UCSC". In: *Genome research* 12 (July 2002), pp. 996–1006 (cit. on p. 36).

[80] Patrick Ng. "dna2vec: Consistent vector representations of variable-length k-mers". In: *ArXiv* abs/1701.06279 (2017) (cit. on p. 37).

[81] Ehsaneddin Asgari and Mohammad RK Mofrad. "Continuous distributed representation of biological sequences for deep proteomics and genomics". In: *PloS one* 10.11 (2015), e0141287 (cit. on p. 37).

[82] Laurens van der Maaten and Geoffrey Hinton. "Viualizing data using t-SNE". In: *Journal of Machine Learning Research* 9 (Nov. 2008), pp. 2579–2605 (cit. on p. 39).

[83] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780 (cit. on pp. 41, 50).

[84] Kyunghyun Cho et al. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: *arXiv e-prints*, arXiv:1406.1078 (June 2014), arXiv:1406.1078. arXiv: 1406.1078 [cs.CL] (cit. on p. 41).

[85] Yonghui Wu et al. "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation". In: *arXiv e-prints*, arXiv:1609.08144 (Sept. 2016), arXiv:1609.08144. arXiv: 1609.08144 [cs.CL] (cit. on p. 42).

[86] Sebastian Ruder. *A Review of the Neural History of Natural Language Processing*. ruder.io/a-review-of-the-recent-history-of-nlp/. 2018 (cit. on pp. 42, 44).

[87] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: *arXiv e-prints*, arXiv:1409.0473 (Sept. 2014), arXiv:1409.0473. arXiv: 1409.0473 [cs.CL] (cit. on pp. 42, 43).

[88] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. "Effective Approaches to Attention-based Neural Machine Translation". In: *arXiv e-prints*, arXiv:1508.04025 (Aug. 2015), arXiv:1508.04025. arXiv: 1508.04025 [cs.CL] (cit. on p. 42).

[89] Denny Britz. "Deep Learning's Most Important Ideas - A Brief Historical Review". In: (2020). URL: dennybritz.com/blog/deep-learning-most-important-ideas (cit. on p. 42).

[90] Tom B Brown et al. “Language models are few-shot learners”. In: *arXiv preprint arXiv:2005.14165* (2020) (cit. on pp. 42, 44).

[91] Sungyong Seo et al. “Interpretable convolutional neural networks with dual local and global attention for review rating prediction”. In: *Proceedings of the Eleventh ACM Conference on Recommender Systems*. 2017, pp. 297–305 (cit. on p. 43).

[92] Jan Chorowski et al. “Attention-Based Models for Speech Recognition”. In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*. NIPS 2015. Montreal, Canada: MIT Press, 2015, pp. 577–585 (cit. on p. 43).

[93] Victor Sanh et al. “DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter”. In: *arXiv e-prints*, arXiv:1910.01108 (Oct. 2019), arXiv:1910.01108. arXiv: `1910.01108 [cs.CL]` (cit. on p. 44).

[94] Gábor Melis, Tomáš Kočiský, and Phil Blunsom. “Mogrifier LSTM”. In: *arXiv e-prints*, arXiv:1909.01792 (Sept. 2019), arXiv:1909.01792. arXiv: `1909.01792 [cs.CL]` (cit. on p. 44).

[95] Daniel Keysers et al. “Measuring Compositional Generalization: A Comprehensive Method on Realistic Data”. In: *arXiv e-prints*, arXiv:1912.09713 (Dec. 2019), arXiv:1912.09713. arXiv: `1912.09713 [cs.LG]` (cit. on p. 44).

[96] Paul Michel, Omer Levy, and Graham Neubig. “Are sixteen heads really better than one?” In: *Advances in Neural Information Processing Systems*. 2019, pp. 14014–14024 (cit. on pp. 48, 76).

[97] Jim Clauwaert and Willem Waegeman. “Novel transformer networks for improved sequence labeling in genomics”. In: *bioRxiv* (2020) (cit. on pp. 48, 68, 69).

[98] Guillaume Klein et al. “OpenNMT: Open-Source Toolkit for Neural Machine Translation”. In: *Proc. ACL*. 2017. URL: `https://doi.org/10.18653/v1/P17-4012` (cit. on p. 49).

[99] Byunghan Lee et al. “DNA-Level Splice Junction Prediction using Deep Recurrent Neural Networks”. In: *arXiv e-prints*, arXiv:1512.05135 (Dec. 2015), arXiv:1512.05135. arXiv: `1512.05135 [cs.LG]` (cit. on p. 50).

[100] Mike Schuster and Kuldip Paliwal. “Bidirectional recurrent neural networks”. In: *Signal Processing, IEEE Transactions on* 45 (Dec. 1997), pp. 2673–2681 (cit. on p. 50).

[101] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *arXiv preprint arXiv:1502.03167* (2015) (cit. on p. 50).

[102] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035 (cit. on p. 51).

[103] Radim Řehůřek and Petr Sojka. “Software Framework for Topic Modelling with Large Corpora”. English. In: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. `is.muni.cz/publication/884893/en`. Valletta, Malta: ELRA, May 2010, pp. 45–50 (cit. on p. 51).

[104] Xiaokang Zhang et al. “Recognition of alternatively spliced cassette exons based on a hybrid model”. In: *Biochemical and Biophysical Research Communications* 471 (Feb. 2016) (cit. on p. 59).

[105] Juan A. Botia et al. “G2P: Using machine learning to understand and predict genes causing rare neurological disorders”. In: *bioRxiv* (2018) (cit. on p. 59).

[106] L. Li et al. “A classification of alternatively spliced cassette exons using AdaBoost-based algorithm”. In: *2014 IEEE International Conference on Information and Automation (ICIA)*. 2014, pp. 370–375 (cit. on p. 59).

[107] Gene Yeo et al. “Variation in alternative splicing across human tissues”. In: *Genome biology* 5.10 (2004), R74 (cit. on pp. 59, 72, 75).

[108] Maayan Amit et al. “Differential GC content between exons and introns establishes distinct strategies of splice-site recognition”. In: *Cell reports* 1.5 (2012), pp. 543–556 (cit. on p. 63).

[109] Eli Koren, Galit Lev-Maor, and Gil Ast. “The emergence of alternative 3’ and 5’ splice site exons from constitutive exons”. In: *PLoS Comput Biol* 3.5 (2007), e95 (cit. on p. 77).

[110] Peter Shepard et al. “Efficient internal exon recognition depends on near equal contributions of the 3’ and 5’ splice sites”. In: *Nucleic acids research* 39 (July 2011), pp. 8928–37 (cit. on p. 77).

[111] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. “Learning important features through propagating activation differences”. In: *arXiv preprint arXiv:1704.02685* (2017) (cit. on pp. 78, 81).

[112] H. Bretschneider et al. “COSSMO: predicting competitive alternative splice site selection using deep learning”. In: *Bioinformatics* 34.13 (July 2018), pp. i429–i437 (cit. on p. 80).

[113] Paulina Kolasinska-Zwierz et al. “Differential chromatin marking of introns and expressed exons by H3K36me3”. In: *Nature genetics* 41.3 (2009), p. 376 (cit. on p. 80).

[114] Kevin Clark et al. “What does bert look at? an analysis of bert’s attention”. In: *arXiv preprint arXiv:1906.04341* (2019) (cit. on p. 81).