Deep Learning for Alternative Splicing

Richard Brown

Brasenose College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity 2019

I dedicate this work to Elizabeth Brown

Acknowledgements

I would like to thank my supervisor Gerton Lunter for his help and support while doing this research and writing the thesis. His positivity and support during challenging moments was crucial for this work, and I cannot thank him enough. I would also like to express my appreciation to the rest of the Lunter Group, including our new members, with whom I have valued discussing research and other less serious topics over the last 4 years.

Further thanks to Jim, Damien and the rest of the Hughes lab. Their biological expertise and experimental skill has been humbling and impressive.

I also want to thank Robert Esnouf, Jon Diprose, and Colin Freeman who have been incredibly helpful to me over the last few years. The research computing systems they have built and maintained have been invaluable to my research and that of countless others. Further thanks to Tim Bardsley for help maintaining our group servers.

I finally want to thank my family and friends for their support throughout this time. Words cannot express my gratitude to you all.

Abstract

Mutations that affect RNA splicing can have severe phenotypic consequences, and contribute to rare and sporadic human disease. Whole-genome sequencing promises to improve diagnosis, but it is often difficult to identify mutations that disrupt splicing, except when they affect canonical donor and acceptor sites. In particular, cryptic mutations that cause novel splice junctions to appear deep within introns of genes are very hard to identify. Accurate computational models are therefore crucial for effective diagnosis.

The availability of large amounts of expression data across a range of genomes and cell types, and the development of neural network technologies that can learn features directly from data, together provide an opportunity for developing computational models that predict splicing activity directly from genome sequence.

In this work, I first develop and validate a methodology to improve the training of neural networks operating on sequence data using the formalism of equivariant maps. We demonstrated increased representational stability for networks constructed using these techniques and subsequently used this approach to discover and quantify novel binding sites for PRDM9.

Next, I develop models to predict splice site location and exon inclusion ratios simultaneously using modifications to the dilated convolutional network and show state-of-the-art performance at this task. I apply these models to mutation databases and find that the model can make interpretable predictions about the consequences of deep intronic mutations, explaining 27% of pathogenic cryptic splice variants.

Contents

Chapter 1

Introduction

Next-generation sequencing (NGS) technologies have dramatically reduced the cost of sequencing the human genome, from approximately $ 3 billion for the reference constructed in the human genome project [International Human Genome Sequencing Consortium and others, 2001] to thousands of dollars for the typical cancer or rare disease patient [Schwarze et al., 2019]. Additionally, there has been an explosion of lab-based techniques designed to capture additional associated information such as protein-genome interaction [Ren et al., 2000] or chromatin accessibility, allowing large scale consortium efforts [ENCODE Project Consortium and others, 2012, Lonsdale et al., 2013, Forrest et al., 2014] to compile extensive libraries of associated epigenetic information.

This explosion in data quantity and quality has caused corresponding improvements in the computational methods used to analyse it. It is now possible to fit much richer models than even a few years ago, with recent approaches containing tens of millions of parameters, allowing us to attempt to capture the nuances of complex biological processes.

We add to this effort in two ways: By developing methodology related to the fitting of these models in a fashion specific to the symmetries presented by the structure of DNA, and through the practical application of these techniques to model the process of RNA splicing, a fundamental driver of transcriptomic diversity.

In this thesis, I describe these two contributions in detail. In chapter 2 I give an outline of our approach to allow existing deep learning methods to be designed robustly for this problem domain. In chapter 3, I then give an empirical validation of this approach, both on simulated data, and on a biological dataset related to the locations of meiotic 'recombination hotspots'. Here, I show that these techniques lead to more accurate modelling and more transparent and robust interpretation of these highly parameterized models. I then use our model to discover a binding motif for PRDM9 not previously observed in this data, quantifying the importance of this binding motif to hotspot determination.

In chapter 4, I discuss our approach to predicting the locations of splice sites in large genomic windows purely from genomic sequence. I show that this approach gives formidable accuracy in comparison to current literature approaches, and go on to study how the model comes to its predictions. I find that polymorphisms predicted by the model to be splice altering are under purifying selection in human populations. Finally, I find that this approach gives us an ability to predict cryptic splice sites, and observe an excellent corroboration of our predictions in literature.

1.1 Approaches for Annotating Genomic Sequences

As mentioned above, large scale sequencing and annotation efforts are only one reason for the advancement in phenotypic models from genomic sequence. The other component to this is the substantial improvement in techniques and methodology related to machine learning, termed colloquially 'deep learning', in the last 5 years. The improvement of these tools has begun to allow us to consistently fit rich models capable of capturing potentially complex interactions, such as those between transcription factors with each other and other sequence elements [Zhou and Troyanskaya, 2015b, Alipanahi et al., 2015a]. Here, we briefly discuss these advancements and introduce some of the methods that are used throughout this work.

1.1.1 Classical Approaches to Genomic Sequence Classification

A common task in genomics is assigning sections of genomic sequence into classes. These classes can represent membership of a regulatory group, such as a promoter or enhancer, and therefore, one can use them to annotate the genome.

Classical approaches are varied in implementation, but a common theme is the need to manually 'featurize' (determine important features of the sequence) the sequences in question. A very common approach used to extract features from sequences is the 'spectrum kernel' [Leslie et al., 2001], this method converts a string composed of letters in an alphabet A into a vector of features in the following way: Imagine a substring of length k, $a \in A^k$, we calculate the function ϕ_a on the string x as

$$\phi_a(x) = \#\text{occurances of } a \text{ in } x \tag{1.1}$$

So that for the genomic alphabet $A = [A, C, G, T]$, and sequence $x = CAGTA$, we

have

$$\phi_A(x) = 2$$

$$\phi_{CA}(x) = 1$$

Using this function, one can then calculate a feature vector for any input string as

$$\Phi_k(x) = \left[\phi_{a_1}(x), \phi_{a_2}(x) \dots \phi_{a_{|A|^k}}(x)\right] \tag{1.2}$$

i.e. calculating the spectrum kernel for each string $a \in \{A, C, G, T\}^k$ for a specified substring length k, to be used with classical machine learning methods. This approach found a huge number of successful applications, such as homology detection [Saigo et al., 2004], predicting protein-ligand interaction [Jacob and Vert, 2008] and protein-protein interaction [Ben-Hur and Noble, 2006]. Despite this, it suffers some serious downsides. Firstly, the number of unique features grows exponentially as a function of k as $|A|^k$, making it challenging to capture longer motifs (we later find 21-22 base motifs, so would require a total of 4,398,046,511,104 features to represent these). Additionally, this approach doesn't allow the sharing of information between similar motifs probably corresponding to the same semantic features. This latter problem is partially addressed in numerous methods, such as allowing mismatches in substrings [Leslie et al., 2004], although this often comes with an increased computational cost and does not reduce the number of parameters.

An alternative approach which does a superior job at grouping similar motifs together is the position weight matrix (PWM) [Staden, 1984, Claverie and Audic, 1996]. The PWM as originally introduced, is calculated using a set of aligned sequences $S = [s_1, s_2, ...s_N]$. We denote by C_{ij} the count of sequences that at the ith position have base indexed by j for $j \in [1, 2, 3, 4]$. The PWM w_{ij} is calculated as

$$w_{ij} = \log\left(\frac{C_{ij}}{\sum_j C_{ij}}\right) \tag{1.3}$$

For a genomic sequence s_i the same length as the training sequences, one can then calculate a similarity score ss

$$ss = \sum_i w_{i,k(i)} \tag{1.4}$$

where $k(i)$ is the index of the ith base in the sequence. As w_{ij} is defined as the log probability at each base, one can interpret this similarity score as the log probability of the sequence s_i being sampled from the aligned sequences. One can then conceive a classifier that considers a sequence part of the aligned set if this similarity score is greater than some threshold. This approach again has been used extensively for tasks such as predicting pol II promoters and other features of genes [Prestridge, 1995, Burge and Karlin, 1997], but presents the issue of requiring the practitioner to know a substantial amount about the problem in order to generate these matrices, though attempts have been made to improve PWM weights initially calculated from aligned sequences to empirically maximize area under a receiver operator curve (AROC) [Li et al., 2007]. It is also not clear how to handle non-linear interactions between motifs that are both relevant in determining the purpose of the sequence being analysed.

1.2 Artificial Neural Networks

Artificial neural networks are a class of algorithms capable of approximating arbitrary functions [Hornik et al., 1989]. They were originally developed in 1958 [Rosenblatt, 1958] as an attempt to create a model of way humans reason about the world, but have undergone substantial evolution in the last 10 years, substantially increasing their utility. Here, we briefly discuss the simple feedforward ANN and the types of approaches that have become common in the literature for our task of predicting biological phenotype from genomic sequence.

1.2.1 The Dense Feedforward ANN

The classical neural network is the Dense Feedforward ANN (also called the multilayer perceptron, or MLP). This model is specified for an input vector x^0, and a target

vector (the truth) y in a hierarchical fashion. For a user-specified K:

$$x_i^k = f^k(\sum_j w_{ij}^k x_j^{k-1} + b^k) \qquad k = 1, 2....K-1$$

$$\hat{y} = f^K(\sum_j w_{ij}^K x_j^{K-1} + b^K)$$

Here, $\hat{y}$ is the output vector, that would ideally equal the target vector y for each input x^0 in the case of a well-trained model. The vectors $x^1, x^2....x^{K-1}$ are referred to as hidden layers, and the vectors b^i are denoted bias neurons which are simply trainable constants. In subsequent sections, these are ignored in terms of the discussion, but they are always present. Similarly, the tensors W^i are trainable constants, that determine the behaviour of the model. The functions f^i are termed activation functions and are critical for the utility of this model. They are applied elementwise to the vector calculated for each k. The choice of each activation function is user-specified, but at least one must be non-linear to permit the network to model behaviour in the target vector that varies non-linearly in the input vector [Hornik et al., 1989]. Without this activation function, this behaviour is never possible, regardless of the number of hidden layers.

Though this specifies a complete ANN, models of this form are not common in the literature today and have several disadvantages. Instead, modifications to this framework have become more common, in a field that has become known as 'Deep Learning', which we briefly discussed below.

1.3 Deep Learning

The initial iteration of the artificial neural network took unstructured data, possibly a feature vector curated by a human, and attempted to associate it with an output. For the simple case of a genomic sequence classification, as discussed above, this might simply be the output of a spectrum kernel or the maximum similarity score for a series of PWMs. Without knowing anything more about this approach, it is clear

that to use these inputs would cause problems identical to those faced by classical machine learning techniques such as support vector machines, logistic regression or decision trees. The main benefit of modern neural networks is the ability to operate on structured data, in this case, an encoding of the sequence itself, and to learn which features (in this case subsequences or PWMs) are important for the problem, as an alternative to human-designed kernels. This implicit featurization is what is known as 'deep learning'.

The notion of a network learning what features of the data are important for solving a particular problem is meaningless without a definition of a 'problem' and a way of evaluating the quality of a solution. The way this is specified here, as with other machine learning approaches is through a cost (often called loss) function. If the tensor x_{ij}, called the design matrix, denotes the jth feature of the ith example, and the tensor y_{ij} is the jth training feature for the ith output function, then the learning problem is specified through as cost function:

$$C(x, y, W) \tag{1.5}$$

where W represents the network parameters. Training an ANN is finding a value of W that minimizes C for the training data and labels. Crudely, one could then imagine that an unimportant feature (the jth column of the design matrix for examples) would have a low weight associated with it, meaning that it would not impact the cost function calculation. The failure of conventional methods for sequence modelling, and conversely the success of the ANN, is that recent techniques have allowed us to find W's that specify which sequence features are important for the minimization of our chosen cost function to an extent greater than even a practitioner with domain expertise.

1.3.1 The Convolutional Layer

A cornerstone of this automatic featurization in deep learning is the convolutional neural network. This technique was introduced by Yann LeCun [LeCun et al., 1989,

1990] in the field of computer vision to recognise handwritten digits. This is analogously applied to genomic sequences by treating them as images through careful choice of encoding. Specifically, a sequence of length N is converted into an array of shape $[4 \times N]$ by selecting a series of orthogonal vectors in place of the genomic bases. There are infinitely many degenerate representations, but a typical one is to use unit vectors such as:

$$A := [1,0,0,0], C := [0,1,0,0], G := [0,0,1,0], T := [0,0,0,1] \tag{1.6}$$

(and a number of analogous choices). Performing this encoding yields 'binary images' of the form:

$$AC = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \tag{1.7}$$

$$GT = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{1.8}$$

The convolutional network then acts on these images in a way such as to minimize a cost function relevant for the particular problem.

The fundamental component of the convolutional neural network is the convolutiuonal layer, defined in our case as a function F acting on a 2 dimensional tensor X_{ij} of shape $[d_m \times l]$, defined:

$$F(X_{ij}) = f\left[\sum_{m=1}^{d_m}\sum_{n=1}^{f_l} X_{m,j+n-1}W_{mni}\right] \tag{1.9}$$

where f is the user-specified activation function. Here, the parameter l is interpreted biologically as the sequence length. As with the previously discussed ANN, this

function is required to be differentiable almost everywhere, and not linear. Note that due to the sparse choice of basis vectors to encode the sequence, this operation is identical to performing the calculation in equation (1.4), calculated for every position in the input sequence for a number of different PWMs simultaneously. The tensor W_{mni} is colloquially referred to as the 'weight tensor' or the 'filter tensor', and is a user-specified shape, subject to certain constraints. The m dimension (of size d_m) must be the same as the i dimension of the input tensor. For this layer directly applied to genomic sequences, this then has to be 4. The n dimension is the length of the 'PWMs', called filter length f_l, (though the weights in this layer are not strictly PWMs as they do not have a probabilistic interpretation), and the i dimension is the number of 'PWMs', n_f, calculated simultaneously. The shape of the output tensor of this operation is $[n_f \times (l - f_l + 1)]$.

A convolutional network is typically composed of sequential convolutional (and other) layers, with the output tensor of one used as input to a subsequent layer, and each convolutional layer having a unique weight tensor. These tensors are initially drawn from a user-specified distribution (see section 1.3.5) with the values then set during training to minimize the cost function (section 1.3.5). The result of this, currently very opaque, process is a function that maps inputs (in our case 2-dimensional tensors representing genomic sequences) directly to tensors representing our predicted annotation without the need for human intervention to define the important aspects of these sequences.

1.3.2 Pooling Layers

The convolutional layers described above can be applied sequentially, to build richer models capable of modelling non-linear interactions between features. However, there is more to modern convolutional networks than just repeated convolutional layers, and they would be substantially less effective without pooling layers. The issue with just using convolutional layers is that the filter size (in our 1-dimensional case filter length)

is often much much smaller than the input to the network. Whilst this has the nice externality of reducing the number of parameters, it makes the network unable to reason using features that are far away from one another. The receptive field of a network is the size of the area of the input that can impact the output of a layer within the network. For a 1D convolution as defined above, the receptive field size is simply f_l and assuming n identical layers with the same filter length used sequentially, the receptive field is length $(n-1) \times (f_l - 1) + 1$, meaning that systems that require a combination of distal information would require an impractically large number of layers.

A way to overcome this problem is termed 'pooling'. These strategies derived from the approach in [LeCun et al., 2004] where the authors introduced an operation that 'subsampled' the previous layer by computing a weighted sum of the outputs of nearby weights, thus reducing the size of the layer size by a factor of n, with n being the number of values combined in this sum. The approach introduced in [Marc'Aurelio Ranzato et al., 2007], was a modification of this, with instead of the weighted sum calculated, the maximum of N neurons is calculated. For the 1D sequence convolution above, we pool our 2D output from a convolutional layer only along the spatial axis such that for a pool length p_l, we have the above operation given by

$$MP(X)_{ij} = \max_{k \in [1+(j-1)\frac{N}{p_l}, j\frac{N}{p_l}]} (X_{ik}) \tag{1.10}$$

with N the number of columns in X_{ij}. This has become a more common operation than the weighted sum pooling in [LeCun et al., 2004], with empirical evidence of superior performance [Scherer et al., 2010], although we use both approaches in chapter 2. A schematic of a typical convolutional network for genomic sequences is given in figure 1.1, illustratring this composition between pooling layers and convolutional layers.

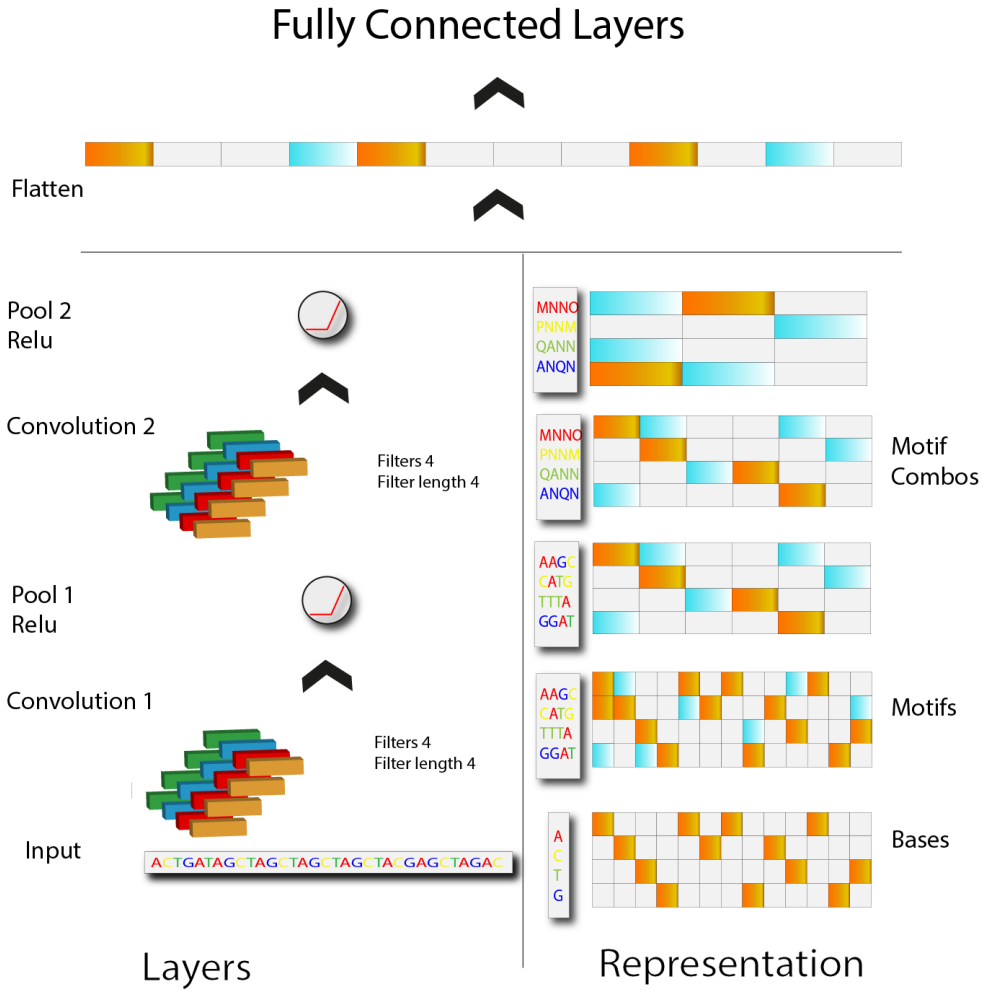

Figure 1.1: A typical convolutional architecture for analysing genomic sequences. The interleaving of convolution and pooling layers yields sequentially more abstract features.

1.3.3 Composition of Sequence Convolutional Neural Networks

There has been a general trend in the machine learning community, and the bioinformatics community as well, for an increasing number of layers in the neural models used for prediction. Initially, convolutional networks were introduced in 2015 for genomic sequence modelling [Alipanahi et al., 2015a], with just 1 convolutional layer, meaning that this was effectively just a self-assembling PWM approach, but the size of these networks, both in terms of layers and number of parameters has rapidly expanded. This trend is illustrated for a small number of the more highly cited literature methods in figure 1.2, though this still lags some way behind the computer vision field, where networks can have over 1000 layers. The reasons for preferring deep networks, that contain many hierarchical layers with a small number of parameters in each, versus shallow networks with a small number of layers but more parameters in each layer is predominantly empirical, with shallow networks memorizing data very well, but failing to generalize as well to unseen data. Work persists on discovering the reasons for this phenomenon, with recent results [Poggio et al., 2017] illustrating that for certain classes of problems deep networks, due to their compositional nature, provide superior theoretical guarantees on their complexity to achieve certain approximations errors for known functions.

1.3.4 Dilated Convolutions

Operating in a similar manor to convolutional layers, dilated convolutions calculate elementwise products of filter weights and the input tensor. The difference is that these dilated layers have a configurable parameter 'dilation rate' that alters which input data is multiplied. For dilation rate r_d, the calculation performed is:

$$F(X_{ij}) = f\left[\sum_{m=1}^{d_m}\sum_{n=1}^{fl} X_{m,j+r_d\times n-1}W_{mni}\right] \tag{1.11}$$

ignoring end effects. This has an impact similar to pooling layers of allowing information that is spatially further apart to be combined by a single convolutional layer.

1.3.5 Advances in Model Optimization

Our ability to train deeper networks has been greatly improved by advances in the way the parameters of these models are trained, in addition to architectural developments. Most modern ANNs are trained using stochastic gradient descent approaches, which when combined with memorization to avoid repeated re-calculation of parameter gradients is termed 'error backpropagation' (introduced independently of ANNs in [Linnainmaa, 1970], and first applied to ANNs in [Werbos, 1982]).

Fitting such large models such that they converge consistently to a good optimum is understandably difficult. Problems that plagued fitting of large, and especially deep networks in the early days were the non-convexity of the cost function with respect to the model parameters, and gradients of this cost function becoming vanishingly small for early layers, leading to inconsequential parameter updates. The latter issue is especially troublesome when fitting networks with a great number of layers.

There has been substantial progress in recent years concerning both of these issues. A large number of techniques have been proposed to aid in the efficient fitting of non-convex functions via gradient descent, typically attempting to calculate adaptive learning rates and momentum terms [Kingma and Ba, 2014, Nesterov, 1983, Tieleman and Hinton, 2012, Duchi et al., 2011] with ADAM being the most widely used of these. Additionally, improved initialization schemes[Mishkin and Matas, 2015, Glorot and Bengio, 2010b, He et al., 2015] combined with novel activation functions (eLU, reLU etc), have substantially improved the issues associated with vanishing gradients .

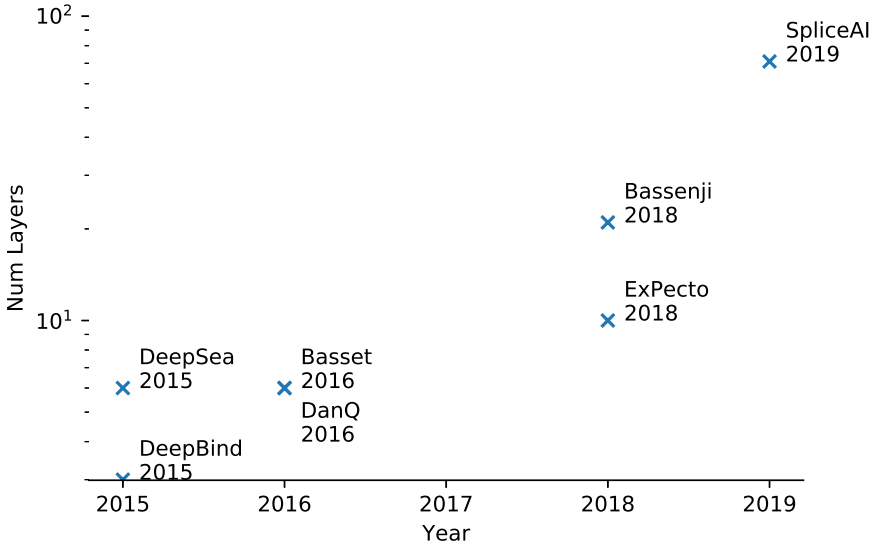

Figure 1.2: Log number of network layers (excluding batch norm, flattening and activation layers) for a number of literature approaches

1.3.6 Choosing Network Topologies

Given an ANN, a practitioner can use the above techniques to attempt to find the values of the weights (use W to denote all weights that belong to the net, probably split over multiple tensors) that minimize the cost function. A related problem is defining the network in the first place. This problem is different from training a given ANN, as the optimal structure of a network is non-differentiable so can not be learned from data in the way that W can.

Currently, the way that most network structures are optimized is by specifying the network in terms of a set of parameters H, known as hyperparameters. The types of hyperparameters specified in H include the number of layers, type of layers, type of activation functions and other features that specify a layer's operation (such as the filter length f_l in a convolutional network). The cost function is non-differentiable with respect to these hyperparameters, and so they must be optimized with other non-gradient methods, presenting a unique set of problems. The main issue with the search for the optimal hyperparameters is that each trial requires training the specified ANN

and measuring the cost function on validation data. This is an expensive operation; typical literature methods can take weeks to train to convergence. Consequently, it is very difficult to explore the space of hyperparameters and find a good structure for the final network. This is a fundamental issue, that does not have an adequate solution, although numerous algorithms have been devised to search this space more efficiently than either choosing random parameter values or cycling through all possible values systematically.

A general framework for this is sequential model-based optimization (SMBO) [Hutter et al., 2011], which defines a surrogate function $f^*(H)$ for the mapping between a network trained with hyperparameters H and the final cost function. This function f^* is typically a model fit from repeated measurements of $f(H), H$, where $f(H)$ is the true cost function value associated with these hyperparameters. One can then use this approximate function to select a new candidate point in hyperparameter space, predicted to yield the biggest improvement in $f(H)$.

A highly used technique to perform this task is HyperOpt [Bergstra et al., 2011], which we use for model hyperparameter selection throughout this work. This approach uses a Kernel Density Estimation approach for f^*. This algorithm is implemented for public use in python, along with a convenience wrapper called Hyperas which is the form of the algorithm that we use most frequently throughout this work.

1.3.7 Recurrent Networks

The networks we have discussed so far have all been feedforward. This means that the graph that defines the forward pass of their computation (computing $f(x)$) is acyclic. This yields stateless networks that will give independent, if not necessarily deterministic, evaluations for subsequent values of x. This is not the only paradigm however, and recurrent networks are one such alternative. Understandably, these have gained substantial popularity in tasks that revolve around sequence modelling, due to their ability to retain state [Greff et al., 2016]. We briefly look at these for

genomic sequence modelling in chapters 2 and 3 in the context of seeing if our proposed methodology improves a literature method, but generally, this has not proved a very widespread technique in this subfield.

1.3.8 Relevant Software

A large reason for the rapid progression in the field is the evolution of a rich software ecosystem. Within the last 5 years, many packages have emerged to make the specification, training and inference with deep ANNs tractable.

Before these general frameworks (Tensorflow, Theano, Torch etc), a practitioner would have to spend a substantial amount of time manually calculating derivatives for novel algorithms, which could then be used with existing optimizers. This is time-consuming and error-prone, substantially reducing the speed of development. These packages on the other hand typically implement automatic differentiation [Baydin et al., 2018] which eliminates the need for tedious calculations of exotic derivatives of novel functions that can comprise networks by treating them as a composition of elementary functions, each of which has a known analytic derivative.

These packages also substantially abstract the notion of hardware from the practitioner, allowing models and algorithms to be specified in an architecture-agnostic form. This allows easy training on state of the art GPUs, as the frameworks handle the calls of the native libraries (such as NVIDIA's CUDA) seamlessly. The models in this work heavily leverage these advancements; We predominantly use a software combination of Keras and TensorFlow for model specification, and an NVIDIA p100 GPU for training.

1.4 Aim: Learn splice Junctions from Sequence

A key component of this thesis is predicting splice junction location from sequence using the techniques discussed in the previous section. Here, we give a brief introduction to the process of mRNA splicing from a biological perspective. This is important, as it informs modelling assumptions. We then review what other cell epigenetics can impact the regulatory mechanisms for this process, and review the extent that they themselves can be predicted from sequence.

Finally, we review the direct computational methods for predicting splice junction locations, discussing what has been achieved so far, and what has yet to be satisfactorily solved.

1.4.1 mRNA splicing, a brief overview

The proteins that govern cellular behaviour are specified by the genome of each cell. This idea is known as 'the central dogma of biochemistry', where, with high fidelity, the exact ordering of the base pairs in the human genome determines the creation of mRNA molecules which are then translated in the cytoplasm into the proteins that ultimately regulate cellular function.

As is always the case in biology, this process has a substantial amount of hidden complexity; The naive model can not fully explain the proteins that we observe in vivo. Specifically, the nascent mRNA molecules undergo a series of tightly regulated maturation phases before they are used in protein translation. A key step in this maturation pathway is mRNA splicing, which will be the focus of this review, and is the process that we later seek to explicitly model. At a high level, this process regulates the removal of stretches of pre-mRNA that will not code for proteins, and are termed introns. This process must occur with basepair resolution, as incorrect splicing locations have serious consequences, such as exon skipping or loss of reading frame, for the resultant proteins.

While there are around 20,000 protein-coding genes in the human genome, we see

substantially more unique protein isoforms than this, and indeed it is estimated that around 90% of genes can yield multiple proteins. Pre-mRNA splicing is a key driver of this variation and is, therefore, a crucial process to study to understand cellular diversity and function.

1.4.2 The Spliceosome

Splicing of nascent mRNA is regulated by a large family of ribonucleoproteins which are collectively called the spliceosome. The major factors in this family are 5 Uracil-rich small nuclear ribonucleoproteins (snRNPs), which are named U1, U2, U4, U5, U6. These proteins are responsible for chaperoning sections of small nuclear RNA (snRNA), with which they share their names, and coordinating the binding of these snRNAs to the nascent mRNA. This set of proteins is called the 'major spliceosome' as it directs intron removal at around 99.5% of introns [Turunen et al., 2013] (which are called U2 type introns). The remaining 0.5% of introns are spliced by a related yet different machinery known as the minor spliceosome. This consists of factors U11, U12, U4atac and U6atac and the U5 common to the major spliceosome. Introns regulated by the minor spliceosome are therefore known as U12 introns.

The snRNAs contained within these snRNPs are vitally important, as they allow the binding to mRNA in a sequence-specific manner, and thus direct a very strictly controlled process. Given the differing machinery used in the major and minor spliceosomes, the obvious question is to what extent the sequence compositions differ for U2 and U12 introns, and how this directs splicing. It turns out that there are substantial differences in the intronic sequence in U2 junctions compared to U12 junctions. U2 introns typically have the well known GT-AG pairing between 5' and 3' splice sites, whereas U12 permit a pairing of AT-AC. There are additional sequence composition differences within the intron; The branch point (the site of U2 or U12 binding) is substantially different between the two spliceosomes, with a number of orthogonal sequence features required for a successful splicing reaction. This characterization of

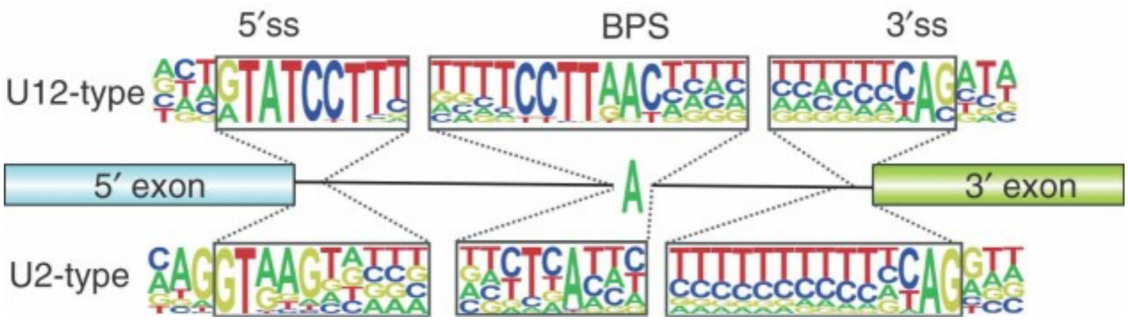

Figure 1.3: Sequence logos around 5' splice site, 3' splice site and branch point differ for U2 and U12 introns (from [Turunen et al., 2013])

the splicing pathway is crucial for our attempts to build sophisticated models for predicting it, as one could imagine they could then learn the complex interplay between particular sequence composition and other splicing factors. The sequence logos for the splice sites and branch points for U2 and U12 introns are presented in figure 1.3.

The spliceosome is assembled on the mRNA sequentially in a multi-stage process:

- **E phase (Early)** snRNP U1 binds to the 5' splice site, along with branch point recognition factors
- **A phase** U2 snRNP binds to the branch point sequence
- **B phase** A complex of U4, U5 and U6 is recruited to the site
- **B$^{\text{act}}$ phase** This complex matures in to the activated complex, with the release of U1 and U4
- **B* phase** Prp2 triggers conversion to B*, which exposes the base A at the branch point
- **C phase** U5 complex binds the 5' exon, creating an intron lariat.
- **C* phase** Prp16 catalyses conformational changes that change the active site to the 3' exon. This is typically the first AG downstream of the branch point, although this is not always the case.

- **P phase** The OH from the 5' exon attacks into the active site
- **Release phase** Prp43 catalyses disassembly of the spliceosome from the sequence and the intron is released.

This process is represented schematically in figure 1.4

1.4.3 Alternative Splicing

Despite being tightly regulated by the spliceosome, the locations of splice sites can vary between individuals and cell lines. This diversity is called alternative splicing. Alternative splicing can cause numerous changes to the resultant mRNA isoform, with by far the most common change the 'cassette exon', which has been seen to account for approximately 50 - 60% of all alternative splicing events [Dvinge and Bradley, 2015]. This form of splicing is characterised by a single exon either being optionally included or omitted, with no other changes to the local isoform. Cassette exons originate from several evolutionary processes. It is known that 'exonization' of intronic sequence and exon shuffling[Gilbert, 1978], whereby existing exons are inserted or duplicated, can lead to the appearance of optionally included exons in pre-existing genes, but it is also the case that constitutive exons can also start to be sub-optimally included due to a slow relaxation in the optimality of the 5' splice site [Lev-Maor et al., 2007]. These mechanisms suggest that cassette exon alternative splicing is a highly important mechanism by which mammals can efficiently explore the proteomic fitness landscape, and indeed the intronic sequence surrounding them displays differing conservation patterns to other introns [Sugnet et al., 2006], indicating cellular machinery exists to regulate their percentage inclusion.

Another, slightly less common, form of alternative splicing modifies the position of either the upstream acceptor (5' ss) or downstream donor(3' ss) sites whilst retaining the bulk of the exon. Interestingly, the rates of these events are different, with alternative donor sites comprising 18% of splicing events and alternative acceptor sites comprising around 8% [Ast, 2004]. Similar to cassette exons, there are sequence

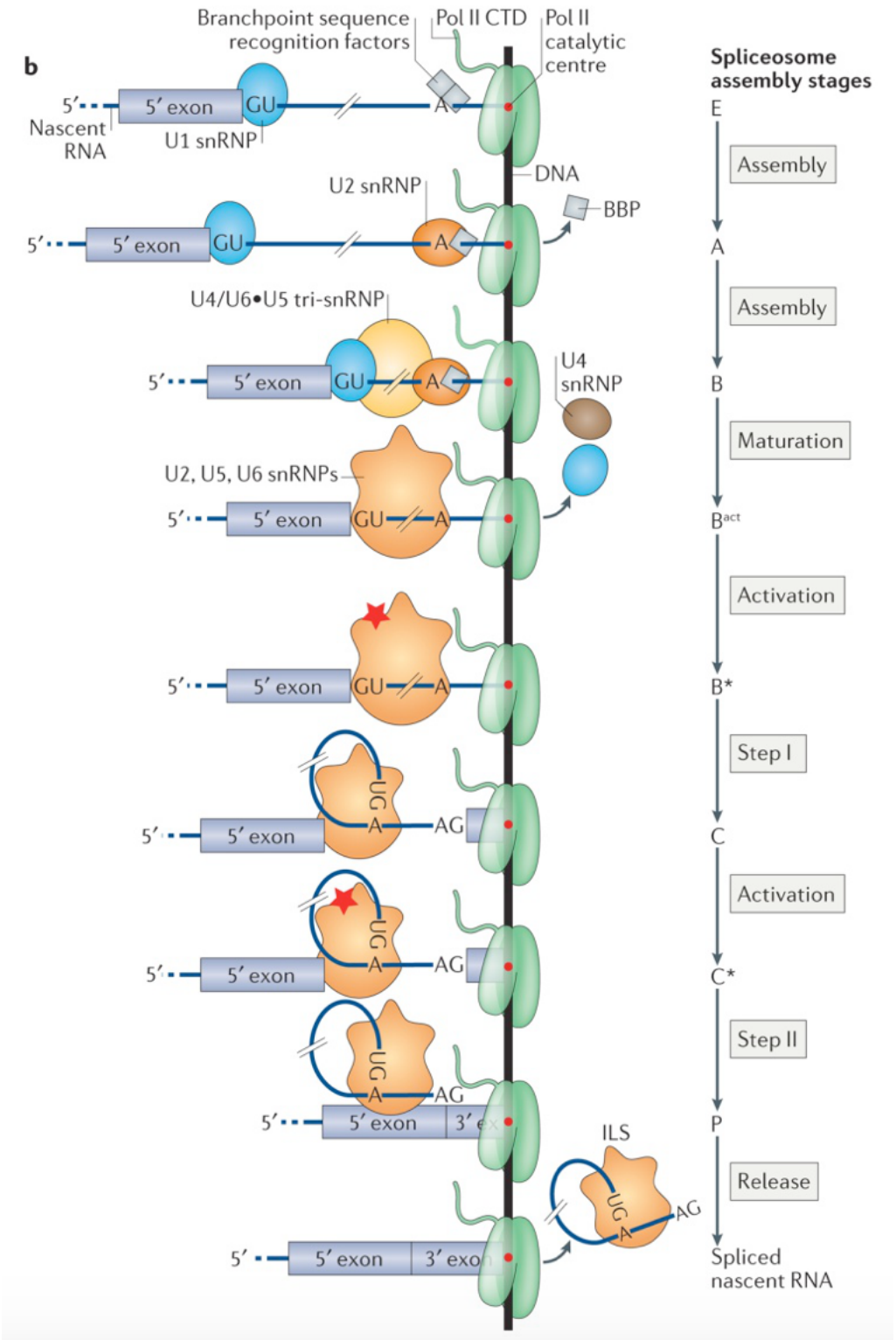

Figure 1.4: Spliceosome assembly and maturation (from [Herzel et al., 2017])

differences in the introns surrounding exons that permit these splicing events. These sequence composition differences are not bilateral, and only in the intron closest to the variable site [Sorek and Ast, 2003].

Further classes of alternative splicing events exist but are much less prevalent. Intron retention is estimated to account for around 3% [Ast, 2004] of events. In this form of alternative splicing, as the name suggests, a typically removed intron is included in the final isoform. Introns that are retained in humans are typically shorter than average and have a high GC content [Galante et al., 2004, Wang and Burge, 2008]. Although most retained intron events are removed via nonsense-mediated decay, these transcripts have been observed to serve important biological functions. Retained intron events, for example, appear to regulate exon definition and impact on local exon skipping. It was observed in [Cho et al., 2014] that in CD45 in lymphocytes, intron retention reduces the expression of particular exons. Additionally, intron retaining transcripts have been observed to have tissue-specific effects [Buckley et al., 2011].

The remaining categories of alternative splicing are more complex, and in contrast to events above, generate isoforms that are non-trivially related. A key class of these events are the mutually exclusive exons, but the behaviour exhibited can be more complicated still.

A schematic outline of the various types of alternative splicing events is shown in figure 1.5.

1.4.4 Further Pathways for Splicing Regulation

Here, we briefly discuss the observed cellular mechanisms that have been shown to impact alternative splicing regulation. To discuss these coherently, we first describe the dynamics of splicing regulation, which provide the motivation for the two principal models for splicing regulation: the kinetic and recruitment models.

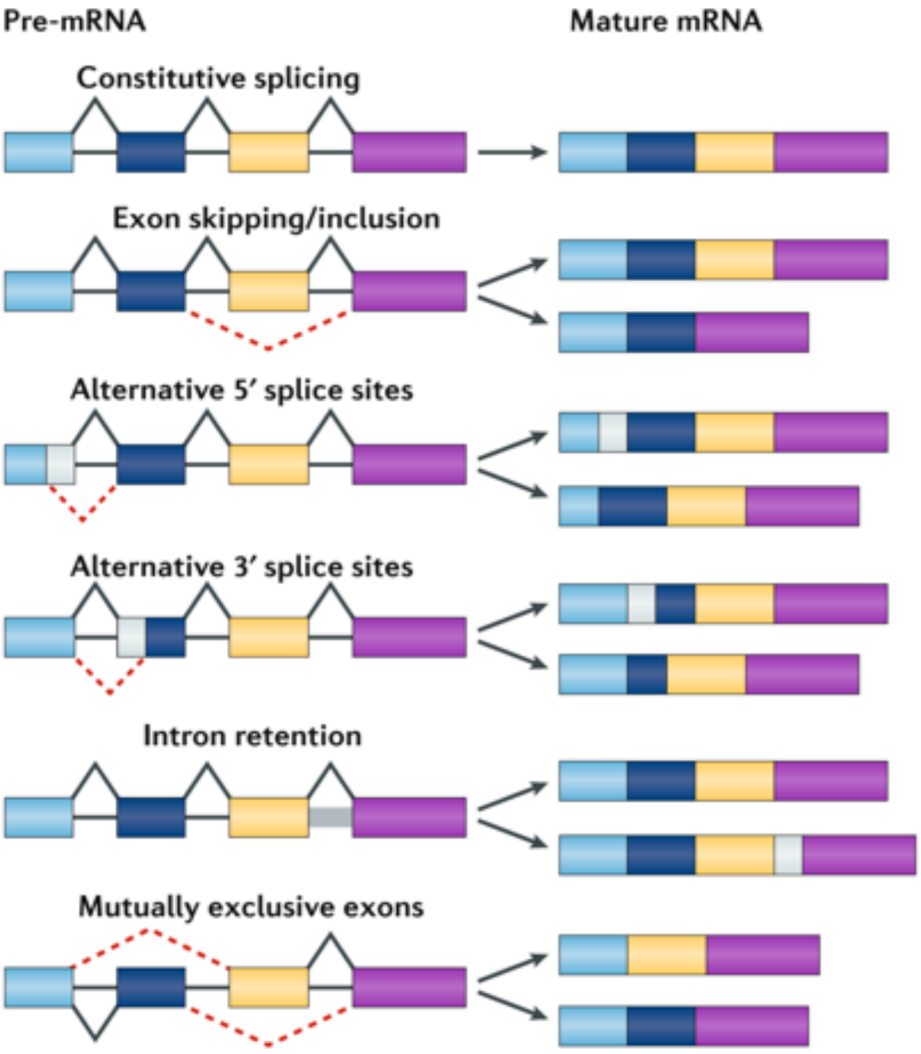

Figure 1.5: The different types of alternative splicing (from Frankiw et al. [2019])

1.4.4.1 Splicing is at Least Partially Cotranscriptional

To appreciate the regulatory mechanisms of splicing, one must understand that it has been observed to be a largely cotranscriptional process. As early as 1988 [Beyer and Osheim, 1988] observed that parts of the spliceosome assembled on nascent mRNA transcripts in Drosophila, after previously noting intron lariats on nascent mRNA [Beyer et al., 1981]. It is important to stress that the splicing may not be completed before transcription is complete, as this process appears to be governed by the kinetics of transcription and splicing, but spliceosome formation on the mRNA could certainly have completed at this point. This is seen as a 'commitment to splice', and in fact from complex 'A' in the spliceosome formation, the potential splicing locations are determined [Lim and Hertel, 2004]. It is perfectly possible to end up with the situation that transcription has fully completed, and the nascent mRNA exists with several putative splice junction locations bound by spliceosome complexes. At this point, the order of intron removal is not necessarily sequential along the genome [Attanasio et al., 2003, Baurén and Wieslander, 1994, Kessler et al., 1993]. Indeed, it appears that altering the order of intron removal seems to impact which introns are included in the final mRNA, [Attanasio et al., 2003, Takahara et al., 2002, Schwarze et al., 1999] indicating interation between spliceosome complexes across exons.

1.4.4.2 The Recruitment Model for Alternative Splicing Regulation

The above observations do not require the coupling of transcription and splicing reactions, and could conceivably be due to a mismatch in the relative rates of these processes; One could expect similar results if the splicing reaction were substantially faster than transcription, making it statistically inconceivable that splicing wouldn't have at least partially started by the time transcription has completed. However, this doesn't appear to be the case, and in fact, there is substantial evidence to suggest that these processes interact through the CTD(C Terminal Domain, a long tail of repeating amino acids) of RNA Pol II [Misteli and Spector, 1999, Hirose et al., 1999], which

appears to assist the coordination of splicing factors on the pre-mRNA. Expanding on this, it has been shown [Das et al., 2007] that there is indeed interaction between the U1 snRNP, which is required early in spliceosome formation which assembles at the donor site, and Pol II. It is further suggested that this mechanism allows U1 to be attracted to the nascent mRNA and therefore promote more efficient spliceosome assembly. Further, there is evidence that acceptor site splice factor U2AF65 interacts with the CTD of pol II [David et al., 2011]. Also, the pol II CTD has been implicated in attracting splice inhibitors [de la Mata and Kornblihtt, 2006], so could modulate splicing through more complex pathways. This general idea that splicing factors are attracted to either pol II or salient chromatin features is known as the recruitment model.

1.4.4.3 The Kinetic Model for Alternative Splicing Regulation

The above evidence for the coupling of transcription and splicing provokes the idea that the rate of Pol II elongation is likely to impact splice site choice at alternatively spliced junctions. We know that Pol II elongation is a dynamic and highly regulated process [Sims et al., 2004], with multiple pathways upstream of this. The idea that the rate of pol II elongation impacts alternative splicing location is known as the kinetic model. An early illustration of this was the behaviour of human fibronectin exon E33, where it was demonstrated that slowing the rate of elongation down favours E33 inclusion [Kadener et al., 2001, Nogués et al., 2002]. The hypothesis is that this mechanism in conjunction with the recruitment model above allows more time for the acceptor complexes to assemble at the 5' end of E33 and thus lead to its inclusion. This was further supported in [Nogués et al., 2002] when transcriptional activation by elongation promoting factor VP16 reduced E33 inclusion. These effects are summarized as the 'window of opportunity' model, where there is a window of opportunity in the recruitment of the spliceosome.

It is important to stress that it is not a definite rule that slow elongation always favours increased exon inclusion, and there have been opposite observed behaviours

for similar elongation rate changes [Dutertre et al., 2010, Ip et al., 2011, Muñoz et al., 2009]. Specifically, Dujardin et al. [2014] show that CFTR alternative exon 9 (E9) is preferentially skipped in the context of slow elongation rate. The mechanism for this is the increased recruitment of the negative splice factor ETR-3 onto the 3' splice site, illustrating the complexity of this process.

1.4.4.4 Chromatin Structure and alternative splicing regulation

The above observations emphasise that the regulation of alternative splicing is a multifaceted process that can not be explained by a simple model. They also suggest a mechanism by which chromatin structure can influence splicing. One could imagine, under the recruitment model, specific splicing factors being attracted to elements in the chromatin. Under the kinetic model, different chromatin configurations could impact elongation speed and thus affect the pathways downstream of this.

Previous studies into Pol II elongation rate have already implicitly shown that chromatin state can impact splicing through the kinetic model by regulating its speed. Specifically, in [Kadener et al., 2001, Nogués et al., 2002], the chromatin-modifying complex TSA was used to indirectly study splicing by increasing elongation rate[Greer et al., 2015] by increasing histone acetylation.

Chromatin structure has also been shown to act more directly in a manner orthogonal to elongation control through the direct interaction with the snRNPs. Chromatin remodelling agents also appear to interact with U2 [Martinez et al., 2001] and U1 [Cheng et al., 2007], which are crucial in early-stage spliceosome formation.

In addition to the above observations, histone modifications have been implicated as a major factor in alternative splicing regulation. Even after the effects of nucleosome positioning, certain histone marks such as H3K36me3, H3K4me3 and H3K27me2 are enriched at exon-intron boundaries[Dhami et al., 2010, Spies et al., 2009]. One could imagine these histone marks operating under either the recruitment or kinetic model to regulate splicing, and there is substantial evidence for both.

Firstly, both H3K9me3 and H3K9me2 have been previously correlated with tran-

scriptional repression and reduced pol II elongation speed. In the CD44 gene, it was demonstrated that H3K9me3 marks correlate with increased inclusion of alternative exons in the variable region [Vakoc et al., 2005]. This paper shows that H3K9me3 here interacts with heterochromatin protein HP1γ, which is the cause of this slowdown by reducing the transcriptional activity in this region. H3K9me2 has been shown to cause similar retention of alternative exon EDI in the FN1 gene[Alló et al., 2009]. Again, this mechanism is interaction with a heterochromatin modifying protein HP1α and reduced elongation speed. These are by no means uncommon pathways, and there have been substantially more histone-splicing interactions noted, with H3K9ac, H3ac, H4ac and H3K36me3 all being observed to have a role in splice regulation [Zhou et al., 2013].

Histone modification has also been observed as a real driver of tissue-specific splicing. By studying the tissue heteromorphic gene FGFR2, [Luco et al., 2010] looked at the inclusion of two mutually exclusive exons to understand how the vastly different inclusion ratios were related to histone modification in epithelial and mesenchymal cells. Specifically, there is a significant differential histone concentration at the alternatively spliced region between the samples, with H3K36me3 and H3K4me1 highly prevalent in mesenchymal cells vs high concentrations of H3K27me3 and H3K4me3 in epithelial. Further, the authors demonstrated a degree of causality in this relationship, by specifically modulating histone marks and observing that this changed the splicing patterns observed.

1.4.4.5 Exon Recognition

Additionally, large scale chromatin mapping indicates increased nucleosome occupancy (meaning the fraction of cells in bulk tissue that have a nucleosome at a specified position) around exons [Andersson et al., 2009, Chodavarapu et al., 2010], and has shown that changes in these nucleosome configurations can impact the splicing at the relevant junctions [Iannone et al., 2015]. Further, this phenomenon appears to regulate splicing in a greater way than simply because nucleosome positioning im-

pacts pol II pausing locations [Churchman and Weissman, 2011] or because contained histones attracting splice factors under the recruitment model.

This general idea of marking exon location is known in the literature as 'exon definition' and is an important factor in splicing regulation[Berget, 1995]. This observation goes some way to explain the apparent missing information looking only at canonical splice sites and branch points. Specifically, the observed phenomenon is that to make a good splice location one must have a 5' acceptor motif paired with a 3' donor motif (fig 1.3) paired across a section of sequence that has 'exonic' sequence context and epigenetic features. This synergy appears to be caused by an interaction between the downstream U1 and the upstream U2 elements that appears to stabilize the early spliceosome complex [Moldón and Query, 2010].

There has been substantial work to define these features that are necessary for this 'exon definition' process, revealing several trends. Indeed, further to simple nucleosome enrichment at exons, it was shown in [Schwartz et al., 2009] that nucleosome enrichment was positively correlated with exon inclusion ratio, with the highest mean nucleosome occupancy at constitutive exons, and progressively lower at more alternatively spliced exons. Additionally, specific histone modifications appeared to mark exons, as opposed to just nucleosome positioning. H3K36me3 was by far the strongest signal, but monomethylated H3K79, H4K20 and H2BK5 were also observed to be enriched. In terms of related sequence features that are important to this process, the authors showed that GC content was correlated with this nucleosome occupancy. Additionally, the authors provided further evidence that this increased nucleosome concentration acts as an RNA Pol II 'speed bump', suggesting that this exon definition interacts synergistically with the kinetic model of splicing to favour exon inclusion.

1.4.4.6 Splicing Enhancers and Silencers

Another important pathway of splicing regulation is that of splice enhancers and silencers which, at a high level, appear to act through the recruitment model to either

attract or repel splicing factors. The genomic sequence within genes contains small enhancer or silencer motifs have been shown to bind uniquely to SR proteins, characterised by an RNA recognition motif (RRM) and an RS domain for protein-protein interactions, [Liu et al., 1998] that then interact either synergistically or antagonistically with members of the spliceosome. For example, an implicated pathway is the recruitment of sequence recognition factor U2AF35 by exonic splicing enhancers [GRAVELEY et al., 2001]. Indeed, exonic splice enhancers are thought to play a role in facilitating the exon definition complex discussed above[Moldón and Query, 2010].

These motifs can provide additional context to discriminate between equally plausible acceptor or donor sites, and mutations that would otherwise be silent have been shown to impact this pathway to alter spliceosome recruitment and change splicing outcomes [McVety et al., 2006]. They have been shown computationally to be enriched around exon ends, and appear to be somewhat conserved [Fairbrother et al., 2004], indicating the importance of their function in determining certain splicing locations. Specifically, this mechanism appears to be most important in deciding, one way or the other, whether locations with somewhat weak canonical splice motifs [Cáceres and Hurst, 2013] become exons.

1.4.4.7 RNA Secondary Structure Affects Splice Site Accessibility

An orthogonal yet important mechanism in regulating alternative splicing appears to be the modulation of mRNA structure. As we have already discussed, splicing is co-transcriptional, meaning that full folding may not occur before splicing is completed. What this means is that the impact of structure could plausibly also be moderated by the speed of elongation. A hypothesized way in which mRNA structure could impact splicing is by making certain binding motifs less accessible to splicing factors [Warf and Berglund, 2010]. This effect has been concretely observed, with structures around the 5' or branch point being shown to impact the recruitment of U1 and U2 [Goguel et al., 1993, Halfter and Gallwitz, 1988, Solnick and Lee, 1987, Eperon et al., 1988] respectively and thus impinging on spliceosome formation. This has been more

directly explained as U2AF65, which is involved in an earlier step of spliceosome formation and ultimately aids in U2 recruitment, better recognising single-stranded binding sites [Sickmier et al., 2006] and thus structure here can block binding and consequent splicing [Warf et al., 2009].

Conversely, mRNA structure can facilitate splicing by shortening the distance between important splicing factors, which can aid the maturation of the spliceosome. A key property of introns is that they can be very long, and it is not uncommon to see lengths of up to 0.5mb. This could certainly be problematic for efficient splicing, as the spliceosome initially forms at either end and needs to come together to initiate catalysis. Long-range interactions have been observed in mRNA structure that appear to bring together the 5' and 3' sites so this process can occur [Muh et al., 2002].

1.4.4.8 RNA Secondary Structure Regulation

Given RNA secondary structure can regulate splicing, it seems natural to ask what regulates this secondary structure, to understand the extent these pathways are likely to be predictable from genomic sequence. Clearly, as RNA folding takes time, the kinetics of this reaction are important, but there are also examples where proteins expressed endogenously are involved in causing structural changes in the mRNA [Lamichhane et al., 2010]. A common theme is the 'looping out' of a cassette exon, where an RNA-binding protein causes splicing to become unviable due to difficulty in the spliceosome interacting across an exon, which prevents the exon definition complex from forming. For example, PTB protein was shown to prevent the interaction between splice sites of neighbouring exons and the cassette exon c-src N1 leading to its repression [Sharma et al., 2008].

1.4.5 The Predictability of Splicing from Sequence

So far, we have discussed several mechanisms that are likely to impact the splicing we observe in vivo. A complete model for splicing regulation should be able, albeit

possibly implicitly, to take these into account when generating predictions of splice site location. Here, we briefly outline the extent to which each of these factors has been shown to be predictable from genomic sequence, and discuss the likely importance of each factor in the success of any final model.

The results of this analysis are tabulated in figure 1.4.5 for the features spliceosome binding, nucleosome positioning, splice enhancer and silencer binding, histone state, RNA secondary structure and differential promoter usage. These results tell an interesting story about our attempts to model splicing regulation, with the trend being that the seemingly more important factors such as learning branch point and splice site motifs have been addressed more completely by in silico tools. This could be a pragmatic response from the community to attempt to pick the low hanging fruit or a result of the fact that we simply didn't know about these more subtle effects until recently.

Neither of these explanations appears to fully explain the situation though. The problems that have less complete solutions appear to be fundamentally more challenging, and not simply neglected by the community. The prediction of Pol II elongation speed change due to differential enhancer-binding, or nascent mRNA folding reducing site accessibility are both likely to require sophisticated simulation for reasonable results. On the other hand, PWM approaches have proven satisfactory for putative splice site identification (at least from a sensitivity viewpoint). We, therefore, appear to be in a situation of marginal gains: We can narrow down tens of thousands of base pairs of putative splicing locations to perhaps a set of tens of reasonable options using simple methods, but to further narrow these down to perhaps 1 or 2 locations that are actually splice-sites requires hugely more sophisticated approaches. Chapter 3 of this thesis is just one example of the application of the techniques discussed previously, and built upon in chapters 1 and 2, to this problem. The hope is that the modern machine learning techniques will be able to featurize the sequence directly, and implicitly learn some of these more sophisticated aspects of the human splicing code.

Splicing Regulator	Relative importance	Literature Predictability
Spliceosome Binding	High- Necessary condition for all junctions	High - There have been well defined models for splice sites, and branch points for a long time using classical methods e.g. [Reese et al., 1997, Lim and Burge, 2001]
Nucleosome Positioning	Medium High - Exon recognition is necessary to discriminate between multiple 'ideal' splice locations	Medium High - Shown to correlate with GC content. Classical models achieve good performance, ([Awazu, 2016] has 0.9 AUROC)) and implicit learning of nucleosome occupancy seen in deep learning models such as [Jaganathan et al., 2019].
Splice Enhancer / Silencer Binding	Medium - Can be very important at determining splice site utilization for cases where the canonical motif is weak	Medium Low - Tools exist to predict these, such as ESEFinder [Cartegni et al., 2003], but this is a challenging problem, as evidenced by [Cáceres and Hurst, 2013], which meta-analysed 4 sets of exonic predicted splice enhancers and found only a 1% overlap between them.
Histone State	Medium Low - Small number of examples that show explicit modification of histone leads to altered splicing (such as [Luco et al., 2010])	Medium - A number of recent deep learning approaches but not a solved problem. In [Zhou and Troyanskaya, 2015b] 104 histone modifications are predicted in various cell lines, albeit with substantially lower accuracies than DNase or tf binding sites.
RNA secondary structure	Unclear	Low - Tools exist for predicting catalytic activity of small RNA by predicting their structure such as [Denman, 1993], but many have sequence length limits, and it is currently not possible to predict how nascent RNA will fold in a reliable way that would facilitate splicing prediction.
Differential Promoter / Enhancer Usage	Unclear, though studies such as [Kadener et al., 2001, Nogués et al., 2002] indicate that this could be important under the kinetic model by changing elongation speed	Unclear how methods to predict promoter usage could be used in relation to polII elongation speed.

1.5 Computational Approaches for Splice Site Prediction

In addition to understanding the extent to which the constituent pieces of the splicing process are likely to be able to be modelled from genomic sequence, it is important to understand the work that has been undertaken over the years to modelling this process directly. This allows us to identify improvements over literature methods, and place our work within the context of literature. To this end, we give a brief review of some of the more highly cited literature approaches and give a discussion of their relative strengths and weaknesses.

1.5.0.1 Classical Models

A very common computational approach to splice site prediction is to simply examine small local sequences and determine how likely they are to be either donor or acceptor splice sites. Perhaps the most well-known example of this approach is MaxEntScan [Yeo and Burge, 2004]. This approach provides two models, one for the donor splice site (which takes a 9bp sequence) and one for the acceptor site (which takes a 23bp sequence). MaxEntScan then defines a probabilistic model to predict, for both of these cases, the probability that the input sequence is a splice site $p^+(x)$ and the probability that it is not $p^-(x)$, returning a log ratio of these probabilities. One can then evaluate all positions in the genome for splice site predictions, computational power permitting. Philosophically similar but methodologically different approaches are the 'Analyzer-Splice-Tool' (http://ast.bioinfo.tau.ac.il/SpliceSiteFrame.htm), HBond [Freund et al., 2003] (only for donor sites) and NNsplice which uses a simple MLP to predict whether sequences are donors or acceptors [Reese et al., 1997]. Similar approaches exist for predicting branch points (U2 binding sites) [Corvelo et al., 2010], as these can be difficult to predict simultaneously with the acceptor junction as their relative position is variable. The trouble with these approaches is that they do not usually take into account large-scale sequence context; They rely on individual models for donor and

acceptor sites. As discussed previously, this is insufficient to predict splicing as there are many more splice site motifs than are used by the spliceosome.

A further weakness with these approaches is that they aren't prescriptive of global splicing behaviour. For example, if a splice site is destroyed or created, how is the isoform structure likely to change? To address this issue, a different class of algorithm is necessary. GENSCAN [Burge and Karlin, 1997] uses an HMM (Hidden Markov Model) in conjunction with these local sequence feature models to derive predictions that are consistent with genome architecture. This method is capable of taking sequences up to 1Mbp, and provide predictions of the genomic context of each base (intronic, exonic, etc). An alternative and more recent approach CRYP-SKIP [Divina et al., 2009] accepts a sequence of up to 4kb containing 1 exon and attempts to predict using sequence features whether splicing mutations are likely to cause exon skipping as opposed to alternative acceptor site usage.

1.5.0.2 Learning the 'Splicing Code'

More recently, a series of computational methods have been devised to learn the 'splicing code'. This high-level approach focuses on predicting cassette exon inclusion ratios as opposed to predicting the specific donor or acceptor junctions. A common methodology here has been to use manually derived sequence features calculated from a window around the exon in question (somewhat similar to a spectrum kernel) as input to increasingly sophisticated models [Xiong et al., 2015, Barash et al., 2010, Leung et al., 2014]. The most recent of these[Xiong et al., 2015] showed that the learned signals had a strong overlap with known binding sites for RNA binding proteins, and found a large number of predicted splice altering polymorphisms outside of the typical splice junction dinucleotides. This indicates a reasonable ability to model the splice regulatory machinery, although the model was only constructed for 10,689 exons that were known to be partially included. Further, this approach only encompasses a subset of alternative splicing events.

1.5.0.3 Deep Learning Approaches

More recent approaches have begun to use more sophisticated methods from the deep learning literature to model a greater number of splicing phenomenon simultaneously. 'COSSMO' [Bretschneider et al., 2018] attempts to predict the location of the next acceptor site conditional on donor location (and the previous donor conditional on the acceptor) for a subset of possible candidates using a recurrent neural network, without making a distinction between different categories of alternative splicing events, at this task it achieves around a 70% accuracy. SpliceAI [Jaganathan et al., 2019], uses a deep convolutional network to predict all possible splice donor and acceptor junctions within a 5kb window, though does not attempt to model inclusion ratios. The authors show impressive sensitivity of this model to deep intronic mutations, providing valuable insight for clinicians seeking to interpret otherwise silent mutations.

1.6 Splicing and disease

As a final motivating statement of this thesis, we perform a very brief review of the pathways by which splice altering mutations have been shown to impact human disease. This is by no means a complete review, as this would be beyond the scope of this work but serves to provide a context to constructing computational models of splicing, beyond esoteric interest. A notable absence in the work below is a discussion of cryptic splice sites, where mutations can trigger usage of a previously unused deep intronic splicing location. These are of great interest to clinicians, as they are currently difficult to interpret and appear to be important in certain disease pathways. Examples of these are discussed in chapter 3, in the context of our models.

1.6.0.1 Consensus sequence altering mutations

There are numerous documented examples of aberrant splicing causing disease phenotypes, and also a number of pathways by which this can occur. A clear class of

splicing altering polymorphism is one that changes in some way either the 5' or 3' consensus sequence, or other splice silencers of enhancers. As outlined in [Scotti and Swanson, 2016], this is a common, and often readily interpretable, mechanism with many literature examples. For instance, the LMNA gene, when spliced incorrectly, has been implicated in both familial partial lipodystrophy [Morel et al., 2006] and limb girdle muscular dystrophy [Muchir et al., 2000] due to donor splice site mutations before exons 8 and 9 respectively, in addition to other consensus sequence effects [Eriksson et al., 2003, Otomo et al., 2005].

1.6.1 Spliceosome Recruitment

Another important pathway that has been shown to pathologically affect splicing is the hindrance of spliceosome formation. An example of this is a mutation `c.2204+6T>C` in IKBKAP impeding the ability of U1 to be recruited and thus yielding a skipped exon 20 in the gene. This has been shown to cause familial dysautonomia [Ibrahim et al., 2007].

1.6.2 Other

In addition to the above, there are also mutations that damage the structure of the spliceosome globally. In [Jafarifar et al., 2014], it was noted that mutations in the RNU4ATAC gene, caused splicing difficulties in junctions spliced by the minor splicesosome. Additionally, [Tanackovic et al., 2011] observed similarly that mutations reduced stability of the major splicesome and thus the fidelity of splicing in retinitis pigmentosa. There are also further, more subtle disease implicated splice-altering pathways. Specifically, faulty splicing has been implicated in certain types of cancer. A tumor supressor gene implicated in gastric cancer CDH1, when faulty, yields around a 50% lifetime probability of developing the disease. Studies indicated that in gastric cancer cell lines, there is a substantial enrichment of transcripts missing the last 83 base pairs of exon 8 [Li et al., 2015], and moreover, this was linked with a loss of

acetylation in H3K16Ac and H4K16Ac local to this exon in the gastric cancer cell lines, suggesting a degree of epigenetic regulation causing disease phenotypes, with a growing body of literature discussing this [Narayanan et al., 2017].

Chapter 2

Tailoring Deep Learning for Genomic Sequences

2.1 Introduction

Neural networks represent incredibly powerful tools in the modelling of complex data due to their ability to represent complex decision functions, coupled with the advent of efficient methods to train them. This allows practitioners who are studying processes which generate large quantities of noisy data for which there is no probabilistic model to generate predictions. The disadvantage of these unstructured models is that they are often highly parameterised and prone to overfitting. In this chapter, we discuss efforts that have been made to regularize neural networks by allowing them to understand the symmetry inherent in the problem domain they are applied to. We then derive the modifications required to make neural networks capable of understanding the reverse complement symmetry that is very common in a large number of biological problems. We go on to discuss briefly the constructions of such networks and finally show the relationship of this approach with classical data augmentation for certain classes of symmetry group.

2.1.1 Group Equivariance

In order to design complex computation ANNs that are capable of interpreting problem specific symmetry, it is important to define the framework by which we specify the meaning of a specific symmetry operation. This is the formalism of an equivariant map.

Specifically, if we have a function f that performs the mapping $f : X \to Y$ for sets X and Y and we have a group $G = \{g_1, g_2, \ldots, g_N\}$. This function is said to be an equivariant map if both X and Y are acted on by G, and each group operation commutes with f i.e.

$$g_i \circ f(x) = f(g_i \circ x) \tag{2.1}$$

where $g_i \in G, x \in X$, with $\circ$ denoting the action of g. One can then imagine that f represents a complex ANN, composed on many layers of sequantial computation. If one could design such a network that exhibited this property with respect to a symmetry group G, then one could specify in a predictable way how the predictions of the network will change given transformations $g_1, g_2 ... g_N$ of the input data. For example, if all group elements act as the identity when acting on the output set Y, then the network is simply invariant under application of each of the symmetry operations, and its output will remain unchanged.

Initially, this idea of group equivariant networks was presented for image convolutional networks and termed GCNNs (group convolutional neural networks) by [Cohen and Welling, 2016a]. In their setting, they derived networks that were equivariant to the symmetry group composed of rotation and flip operations on images. In conjunction with previous architectures, they showed state of the art results of this approach on canonical datasets CIFAR10 and MNIST. Further work in this direction has been recently carried out in the field of computer vision, [Worrall et al., 2017], extended the ideas above using a novel architecture to allow equivariance to continuous rotations using Harmonic Networks. Here, we derive convolutional and recurrant equivariant networks that are applicable to genomic reverse complement symmetry.

2.1.2 Convolutional networks as an implementation of translation equivariance

Before illustrating further modifications to the convolutional layer, it is illustrative to discuss the extend to which the convolutional layer, by design, is group equivariant. Convolutional networks, since they were originally introduced have become ubiquitous in the field of deep learning and computer vision in particular. There are a number of key benefits of the convolutional layer, the first of which is to keep the number of parameters down by using the same parameters repeatedly on a model. Indeed for most of the models used an discussed in this work, the number of parameters is subststantially fewer than the number of outputs in a convolutional layer. In bioinformatics, this can be seen in [Zhou and Troyanskaya, 2015a] which has 5M parameters in its first 3 convolutional layers and 55m in the final fully connected layer, and [Quang and Xie, 2016] that only has 33k parameters in its sole convolutional layer, and 45M parameters in the recurrent layer after that.

The hidden benefit of the convolutional layer is that the structure of the network implicitly enforces equivariance to translation (ignoring edge effects). A 1 dimensional convolutional layer C with a linear activation function acting on a tensor X_{ij} yields output:

$$o_{ij} = C(X_{ij}) = \sum_{m=1}^{W} \sum_{n=1}^{f_l} X_{m,j+n-1} W_{mni} \tag{2.2}$$

. If the input tensor X_{ij} is translated such that $X_{ij} \to X_{i,j-m}$ then it is clear to see that $o_{ij} \to o_{i,j-m}$. This can be expressed as $C(T_m \circ X_{ij}) = T_m \circ C(X_{ij})$, with the T_m operator representing 1 dimensional translation by m units. Note that this is only true for the infinite convolutional network, and as previously noted, edge effects may have a small effect in the practical networks used in computation.

2.1.3 Relation to Literature

Parts of this chapter, have been published in 'An equivariant Bayesian convolutional network predicts recombination hotspots and accurately resolves binding motifs'- Bioinformatics [Brown and Lunter, 2018].

2.2 Equivariant networks for genomic sequence analysis

We use feedforward neural networks to generate a learnable mapping directly from the input sequence to an output response variable, which in the case of chapter 3 is a class assignment. The network is composed of a sequence of layers that define a directed acyclic computation graph, with each layer acting in turn on the output of its ancestor. Explicitly, for a 2 dimensional input tensor X_{ij}, the network can be viewed as a function composition

$$F(X) = (F_n \circ \cdots \circ F_1)(X) = F_n(F_{n-1}(\dots F_1(X)\dots)) \tag{2.3}$$

with component functions F_i representing the actions of layer i; here $\circ$ denotes function composition. Note that component functions are defined on their own tensor spaces, $F_i : T^{(i-1)} \to T^{(i)}$.

In our case, X_{ij} represents a length-N DNA sequence from an alphabet $\{A, C, G, T\}$ which is usually one-hot encoded [Lanchantin et al., 2016, Alipanahi et al., 2015b, Zhou and Troyanskaya, 2015a] so that $T^{(0)} = \mathbb{R}^{4\times N}$. The DNA represented by the input sequences X exists physically mostly in a double-stranded form, with one strand hydrogen bonded to its reverse complement. This means that the sequence seen by the model could just as naturally be represented by its reverse complement, and the network should arrive at identical outputs for these two sequences:

$$F(X) = F(RC(X)) \tag{2.4}$$

where $RC : T^{(0)} \to T^{(0)}$ maps the encoding of a sequence to the encoding of its reverse complement. One way to achieve this symmetry is to require that $F_1(RC(X)) = F_1(X)$. However, this is highly restrictive; the first layer often represents protein binding motifs, which are often not RC symmetric. Intuitively, one wants the output of a layer to exhibit the "equivalent" symmetry appropriate for the encoding of the next layer. The mathematical translation of this is to require equivariance:

$$(F_i \circ RC_{i-1})(X) = (RC_i \circ F_i)(X) \tag{2.5}$$

for all i and all $X \in T^{(i-1)}$. A graphical representation of this relation is given in Figure 2.1. Note that the two operators RC in equation (2.5) are different, as indicated by their index, as they act on different tensor spaces. In particular, in our case RC_n acts as the identity on $T^{(n)}$ to ensure that the full model is invariant under reverse complementing, and on $T^{(0)}$ the operator RC_0 is determined by the chosen encoding. The modeler has freedom in choosing RC_i on intermediate layers, subject to constraint (2.5), which also imposes constraints on the F_i, the initialization, training procedure, and any parameter ties used during training. Note that this setup is not restricted to RC equivariance, and is valid for any group with actions on tensor spaces, including mirror symmetries, rotations and translations. In this general case the operators RC_i in Figure 2.1 are replaced by group actions $J_g^{(i)}$ operating on $T^{(i)}$, where g is a group element. We will not pursue that direction here, but see [Cohen and Welling, 2016b] for an exposition.

$$\begin{array}{ccccccc} T^{(0)} & \xrightarrow{F_1} & T^{(1)} & \xrightarrow{F_2} & \cdots & \xrightarrow{F_n} & T^{(n)} \\ \downarrow{\scriptstyle RC_0} & & \downarrow{\scriptstyle RC_1} & & & & \downarrow{\scriptstyle RC_n} \\ T^{(0)} & \xrightarrow{F_1} & T^{(1)} & \xrightarrow{F_2} & \cdots & \xrightarrow{F_n} & T^{(n)} \end{array}$$

Figure 2.1: Commutative diagram for a reverse complement equivariant network. Compositions of functions along any path in the network that respects the arrow directions depend only on the start and end point, and not on the path taken.

2.2.1 Choice of one-hot basis

We define the vectors

$$A = [1,0,0,0]', C = [0,1,0,0]', G = [0,0,1,0]', T = [0,0,0,1]'$$

where $'$ denotes transposition, and encode a genomic sequence by the concatenation of corresponding column vectors. For a sequence encoded in this way,

$$RC(X)_{ij} = X_{-i,-j} \tag{2.6}$$

with negative indices denoting offsetting from the opposite end of each dimension by that amount. Note that we use 1-based indexing.

2.2.2 Convolutional layer

For sequence classification we use a 1D convolution layer, with n_f filters of length f_l stored in a weight matrix W_{ijk}. The output of such an operation on tensor X_{ij} (ignoring bias terms for simplicity) is

$$C(X)_{ij} = f\left[\sum_{m=1}^{4}\sum_{n=1}^{f_l} X_{m,j+n-1}W_{mni}\right] \tag{2.7}$$

where f is the activation function. Note that we used "valid padding" so the sequence dimension is reduced by $f_l - 1$. Applying the reverse complement operation to the input tensor of shape $[4, N]$ yields

$$C(RC_{l-1}(X))_{ij} = f\left[\sum_{m=1}^{4}\sum_{n=1}^{f_l} X_{-m,-(j+n-1)}W_{mni}\right] \tag{2.8}$$

$$= f\left[\sum_{m'=1}^{4}\sum_{n'=1}^{f_l} X_{m',[-j]+n'-1}W_{-m',-n',i}\right] \tag{2.9}$$

$$= f\left[\sum_{m'=1}^{4}\sum_{n'=1}^{f_l} X_{m',[-j]+n'-1}W_{m',n',-i}\right] \tag{2.10}$$

$$= C(X)_{-i,-j} := RC_l(C(X))_{ij} \tag{2.11}$$

where we used the substitutions $m = -m'$, $n = f_l + 1 - n'$, and $[-j]$ denotes the positive index $(N+1)-(f_l-1)-j$. At (2.10) we assumed that W obeys the symmetry

$W_{m,n,i} = W_{-m,-n,-i}$. Therefore, the convolutional layer satisfies (2.5) if this weight symmetry holds, and if we define RC on the output layer as in (2.11). A schematic of this is shown in figure 2.2A. We note that this specific symmetry was used in a convolutional layer in [Shrikumar et al., 2017a] and was shown to improve inference.

2.2.3 Max-pooling

Spatial Max Pooling along the position dimension of a tensor is used in order to increase the receptive field at the expense of resolution. We consider the special, commonly used, case where the stride length is the same as pool width. For this case we have, for a pool length p_l,

$$MP(X)_{ij} = \max_{k \in [1+(j-1)\frac{N}{p_l}, j\frac{N}{p_l}]} (X_{ik}) \tag{2.12}$$

As long as p_l divides N this defines an equivariant mapping, with RC defined on the next layer in the obvious way.

2.2.4 Reverse complement max pooling

We frequently found it useful to pool along the "filter axis" rather than along the spatial direction. More precisely, we take maxima along orbits (points that are reachable from one another via application of the symmetry operator) under the group action, which in our case consist of two elements. After RC max pooling we therefore have a new output

$$M(x)_{ij} = \max(X_{i,j}, X_{-i,-j}) \tag{2.13}$$

This process halves the size of the output tensor compared to the input. The resulting compression is depicted in figure 2.2B. Equation (2.5) is again satisfied, now with RC acting as the identity on the new layer. A network that contains RC max pooling is therefore automatically symmetric under reverse complementing. In general, more complex symmetry groups may contain nontrivial subgroups, and any of those may be used to do partial symmetric max pooling; see [sec. 6.3] [Cohen and Welling, 2016b].

2.2.5 Dropout

Dropout is a form of stochastic regularization for neural networks designed to prevent over-fitting by adding noise to the output of a layer during network training [Srivastava et al., 2014]. This is commonly implemented by dropping out nodes according to a Bernoulli-distributed random variable, and can be implemented by taking the Hadamard product (elementwise multiplication) between two identically shaped tensors. Define

$$\epsilon_{ij} \overset{\text{i.i.d.}}{\sim} \text{Bern}\left(p\right), \tag{2.14}$$

then dropout applied to a tensor X with dropout rate p is

$$D^{\epsilon}(X)_{ij} = (\epsilon * X)_{ij} = \epsilon_{ij} X_{ij} \tag{2.15}$$

with the tensor ϵ sampled on a batch-wise basis. We denote the Hadamard product by $*$ rather than the more usual $\circ$ to avoid confusion with function composition. Applying RC gives the condition

$$D^{\epsilon}(RC_{l-1}(X))_{ij} = X_{-i,-j}\epsilon_{ij} = RC_l(D^{\epsilon}(X)) \tag{2.16}$$

which holds if

$$\epsilon_{ij} = \epsilon_{-i,-j} \tag{2.17}$$

and if $RC_l = RC_{l-1}$.

2.2.6 Bayesian Equivariant networks

Convolutional networks have been shown to work well on large datasets, but it is known that they overfit quickly when relatively little training data is available [Gal and Ghahramani, 2016]. This is often the case in the biological domain, calling for a principled approach to deal with limited training datasets. Networks with dropout after every convolutional layer and trained via backpropagation can be seen as approximating Bayesian variational inference [Gal and Ghahramani, 2016], promising good behaviour in data-poor settings. The interpretation also suggested to use

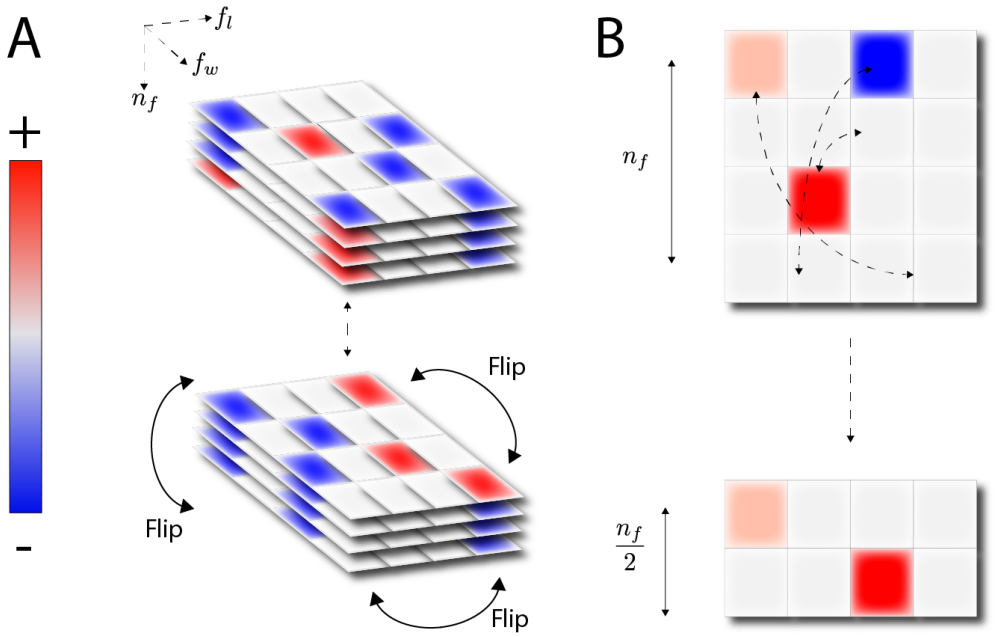

Figure 2.2: **Left:** The 3 dimensional filter tensor has enforced symmetry by weight tying, flipping the second half of the filter axis as shown. **Right:** In contrast to a conventional max pooling along the spatial dimension, this approach permits pooling along filters.

Bayesian dropout rather than traditional weight averaging for making predictions, using dropout to approximate sampling from a posterior weight distribution. In this interpretation, the corresponding prediction is obtained by the average over a sample of instantiations of the network:

$$p(Y^*|X^*, \mathbf{X}, \mathbf{Y}) \approx \frac{1}{K} \sum_{k=1}^{K} F(Y^*|X^*, \epsilon_k) \tag{2.18}$$

for unseen X^* given previous training examples $\mathbf{X} = (X_1, \ldots, X_n)$ and $\mathbf{Y} = (Y_1, \ldots, Y_n)$ used to train F. We refer to the process of using dropout during both training and test time as *Bayesian dropout*, while we use *dropout* to refer to networks that use weight averaging at test time.

The set of random variables $\epsilon = (\epsilon_1, \ldots, \epsilon_K)$ in 2.18 implement a random sample of the weights defining the network F. To implement equivariance, it is sufficient that these weights obey the correct symmetry, e.g. (2.17), as then each term will be RC-symmetric and so will the sum. The resulting function will be stochastic, but as long as ϵ is sampled and fixed beforehand, it will nevertheless exhibit exact RC symmetry.

2.3 Equivariant Recurrent Networks

In addition to convolutional networks, there have been a number of successful applications of recurrant neural networks to sequence analysis. These networks are designed for application on sequences, and operate such that each base is fed into the network sequentially, and the network retains state, 'remembering' the bases that have come before.

Here, we outline the construction of recurrent network topologies that obey reverse complement symmetry that is required for consistent forward and reverse strand computation. As discussed previously, the requirement for upper layers in such a topology is that the action of the layer commutes with the reverse complement operator.

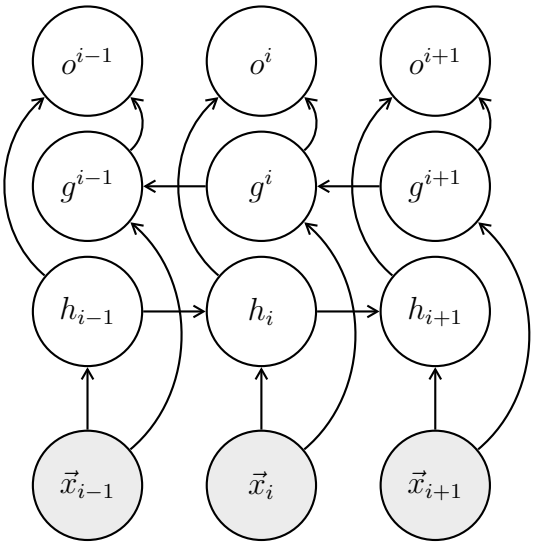

Figure 2.3: Schematic of a classical bidirectional RNN

Specifically for tensor function F_i:

$$F_i \circ RC^{(i)} = RC^{(i+1)} \circ F_i \tag{2.19}$$

where the indexing of RC refers to the fact that the dimension of the input tensor to F_i is not necessarily conserved, and indeed the action of this group may be different here.

2.4 Bidirectional Recurrent Network

We first begin with the model outlined in figure 2.3 which is a standard bidirectional recurrent computation graph. For a single timestep i, we have two recurrent architectures that require knowledge from either the previous or next step. Specifically for states h^i and g^i and abusing notation slightly:

$$g^i = g(X^{(i)}, g^{i+1}) = w^G X^{(i)} + v^G g^{i+1} \tag{2.20}$$

$$h^i = h(X^{(i)}, h^{i-1}) = w^H X^{(i)} + v^H h^{i-1} \tag{2.21}$$

Where in the case of a one hot encoded sequence, X is a length 4 1 dimensional vector. The outputs for these two calculations are concatenated in layer o^i:

$$o^i = g^i \oplus h^i \tag{2.22}$$

For vectors containing the outputs of g^i and h^i. Note that we have ignored both nonlinearities and biases, both of which are required in a practical network.

Define $Y_{ij} = (o^i)_j = (g^i \oplus h^i)_j$, then for the usual definition of the RC function: $RC(X)_{ij} = X_{-i,-j}$, one has

$$RC(Y_{ij}) = (g^{-i} \oplus h^{-i})_{-j} \tag{2.23}$$

$$= (RC(h)^i \oplus RC(g)^i)_j \tag{2.24}$$

Thus for this commutation to hold in the general case, one must have

$$RC(g) = h_{rc} \Leftrightarrow g_{rc} = RC(h) \tag{2.25}$$

with the subscripts indicating that these quantities are calculated with the reverse complement sequence input.

Now performing an explicit calculation of this, one can discover the implications for the weight symmetry that must be implied. Using the convention of summing over repeated indices:

$$h_i^{(1)} = \sum w_{ij}^H X_j^{(1)} + v_{ij}^H h_j^{(\text{init})} \tag{2.26}$$

$$\Rightarrow h_{rc,i}^{(1)} = \sum w_{ij}^H X_{-j}^{(-1)} + v_{ij}^H h_j^{(\text{init})} \tag{2.27}$$

$$= RC(g)_i^{(1)} \tag{2.28}$$

$$= RC(\sum w_{ij}^G X_j^{(-1)} + v_{ij}^G g_j^{(\text{init})}) \tag{2.29}$$

$$= \sum RC(w_{ij}^G) RC(X_j^{(-1)}) + RC(v_{ij}^G) RC(g_j^{(\text{init})}) \tag{2.30}$$

$$= \sum w_{-i,-j}^G X_{-j}^{(-1)} + v_{-i,-j}^G g_{-j}^{(\text{init})} \tag{2.31}$$

where in the penultimate line we have used the property that $RC(\sum_j X_{ij} Y_j) = \sum_j RC(X)_{ij} RC(Y)_j$. Note that for a 1 dimensional tensor $RC(X)_i = X_{-i}$. Now,

comparing coefficients of $X_{-j}^{(-1)}$ in 2.27 and 2.31 it is natural to require:

$$w_{-i,-j}^{G} = w_{ij}^{H} \tag{2.32}$$

The second term in each of 2.27 and 2.31 is more complex, and depends on the initialization of the recurrent states respectively. Subtracting the newly equal first terms from each equation yields:

$$\sum v_{ij}^{H} h_{j}^{(\text{init})} = \sum v_{-i,-j}^{G} g_{-j}^{(\text{init})} \tag{2.33}$$

which holds sufficiently if:

$$v_{ij}^{H} = v_{-i,-j}^{G} \tag{2.34}$$

$$h_{j}^{(\text{init})} = g_{-j}^{(\text{init})} \tag{2.35}$$

and therefore requires a special symmetric initialization. Now, for general k,

$$h_{rc,i}^{(k)} = \sum w_{ij}^{H} X_{-j}^{(-k)} + v_{ij}^{H} h_{rc,j}^{(k-1)} \tag{2.36}$$

$$= \sum w_{-i,-j}^{G} X_{-j}^{(-k)} + v_{-i,-j}^{G} h_{rc,j}^{(k-1)} \tag{2.37}$$

$$= \sum w_{-i,-j}^{G} X_{-j}^{(-k)} + v_{-i,-j}^{G} g_{-j}^{-k+1} - v_{-i,-j}^{G} g_{-j}^{-k+1} + v_{-i,-j}^{G} h_{rc,j}^{(k-1)} \tag{2.38}$$

$$= RC(g)_{i}^{k} + \sum -v_{-i,-j}^{G} g_{-j}^{-k+1} + v_{-i,-j}^{G} h_{rc,j}^{(k-1)} \tag{2.39}$$

which fulfills the required commutation if

$$h_{rc,j}^{(k-1)} = g_{-j}^{(-k+1)} \tag{2.40}$$

for all k, j. Thus:

$$RC(g)_{i}^{(k)} = h_{rc,i}^{k} \Rightarrow RC(g)_{i}^{(k+1)} = h_{rc,i}^{k+1} \tag{2.41}$$

and given that we have constructed it to be true for $k = 1$ it is inductively true for the entire network.

2.5 Advanced Recurrent Topologies

The above classical bidirectional RNN is not the most widely used recurrent architecture in genomics, predominantly due to the problems regarding learning long term dependancies, the so called 'vanishing gradient' problem [Hochreiter, 1998]. Two modifications to this framework that have been devised to counter this issues are the gated recurrent unit (GRU) [Chung et al., 2015] and the long short term memory (LSTM) [Hochreiter and Schmidhuber, 1997]. The idea of both of these networks is to change the recurrence relation for the recurrent units in equation 2.21 such that the ith output is updated additively instead of multiplicatively, allowing longer term dependencies to be learned. These approaches have both been used extensively in genomics in recent years [Ben-Bassat et al., 2018, Quang and Xie, 2016, Shen et al., 2018], and can also be shaped to respect RC symmetry.

A typical LSTM topology has

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \tag{2.42}$$

$$i_t = \sigma_g(W_k x_t + U_k h_{t-1} + b_i) \tag{2.43}$$

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \tag{2.44}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \tag{2.45}$$

$$h_t = o_t \circ \sigma_h(c_t) \tag{2.46}$$

with $\circ$ the hadamard product (i.e. elementwise multiplication), σ_g the sigmoid function and σ_c, σ_h being the hyperbolic tan or relu functions. It can be shown similar to above that a bidirectional architecture composed of LSTM units exhibits RC equivariance if (ignoring biases) $(W_f)_{ij} = (W_f)'_{-i-j}$, $(U_f)_{ij} = (U_f)'_{-i-j}$, $(W_k)_{ij} = (W_k)'_{-i-j}$, $(U_k)_{ij} = (U_k)'_{-i-j}$, $(W_o)_{ij} = (W_o)'_{-i-j}$, $(U_o)_{ij} = (U_o)'_{-i-j}$, allowing these superior memory properties to be incorporated into this framework. Here the $'$ notation is for tensors in the reverse LSTM cell.

In chapter 3, we briefly retrofit existing literature model [Quang and Xie, 2016]

with this framework, and find slightly disappointing results. This could be due to the specifics of the problem though as later discussed, as an approach as recent as July 2019 [Bartoszewicz et al., 2019] has used this exact framework to achieve state of the art performance at their chosen task of predicting pathogenic potential of novel DNA.

2.6 The Connection Between Equivariance and Data Augmentation

An alternative approach to improving the ability of deep learning models to understand symmetry operations applied to input data is that of data augmentation. This idea is a simple one: Manually add examples of the transformed input to the training data. There are two principal types of data augmentation, offline, where the training data is transformed in advance, and online where each batch is augmented with a selection of of the symmetry operations at the beginning of each batch of gradient descent. Here, we show the connections between online data augmentation where each batch contains every symmetry operation for each example and a simple equivariant network.

We start with a network consisting of a single neuron so, ignoring biases, we have

$$\hat{y} = \sigma(w^T x) \tag{2.47}$$

where σ is an activation function. In this case we specify that it is a sigmoid function, but this is not required in general. Now, this network is equivariant to a group of order N containing all the semantically identical transformations of input x that we logically require the same output from the network. For a single training example x, elements can be generated by actions of operator $O^{(1)}, O^{(2)}...$, with $O^{(N)} = I$.

We specify the following structure of the group operations:

$$O^{(k)}O^{(i)} = O^{(m_k(i))} \tag{2.48}$$

with $m_k(i)$ a bijective mapping. An example of such a group could be the order N rotation group $O, O^2, O^3....O^N$ with $OO^k = O^{(k+1) \mod N}$

Codifying this requirement, gives $N-1$ equalities, compactly written as:

$$\sigma(w^T x) = \sigma(w^T O^{(1)} x) = \dots \sigma(w^T O^{(N)} x) \tag{2.49}$$

for all x, w

As our choice of activation is bijective, this reduces to

$$w^T x = w^T O^{(1)} x = \dots w^T O^{(N)} x \tag{2.50}$$

For this network to maintain equivaraince, this is the weight property that must hold, and we therefore require it is initialized that way.

2.6.1 Training

To train this network, we have, for each x, as corresponding supervising example y, and a cost function $C(\hat{y}, y)$

Typically,

$$C(\hat{y}, y) = \sum_i^M (1 - y_i) \ln(1 - \hat{y}_i) + y_i \ln \hat{y}_i \tag{2.51}$$

for a total of M x, y pairs. Calculating the derivative with respect to a single example i yields:

$$\frac{\partial C_i}{\partial w_j}(x_i) = x_{ij}(\hat{y}_i - y_i) \tag{2.52}$$

$$= x_{ij}(\sigma(w^T x_i) - y_i) \tag{2.53}$$

Similarly, we wish to calculate the derivative of the cost function for the same example, that has been operated on by $O^{(ik)}$:

$$\frac{\partial C_i}{\partial w_j}(O^{(k)} x_i) = (O^{(k)} x_i)_j (\hat{y}_i - y_i) \tag{2.54}$$

$$= (O^{(k)} x_i)_j (\sigma(w^T O^{(k)} x_i) - y) \tag{2.55}$$

$$= (O^{(k)} x_i)_j (\sigma(w^T x_i) - y_i) \tag{2.56}$$

We note that the second term in the brackets is the same for both cases due to the earlier equivariance requirement.

We imagine a training regime for this network as follows:

- We train the network using stochastic gradient descent, with batch size N
- Each batch j is composed of the set of training examples: $x^{(j)}, O^{(1)}x^{(j)}, O^{(2)}x^{(j)} \dots O^{(N-1)}x^{(j)}$, and the corresponding $y^{(j)}$, which is identical for each of these inputs.

We then use the above results to calculate the gradient of the cost function with respect to just these batch examples:

$$\frac{\partial C^{(\text{batch } i)}}{\partial w_j} = (\sigma(w^T x^{(i)}) - y_i) \cdot \left(\sum_{k=1}^{N} O^{(k)} x^{(i)} \right)_j \tag{2.57}$$

At the end of this batch, we would then update the weights accordingly:

$$w'_j \to w_j + \alpha \left(\sum_{k=1}^{N} O^{(k)} x^{(i)} \right)_j \tag{2.58}$$

with the sum on the right hand side representing the elementwise vector sum so subscript j is the jth component of this vector

Then we find on application of an operator $O^{(q)}$:

$$\sum_j w'_j (O^{(q)}x)_j = \sum_j \left(w + \alpha \left(\sum_{k=1}^{N} O^{(k)} x^{(i)} \right) \right)_j (O^{(q)}x)_j \tag{2.59}$$

$$= \sum_j w_j (O^{(q)}x)_j + \alpha \sum_j \left(\sum_{k=1}^{N} O^{(m_q(k))} x^{(i)} \right)_j x_j \tag{2.60}$$

$$= \sum_j w_j (O^{(s\neq q)}x)_j + \alpha \sum_j \left(\sum_{k=1}^{N} O^{(m_s\neq q(k))} x^{(i)} \right)_j x_j \tag{2.61}$$

$$= \sum_j w'_j (O^{(s)}x)_j \tag{2.62}$$

by induction for all $s \in 1..N$ and arbitrary x, which is the necessary condition 2.50. The penultimate line follows, as we know the first term to be true by initialization, and the second term is true as each group operator is still included exactly once.

This is the required weight symmetry for the network to maintain equivariance, so at the end of this batch update we retain an equivariant network. Thus, a network of this type with equivariant initialization trained with data-augmented batches that include the complete set of symmetry transformations for each training example retain equivariance and give the same weight updates (modulo scaling factor) that a network that enforced explicit weight tying when trained on a single example at a time.

2.7 Discussion

We have previously illustrated a network symmetry scheme that allows networks to be trained in a manor that allows inputs related by symmetry group actions to identically map to the same outputs. However, this is clearly a small subset of the problems that one would want to model, predominantly classifcation or scalar regression. A clear example problem where the above prescription would not be sufficient would be a sequence based gene model, where we input genomic sequence, and require the prediction of different functional elements, such as promotor, UTR, coding region etc.

This general problem, where the input to the network is a one hot sequence of length N, and the output is a tensor where one of the dimensions of length N is equally addressable with this framework in the following way: Instead of performing reverse complement pooling as in equation 2.13, one can simply have an operation that permutes this output tensor $t_{ijk...n}$ along it's nth axis (the axis that has the same dimension as the input) i.e:

$$O(T_{ijk...n}) = T_{ijk...-n} \tag{2.63}$$

where as before negative indices are offsets from position N in this dimension. This greatly extends the applicability of this past simple classification and regression problems.

In the following chapter we test this approach and show that the equivariant bayesian network, (using equivariant networks with MC dropout) give state of the art results on simulated and real datasets, and demonstrate further desirable charac-

teristics, both in terms of training run-time and representational stability.

Chapter 3

Empirical Equivariant Results

3.1 Introduction

In this chapter, we apply the equivariant networks outlined in the last section to biological datasets. We apply these networks to both a simulated dataset, where we simulate simple transcription factor binding and a real dataset derived from recombination hotspots.

In both of these settings, which we suggest are representative of a significant class of biological problem, we demonstrate that the ubiquitous technique of dropout does not work and find significant improvement both by applying reverse complement equivariance to the networks, and combining this with equivariant bayesian dropout. In particular, we show the benefit of equivariance to be most pronounced when training with a smaller dataset. We then retrofit known literature methods to make them equivariant to reverse-complement symmetry but find somewhat mixed results.

We compare this approach, and the motifs derived from the trained network to current motif finders, and find that we achieve superior performance in finding motifs that discriminate between classes. Additionally, we show that enforcing equivariance in the networks yields substantially improved representational stability (consistency of motifs learned between independent training runs) for the trained networks. This is an important result, as a long term goal for the field is to be able to perform 'in

silico biology', where we use our models to learn from the data and then examine their structure to understand the salient biological features. This improvement in representational stability is important as it reduces noise between training runs, meaning we can recover, for example, more reliable sequence motifs.

We then use this approach to study the problem of understanding the motifs that determine whether or not a piece of genomic sequence is likely to fall into a meiotic recombination hotspot. This approach was able to recover known motifs for zinc finger protein PRDM9 binding, which is crucial for catalyzing this process, in addition to finding a new class of very long (22-23bp), very sparse motif that has been recently validated as predictive of PRDM9 binding via a direct chip-seq experiment. We show that this motif is statistically significantly enriched in hotspots in comparison to our hold-out set of sequences. We finally use a Bayesian approach to quantify the number of hotspots that are caused by PRDM9 binding at this motif and find that it accounts for approximately 15% of all known hotspots.

3.2 Datasets

Simulated data was generated as follows. As a model for a regulatory network involving two binding proteins, we sampled two PWMs (ATAF4 and ERF1) from JASPAR [Sandelin et al., 2004]. We first randomly sampled 40000 times a random $\{0, 1\}$-response with 50% probability for each, representing a measured phenotype of interest. For each response variable with value 1 we sampled a specific motif from each of the two PWMs (reverse complementing them at random) and injected it into a random background sequence with 40% probability. Otherwise we injected the motifs with probability 20%. This procedure resulted in 40000 sequences of length 1000, with 20110 in category 1 and 19890 in category 0. Note that there is a substantial amount of noise in this dataset, making for a challenging classification problem.

We also prepared a dataset of human recombination hot- and coldspots, similar to [Myers et al., 2008]. We first applied a simple hidden Markov model to segment the genome into regions classified as 'hot' and 'not hot'. For emission probabilities we used two exponential distributions for p(observed rate|hot) and p(observed rate|not hot), and used Viterbi training (a simple approximation to the full EM fitting procedure, that uses the maximum likelihood path at every iteration to estimate the model parameters) to set the parameters. The median recombination rate in regions classified as 'hot' was 10.5 cM/Mb. The length of hotspots, once annotated, had a median value of 2448 bp, but with a heavy tail up to around 20 kbp. For our purposes, we wanted to study localized recombination events, so we discarded all hotspots longer than 4 kb. After discarding sequences with unspecified bases, we then sampled a sequence of length 1kb from the centre of the remaining hotspots, yielding a total of 17552 truncated hotspots for classification.

To define a matched set of coldspot regions once hotspot regions were identified, we applied a greedy search within 300 kb of each hotspot region to identify a sequence within 10% GC content, and with a recombination rate below 0.5 cM/Mb. GC matching ensures that GC content, which tends to be higher in recombination hotspots due

to GC-biased gene conversion, cannot be used as a proxy for recombination strength, forcing the network to focus on causal signals of DNA binding motifs. This pipeline yielded a total of 17547 truncated coldspots, also of length 1000 bp.

3.3 Model Construction

For each of the two datasets, we sought to implement a deep learning model to fit to the data; attempting to classify between the two classes in this case by optimizing a cross entropy loss function. We then seek to compare the performances obtained here with that possible using sequence equivariant networks as outlined in the previous chapter. In the following subsection, we give an outline of the construction of these models, and the optimization scheme. We further present observations that motivate our choice of search space, particularly initialization schemes and activation functions.

3.3.1 Hyperparameter Search

To find optimal non-Bayesian non-equivariant networks we performed hyperparameter searches for both datasets independently (see figure 3.1 for details). Next, for each data set we built three extensions as follows:

1. An equivariant network, by adding an RC pooling layer and enforcing weight equivariance in the internal layers;

2. A Bayesian network, by adding one or two MC dropout layers;

3. A Bayesian equivariant network, by doing both, and changing MC dropout into equivariant MC dropout.

To build these networks we performed additional restricted hyperparameter searches on the position of the RC pooling and MC dropout layers, the type of RC pooling (max, sum or average pooling), and the L2 regularization parameter (because both equivariance and dropout are themselves inherently regularizing), while keeping all

other network parameters fixed (see appendix A for details of this, and the topologies of the final selected base and enhanced networks).

3.3.2 Choice of activation function

Rectified linear units (ReLUs), defined as $\mathrm{ReLU}(z) = \max(0, z)$, are activation functions applied at the terminus of every layer, providing the requisite nonlinearity for stacked layers to have richer representational power than a simple linear model. These activation functions have become ubiquitous in deep learning, though usually for networks which are composed of more layers than is typical in genomic problems, as they reduce or resolve the vanishing gradient problem. ReLU elements have the property that for a large number of units, the output will be identically zero. Although sparsity can be advantageous, e.g. by making interpretation easier, we found that it hampers convergence in all problems we have tried. We found that using ELUs [Clevert et al., 2015] result in substantially better convergence behaviour on our dataset (see figure 3.2). The ELU activation function is defined as

$$\mathrm{ELU}(x) = \begin{cases} x & x > 0 \\ \exp(x) - 1 & x \leq 0 \end{cases} \tag{3.1}$$

which the authors suggest yield improved results by bringing the mean activations for a layer closer to zero. We observed qualitatively similar results using shifted ReLUs ($\mathrm{SReLU}(z) = \max(z, -1)$) (data not shown).

Recombination

Parameter	Options
# Convolutional layers	1 . . . 5
Filter length of 1st Convolution layer	10, 15, 20, 30
Filter lengths of internal layers	4, 8, 12
Max Pool sizes (stride is always pool size)	4, 8, 12
Number of filters	8, 16, 32, 64
L2 regularization	0, 0.0001, 0.0003, 0.001
Learning rate	0.1, 0.01, 0.001, 0.0001

Simulation

Parameter	Options
# Convolutional layers	1
Filter length of 1st Convolution layer	14
Number of filters	4, 6, 12
L2 regularization	0, 0.0001, 0.0003, 0.001
Learning rate	0.1, 0.01, 0.001, 0.0001

Figure 3.1: The hyperparameter search space for the networks selected for the two datasets. For the Recombination dataset, networks were. sampled at random 1000 times and trained to convergence with ADAM. For the Simulation datatset, networks were sampled at random 200 times and trained to convergence, also with ADAM.

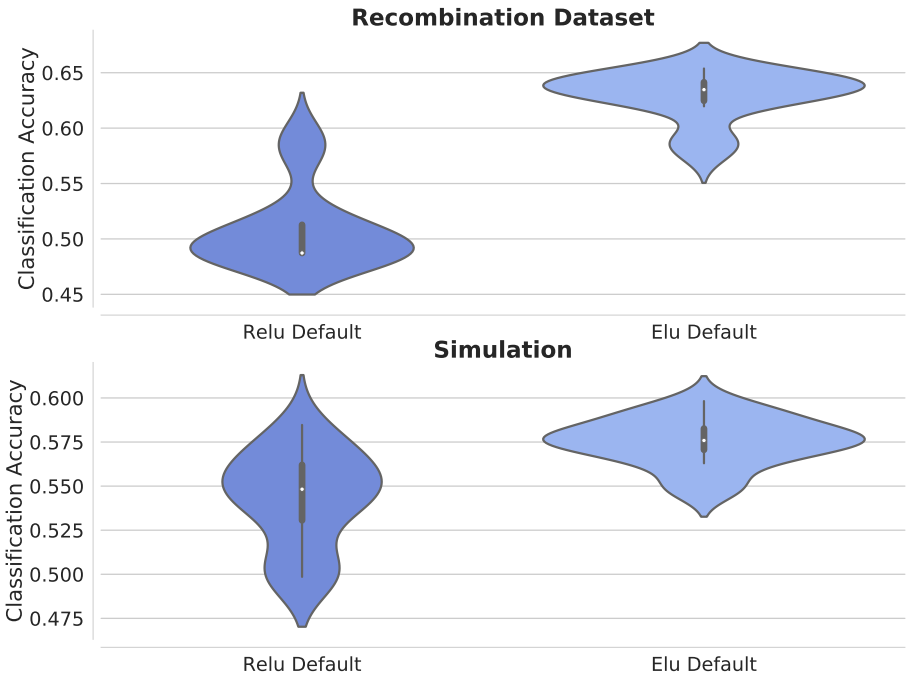

Figure 3.2: Convergence accuracy distributions for the best default topology as found in appendix A with the only difference being the activation functions. These distributions were built from 50 complete training cycles. Note that in the null case (where the classifier selects a class at random), we expect an AUROC of 0.5 for balanced classes.

3.3.3 Recovery From Partial Initialization

As a further test of the robustness of the ReLU activation function, we tried a partial initialisation of the filters on the simulated data where we know that there are 2 driving motifs and their reverse complements, ATF4 and EGR1. In order to aid the network in convergence, two of the filters equal to the PWM of ATF4 and it's reverse complement, and then trained an equivariant network from this point to observe the classification accuracy at convergence. Each column of the PWM sums to 1. The results of this analysis are in figure 3.3. Strikingly, we found the ReLU activation

function struggled to converge from this seemingly favourable initialization, and was almost always unable to find the EGR1 motif. By way comparison we built an identical network, swapping only the activation functions for either an Exponential Linear Unit (ELU) of a shifted ReLU (SReLU). Both of these functions performed substantially better, and were much more readily able to detect the remaining signal.

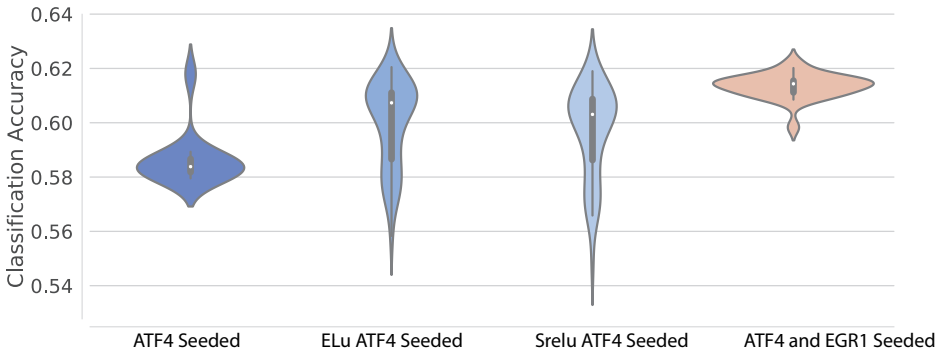

Figure 3.3: Convergence accuracy for Equivariant network with ReLU, ELU and SReLU activation functions on the simulated dataset, built from 50 trial runs. We seeded the filter bank with PWMs for the ATF4 motif, and its reverse complement and then trained the network to convergence

3.3.4 Initialization of the output layer

We found that using a custom initialization of the output layer, providing the classification scores before a final softmax transformation, substantially improved convergence for the networks we considered (figure 3.4 top). We initialized the weight matrix of the final layer with 1, and the two bias parameters (corresponding to the two nodes representing class probability) with $\{1, -1\}$. A motivation for this choice was the observation that on a number of problems the output layers weights were always close to these values, so it appears that this initialization is closer to the global optimum than traditional approaches such as those proposed by [Glorot and Bengio, 2010a].

We also tested networks with Batch normalization [Ioffe and Szegedy, 2015] applied after the convolutional layers both with and without custom initialization to address this problem, but found that it did not improve convergence by as much as custom initialization did (figure 3.4 bottom).

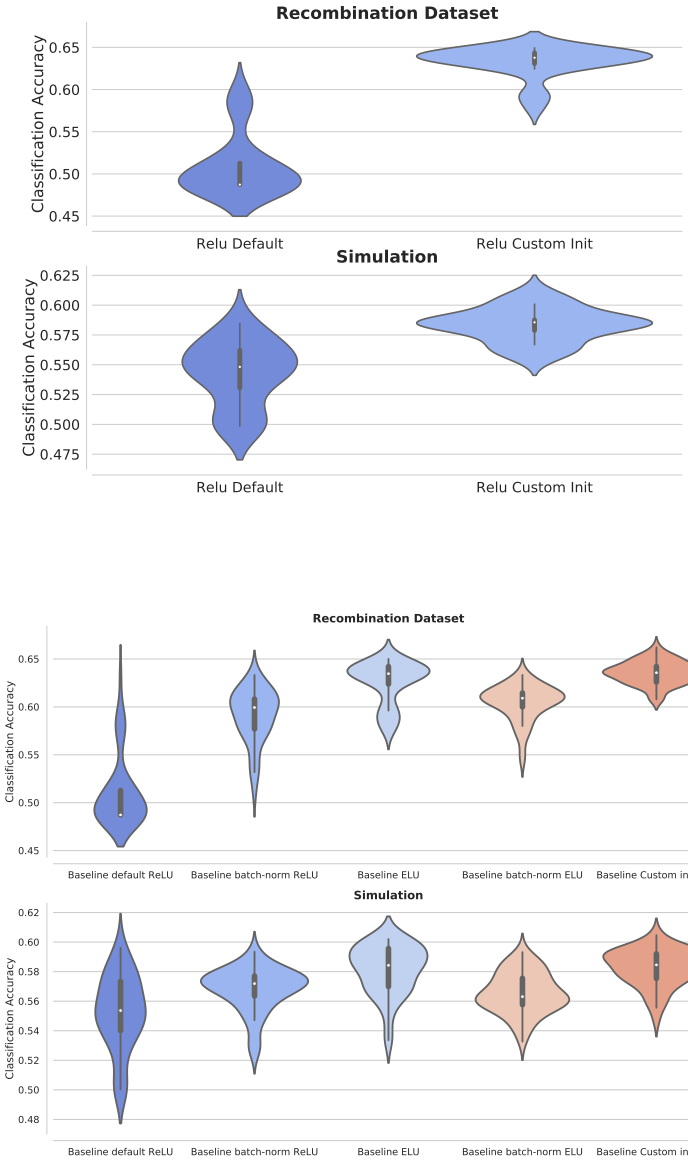

Figure 3.4: **Top:** The difference in network convergence accuracy for 50 trials for the best default network in appendix A with ReLU activations with and without the custom initialization of the final layer. **Bottom** The same network as above with the optional addition of batch normalization after the convolutional layer(s) with and without custom init with ReLU and ELU activation functions. The best performing topology across both networks is ELU, no batch norm, custom init.

3.4 Results

3.4.1 Equivariant Bayesian networks improve classification accuracy

Comparing the best-performing networks resulting from the optimization procedure outlined in section 2.10 we found that for both datasets the equivariant Bayesian networks outperformed the best-in-class non-equivariant networks, achieving significantly better test accuracy over a sample of 50 runs (Figure 3.5). We found that both equivariant non-Bayesian and Bayesian non-equivariant networks were significantly more accurate than the best in class convolutional network, and that the combination of Bayesian dropout and equivariance was again significantly more accurate than either.

We also investigated how well the equivariant Bayesian network performed for these two problems in comparison to classical data augmentation, whereby the reverse complement of every sequence is added to the dataset, at the cost of doubling the training time in comparison to the unaugmented non-equivariant networks.

To perform this experiment, we repeated the hyperparameter optimization procedure in section 3.3.1 for each of the two datasets with the reverse complemented sequences added to the dataset. This was necessary, as certain features, such as the number of filters or the L2 regularization may change for the best in class networks with data augmented, so any comparison to equivariant networks for the unaugmented datasets would be misleading. In fact, for both datasets the number of convolutional filters was doubled, and the for the recombination dataset, it was optimal to remove the l2 regularization completely.

As expected, data augmentation improved the accuracy compared to standard networks trained without data augmentation. Equivariant Bayesian networks remained significantly better than the best discovered Bayesian only network trained with data augmentation for the simulated dataset, while results were comparable for the re-

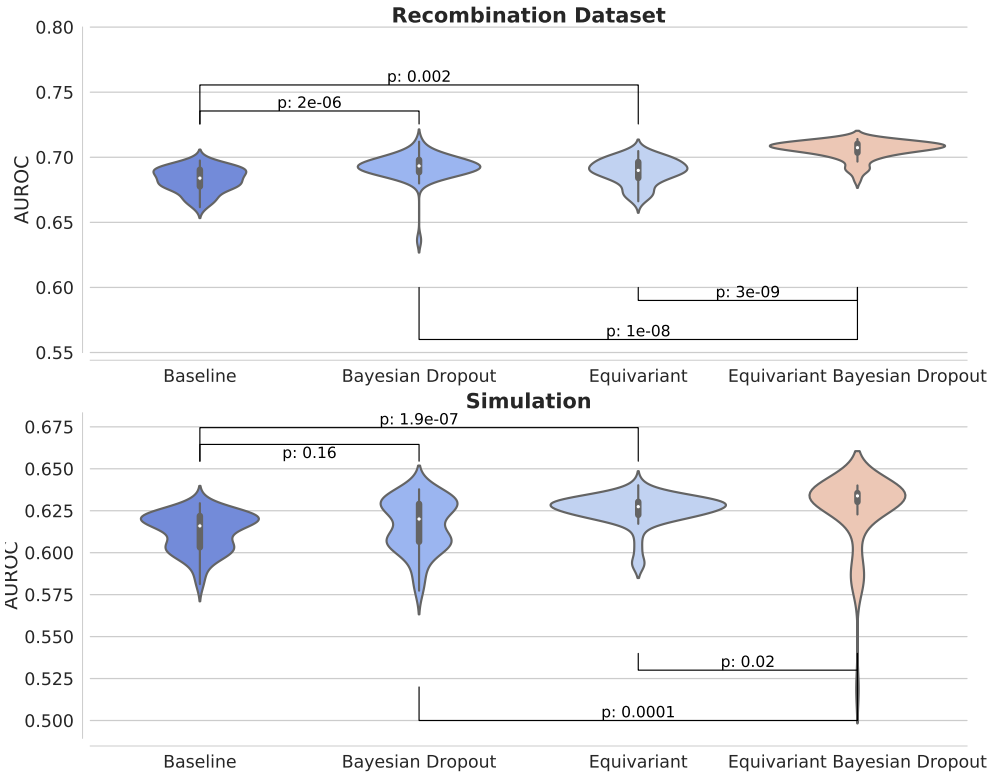

Figure 3.5: For every network configuration, we trained to convergence a total of 50 times and plotted the distribution of the final AUROC (area under the ROC curve) statistic. For both the simulation (top) and the recombination data (bottom) we find a statistically significant improvement over the case where either equivariance or Bayesian dropout is not applied. p-values calculated using the Wilcoxon test.

combination dataset (see supplementary figure 1 in appendix A). Here, we still note an empirical advantage in the gradient update speed in the equivariant network of 32% in comparison to 2 updates in the augmented data case, and 8% in comparison to the augmented case where we double the batch size for improved computational efficiency. The reason that this is not double the speed of the data augmented case is due to the additional gradient updates required to maintain weight tying in these equivariant networks.

Finally, we performed a robustness experiment to determine how the specified networks performed when the data quantity is reduced. We compared the networks with weight equivariance but no MC dropout (the 3rd network across in figure 3.5 in the main text, which is the same as the networks in appendix A without dropout) and equivariance with MC dropout to the default networks in appendix A, in the regime where we remove first 50% and then 75% of the data. For the recombination network, we find that the equivariant networks here provide increased convergence accuracy in comparison to the asymmetric versions, suggesting that this approach may be even more useful when there are fewer training examples than the c. 26,000 we have in our datasets. (Note that this isn't the same number of sequences given in the main experiment as we split into training, validation and testing tranches.)

For the simulated data, we observed a bimodal distribution with dataset size reduction, as the network struggles to converge reliably, possibly due to the regularizing effect of both dropout and weight equivariance in the filters. What is striking, is that the best results still belong to the equivariant Bayesian network, but this is dragged down by the increased numbers of failed convergences. Note that these results represent an incredibly noise domain with a convergence auroc of only around 0.54. Results in 3.6.

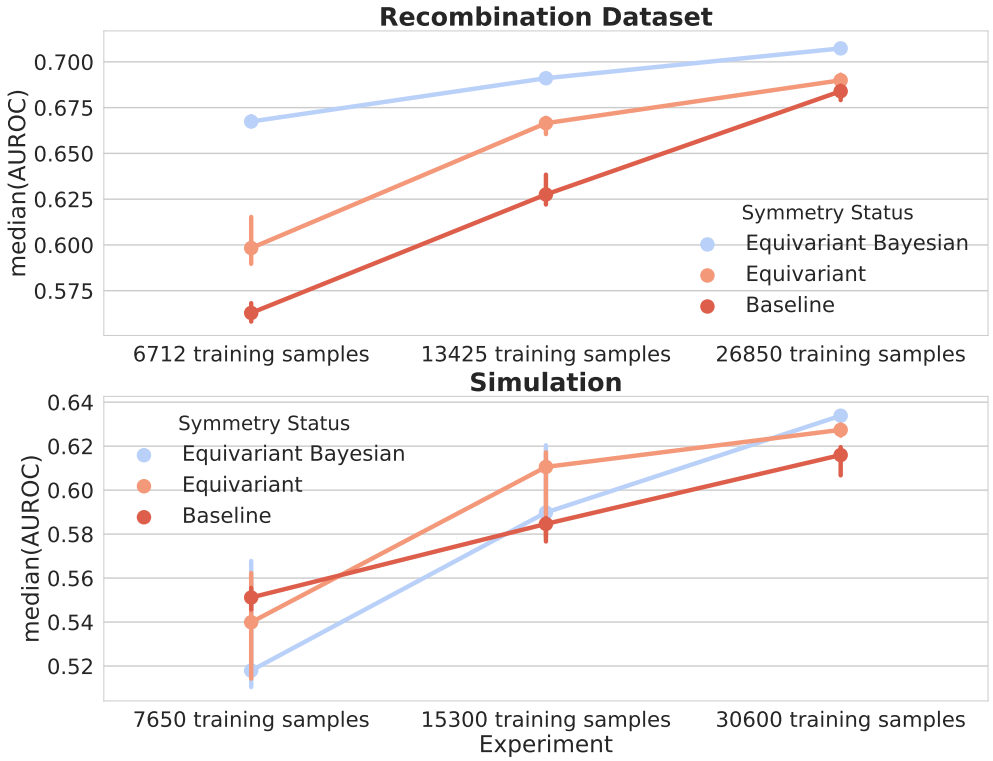

Figure 3.6: Comparison of the mean test AUROC for equivariant vs asymmetric networks show increased accuracy as train set size is reduced.

3.4.2 Conventional dropout considered harmful

To understand the contribution of Bayesian dropout on the performance of the network, we compared the final asymmetric topology for both datasets with no dropout, Bayesian dropout and conventional dropout; the latter uses dropout during training but weight scaling during test time. We recorded the mean accuracy over 50 trials. Compared to Bayesian dropout, an identical network that used the conventional dropout procedure yielded substantially inferior results (Table 3.1), and performs worse than the baseline without dropout. This behaviour was seen in both datasets. It appears that the approximation of weight averaging breaks down in this setting, apparently causing biases in the output leading to reduced accuracies.

Dataset	Baseline	Bayesian Drop.	Conventional Drop.
Simulation	58.2 ± 0.3	**59.2± 0.2**	56.1 ± 0.2
Recombination	63.5 ± 0.2	**64.5± 0.2**	58.2 ± 0.3

Table 3.1: Mean test accuracies (± one standard error) for the symmetric networks with no dropout, Bayesian dropout and conventional dropout.

3.4.3 Equivariant Bayesian networks yield more consistent predictions between runs

For the purposes of motif discovery and interpretation, it is advantageous for networks to result in consistent internal representations between training runs. It has been noted previously [Shrikumar et al., 2017a] that convolutional neural networks tend not to learn stable internal representations across different training runs, increasing the potential for different prediction results on the same test data. Enforcing equivariance reduces the potential for such representational instability, and we hypothesize that this would increase the consistency of predictions on hold-out data.

To test this hypothesis, we trained networks 50 times independently on identical training data, computed predictions on identical hold-out data, and computed 1225 pairwise correlations of these predictions. We performed this experiment for the previously determined optimal networks for the case of training with and without data augmentation, and for both cases we made the model equivariant without otherwise changing the topology. We found that in all four cases, the equivariant networks achieved significantly higher correlation between runs than their non-equivariant counterparts with the same number of filters (Fig. 3.7). We note that a hyperparameter search specifically for equivariant networks might further improve performance, but for this experiment we did not pursue this. The improved prediction consistency indicates that equivariance result in more training that converges more reliably to the same optima, which would be expected to improve the consistency of

the harvested motifs.

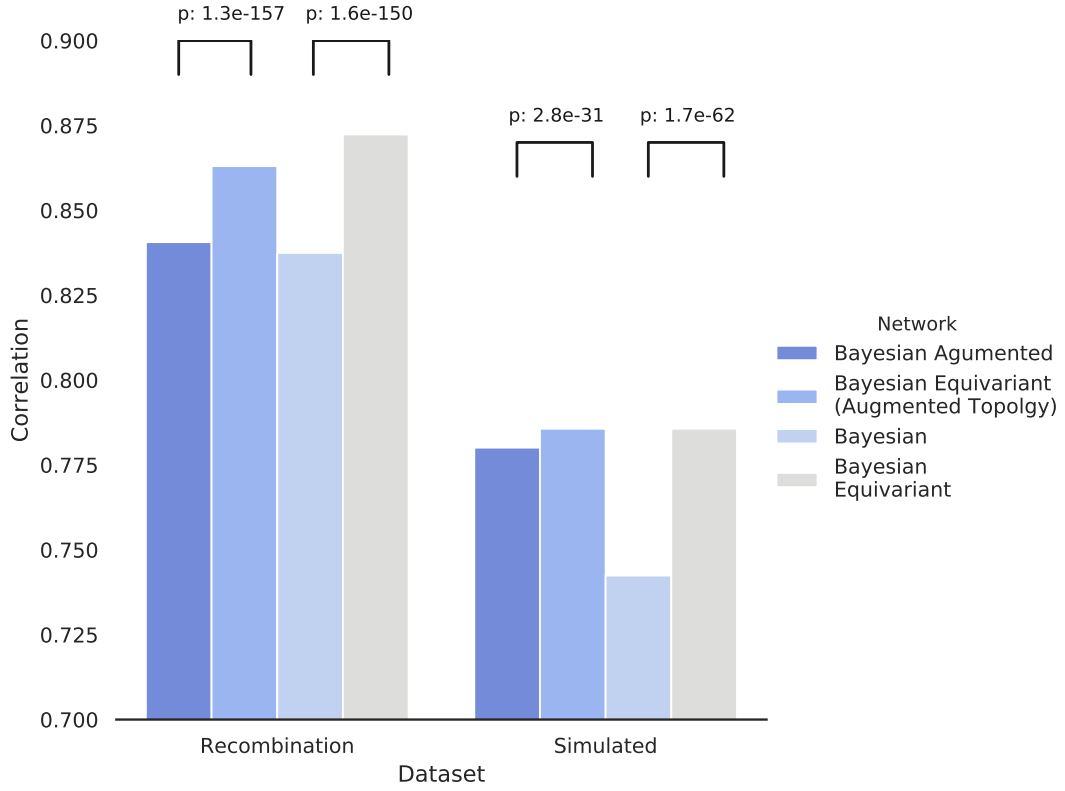

Figure 3.7: Mean of correlation of predictions on unseen data for pairs of networks that were independently trained on identical data. Shown are results for Bayesian networks optimized on both augmented and unaugmented data, and for the same networks but constrained to be equivariant; the latter show significantly more consistent predictions than non-equivariant networks. Full architectural details given in supplementary materials section A.

3.4.4 Equivariant Bayesian networks improve upon existing motif finders

To assess our approach we compared our results with two state-of-art motif finders, DeepMotif (DeMo; [Lanchantin et al., 2016]) and HOMER [Heinz et al., 2010], using the recombination dataset.

For the neural-network-based algorithm DeMo we benchmarked our results against

the three network configurations offered by the package, CNN, RNN and CNNRNN. All of these have substantially more complexity and weights than the model we used, but failed to perform well on this task. Indeed the CNN model (a convolutional neural network) often failed to converge at all, a problem that we also observed when training our vanilla convolutional networks on these datasets. The first training attempt that converged to better than random predictions had AUROC 0.58. Both the RNN and CNNRNN toplogies converged reliably, but achieved AUROCs of 0.64, substantially less than the median AUROC of 0.71 achieved by our equivariant network.

3.4.5 Homer Motif Results

We ran HOMER(Heinz 2010) on the same dataset, using default parameters except for setting a target motif length of 22 in an attempt to find the sparse motif we discovered, but were unable to recover it. We did however identify the canonical PRDM9 motif, and a number of spurious looking motifs (see Fig. 3.8). Whilst it is possible that another parameter setting would yield better results, it is difficult to tune these parameters in an analogous fashion to performing a hyperparameter search due to the lack of a cost function to optimize and the fact that it takes a long time for each run to execute.

Whilst we fail to generate the motif of interest we do generate 1 interesting result: Specifically, motifs 1-3 contain within in them stretches that correspond to the canonical PRDM9 13mer binding sites that was discovered in Myers (2008). This is an unsurprising find, as it was discovered on a similar dataset curated from recombination hotspots using an exhaustive search. Aside from this, we only find motifs 5,9 and 12 with substantial presence, however they bear little resemblance to the discovered sparse motif we identify.

Rank	Motif	P-value	log P-pvalue	% of Targets	% of Background
1		1e-200	-4.607e+02	0.80%	0.01%
2		1e-145	-3.356e+02	62.76%	53.13%
3		1e-116	-2.678e+02	1.86%	0.38%
4		1e-64	-1.491e+02	0.34%	0.01%
5		1e-53	-1.238e+02	36.39%	30.91%
6		1e-51	-1.191e+02	0.29%	0.01%
7		1e-51	-1.188e+02	0.24%	0.01%
8		1e-40	-9.411e+01	0.36%	0.03%
9		1e-32	-7.444e+01	65.12%	60.76%
10		1e-31	-7.185e+01	1.79%	0.85%
11		1e-14	-3.445e+01	0.10%	0.01%
12		1e-12	-2.793e+01	75.44%	73.10%

Figure 3.8: Top motifs found using a discriminate search with HOMER. Running on a single 1.7 GHz Intel Core i7 core, the results presented were generated in c. 24 hrs

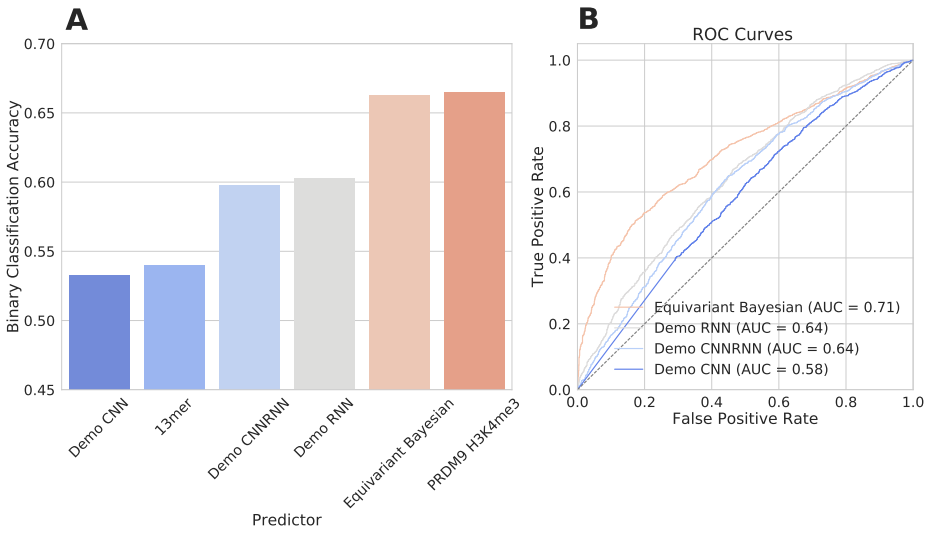

Figure 3.9: Classification accuracies (left) and (area under) receiver operator curves (right) for the equivariant Bayesian network and the three models provided by the Deep Learning based motif finder DeMo [Lanchantin et al., 2016]. For comparison we also show the classification accuracy of the canonical PRDM9 13mer motif, as well as the classification accuracy obtainable by using the differential histone mark H3K4me3 under PRDM9 overexpression, a direct proxy for PRDM9 binding [Altemose et al., 2017].

3.4.6 Discovery of novel PRDM9 binding motifs

Using the equivariant Bayesian network with the best classification accuracy, we recovered binding motifs by identifying a subset of input sequences responsible for the activation of particular nodes at the first layer, finding the subsequence driving the activation in each of those, and building a Position Weight Matrix (PWM) from the aligned subsequences.

This process identified five motifs, three of which are versions of the classical 13mer that was identified via enrichment analysis and exhaustive search of motifs in [Myers et al., 2008] (Fig. 3.10c-e). In addition we identified two substantially longer (22 nt) and sparse motifs that were not previously identified using this dataset (Fig. 3.10a,b). Since our networks are only approximately Bayesian, to confirm that these motifs were not artefacts of overtraining, we confirmed their significance by frequentist tests of significance for enrichment on hold-out data (Table 3.2). The enrichment of the sparse motif is similar to that of the canonical 13mer originally associated with PRDM9 binding, but its complexity and under-constrained nature (only about 8 out of the 22 bases of the motif have substantial information content) mean that it was not originally discoverable using a traditional enrichment approach. We discuss this finding further in the next section.

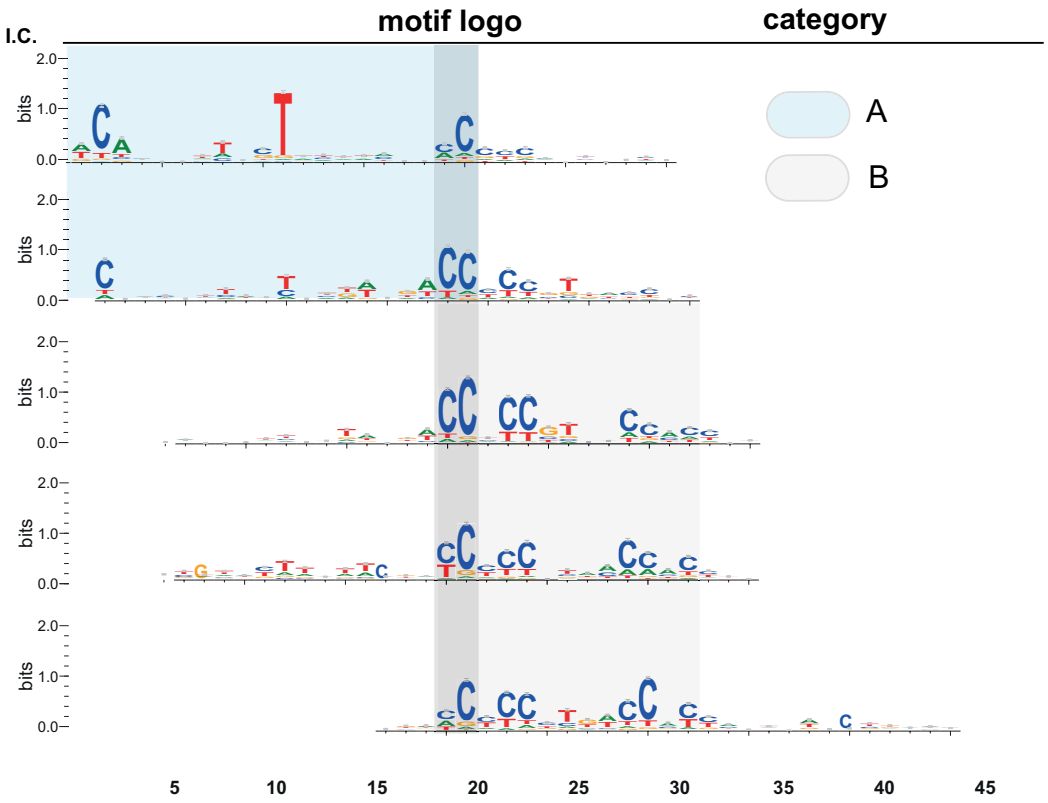

Figure 3.10: **Top:** We found the sequences corresponding to the maximum activation on each filter, and from these built sequence logos. We found a number of motifs (category B) corresponding to the classical PRDM9 binding motif that was discovered with an exhaustive search of an analogous dataset [Myers et al., 2008]. Additionally, we found two novel 22-base motifs (category B) that were previously undiscovered on this dataset. It is notable that they bear a striking resemblance to the recently discovered motif by [Altemose et al., 2017].

Motif	Total hot	Total cold	Test hot	Test cold	Odds ratio	Corrected p value
CANNNNTNNTNNNNNNNCCCC	2045	1186	83	38	2.16	2.03×10^{-8}
CANNNNTNNTNNNNNNNCCCCC	652	251	29	8	3.63	1.4×10^{-5}
CANNNNTNNTNNNNNNNCCNCC	3919	2933	158	109	1.45	4.7×10^{-6}
CCNCCNTNNCCNC	5695	3418	436	265	1.64	3.92×10^{-29}
CCTCCCTNNCCAC	400	77	57	14	4.07	5.34×10^{-12}
sub motif CCNCC	22370	22273	1431	1496	0.96	0.31

Table 3.2: Comparison of the novel motif with previously known PRDM9 binding site in both the training and test datasets. Total hot/cold, number of motifs in the full hotspot/coldspot dataset; test hot/cold, number of motifs among hold-out test data consisting of 1454 hotspots and 1530 coldspots; corrected p value, Bonferroni-corrected p values for association (Binomial test).

3.5 Discussion

Deep learning approaches have been shown to be very effective in building sequence models on large scale data [Zhou and Troyanskaya, 2015a, Kelley et al., 2016, Quang and Xie, 2016]. However, through simulated and biological data we show here that models designed using traditional building blocks for neural networks may struggle to converge consistently and produce reliable results in cases where the signal in the data is weak, and the amount of training data is limited. This problem was seen across a large number of network architectures, as well as in methods specifically designed to identify transcription factor binding sites from sequence data [Lanchantin et al., 2016]. To improve on this situation, we showed how to combine equivariant neural networks (here, neural networks that exhibit exact reverse-complement symmetry) with Bayesian dropout. While a naive combination of these ideas would result in networks that are only in expectation reverse-complement symmetric, we show that it is possible to achieve exact RC symmetry. In addition, we find that by modifying the activation functions, and the initialization of the output layer, we obtain a further significant improvement in accuracy.

Equivariant networks can be implemented in several ways. We chose to implement a standard (non-equivariant) network, enforcing equivariance by requiring certain identities on the parameters. Although this introduces some extra computation, this approach yields a number of advantages in comparison to data augmentation, where the training data is made symmetric and symmetry must be learned by the network. We find that in comparison to data augmentation, an equivariant network reduces the update time and results in equal or better test accuracies and improved consistency between training runs. Equivariant networks have the additional advantage of guaranteeing identical predictions on the forward and reverse strand, which may be desirable in applications.

We also found that Bayesian dropout resulted in a substantial performance improvements. This is striking, as dropout is normally thought of as a regularization

technique that reduces the tendency of overtraining often found in large model. By contrast, in our regime the final models were small in comparison to the number of samples in the training set, and we did not see much evidence of overfitting either with or without using dropout, such as a substantial continued decrease of training loss after test loss stabilised, and we did not find that the model latched on to spurious motifs that were not statistically significant on test data. It appears that, in addition to addressing overtraining, Bayesian dropout leads to superior learning and feature extraction. It would be interesting to confirm this phenomenon in different settings.

We then applied a network incorporating both Bayesian dropout and equivariance to the problem of predicting meiotic recombination hotpots directly from sequence. It was straightforward to interpret the resulting model, and interrogation of the sequences that maximally activated the input layer revealed the previously characterized 13-base PRDM9 binding motif [Myers et al., 2008]. Additionally, we discovered a much sparser motif, with only about 8/22 bases showing substantial information content. Like the classical motif, these new motifs were statistically significant on hold-out data, indicating that they were truly predictive features and not artifacts of an overtrained model. These sparse motifs (category A in figure 3.10) bear strong resemblance to motifs recently shown to be associated with PRDM9 binding [Altemose et al., 2017], lending further support to this conclusion. In fact, the observation that the model was able to achieve predictive accuracy on a par with hotspots predicted using differential H3K4 trimethylation as an input (also from [Altemose et al., 2017]) is consistent with the hypothesis that the neural network model represents a near-optimal model for PRDM9 binding (Fig. 3.9). The alternative but less parsimonious explanation is that our model and the the H3K4me3 assay are suboptimal to the same degree.

From a practical point of view, we noted that the motifs generated from this network with Bayesian dropout were qualitatively different and more numerous than the motifs identified with a classical convolutional approach, and we see a degree of degen-

eracy among the motifs learned by our network. It remains to be seen whether these different binding motifs correspond to related but slightly different binding modalities, or whether these motifs are a result of dropout training and provide a way for the network to robustly identify motifs in the presence of weight noise. This would be an interesting direction for further research. Certainly, if indeed these motifs do correspond to different binding modalities, each with slightly different binding affinities, it would explain why these networks are able to achieve the superior classification accuracies compared to standard models.

3.6 Quantifying the activity of these novel motifs

In order to quantify the importance of these newly discovered motifs, we performed an analysis using a bayesian approach analagous to that in [Myers et al., 2008]. Specifically, what we wish to calculate is the faction of hotspots that are caused by the new spare 22-23 base pair motifs over and above the canonical 13mer that was discovered originally. To do this we define the same bayesian model originally used, that allows different instances of the motif to have different effect sizes. Additionally we note the context that the motif appears in, and further stratify motif categories by this as well. This is important, as the motif has highly variable penetrance depending on small sequence and context modifications, and as such failure to account for these differences yields an underestimate of the fraction of causal motifs [Myers et al., 2008].

We first reconstructed a similar dataset to the original, with all hotspots less than 5kb from the 1000 genomes data matched with identical length coldspots that were matched to within GC content within 10% and snp density within 10% with a search similar to before. We searched for coldspots as close as possible to the respective hotspots, moving incrementally further away by a distance of 0.2 hotspot length, first upstream and then downstream. If we had been unable to locate a suitable coldspot within a distance of 1000 hotspot lengths, then we ceased the search. This procedure produced a total of 22694 hotspots and 22655 coldspots.

We used the UCSC hg19 repeatmask track to locate known repeats, and their positions within the hotspots. Finding similar hotspot enrichment to that observed in table 1 in [Myers et al., 2008]. We note 2 qualitative differences between our data here and the original results. Firstly, we see a substantial increase in the number of L2 elements, relative to the original analysis, which is explained by a substantially increased number of L2 repeats noted in in the hg19 repeatmask track vs the authors original version (there are around 50000 more l2 repeats). Further to this, we notice a different effect direction in the ALUY repeats. This is only a mild effect however, and was the least statistically significant previously reported effect sizes. The large odds ratios for hotspot and coldspots for the remaining elements (especially THE1A) still remain. Our repeat counts are presented in table 3.6.

Element	**#Hotspot**	**#Coldspot**
L2	19328	15765
AluY	3525	3779
AluSg	1217	1162
AluSc	974	1119
GA-rich	661	452
THE1A	294	129
THE1B	1334	776
THE1C	315	369
THE1D	473	456
(CCCT)n	45	34

Table 3.3: Repeat element counts for our curated hotspot and coldspot dataset

3.6.0.1 Model Outline

Here we give an outline of the model used in [Myers et al., 2008] used to calculate the original estimate that around 41% of hotspots have an active PRDM9 binding motif. This model stipulates that hotspots can be caused by motifs in one of N categories, and that within each category, each motif has the same probability p_m of leading to a hotspot. We call such a motif an 'active' motif. As a special case of the binomial distribution if within a genomic interval there are k instances of the motif, then there is a probability $p = (1 - p_m)^k$ that there are no active motifs. The complement of this is the probability that there is at least one active motif in this interval: $p_{\geq 1} = 1 - (1 - p_m)^k$.

It is further assumed that all motifs found in coldspots are inactive. This is effectively an assumption that motifs can only act proximally to cause hotspots, and that none of our coldspots are called incorrectly. Therefore, one can write down the probability of observing k motifs in a region designated to be a coldspot as:

$$P(k|\text{cold}, m) = \frac{P(k \cap \text{cold})}{\sum_l P(l \cap \text{cold})} = \frac{(1 - p_m)^k f(k)}{\sum_l (1 - p_m)^l f(l)} \tag{3.2}$$

Here, $f(k)$ is the prior distribution of seeing k motifs in category m within this window. This is defined below, and is a function both of the genome wide prevalence of motif m and the width of the interval in question.

The key target of inference in this model is a_m, i.e the fraction of hotspots that contain an active motif of class m. Thus for a single hotspot: We have $P(\text{Active motif}) = a_m$.

We note that a hotspot with no active motif, is treated identially to a coldspot, and thus:

$$P(k|\text{Hot, No active motif}, m) = P(k|\text{cold}, m) \tag{3.3}$$

and again the formula for k occurances of motif in category m in a hotspot that contains an active motif in that category is:

$$P(k|\text{Hot, Active motif}, m) = \frac{(1 - (1 - p_m)^k) f(k)}{\sum_l (1 - (1 - p_m)^l) f(l)} \tag{3.4}$$

and thus it is possible to marginalize over these two terms to get a formula for all hotspots without knowledge of whether they contain an active motif or not:

$$\begin{aligned} P(k|\text{Hotspot}) =& P(k|\text{Hot, Active motif}, m)P(\text{Active motif}) \\ &+ P(k|\text{Hot, No active motif}, m)P(\text{No active motif}) \end{aligned}$$

and substitute in for the terms defined above.

For the prior distribution $f(k)$, the authors originally used a poisson distribution, but noted greater variability in the data than would be expected under this assumption. Consequently they used a negative binomial distribution with 2 free parameters, which is also the approach we take. The parameterization of this distribution is the following:

$$P(k|\text{Hotspot width:}w, \theta, \lambda) = \frac{\Gamma(k+\theta w)}{\Gamma(\theta w)k!}\left(\frac{1}{1+\frac{\lambda}{\theta}}\right)^{\theta w}\left(\frac{\frac{\lambda}{\theta}}{1+\frac{\lambda}{\theta}}\right)^{k} \tag{3.5}$$

which permits the nice property that genomic intervals are additive.

Therefore, it is now possible to perform a maximum likelihood calculation for each category. If each of the N motif categories has hotspot and coldspot counts $\bar{h}, \bar{c}$ (where these are vectors the same length as the number of hotspots and coldspots respectively), and the width of the hotspots and coldspots is $\bar{w^h}, \bar{w^c}$. Then one can maximize over the parameters a_m, θ, λ. The full likelihood can then be written down as:

$$L(\bar{h}, \bar{c}|\bar{w^h}, \bar{w^c}, a_m, \theta, \lambda) = \tag{3.6}$$

$$\prod_i \left((1-a_m)\frac{(1-p_m)^{h_i}}{\sum_l (1-p_m)^{h_l} p(h_l; w_i^h, \theta, \lambda)} + a_m \frac{(1-(1-p_m)^{h_i})}{\sum_l (1-(1-p_m)^{h_l}) p(h_l; w_i^h, \theta, \lambda)}\right) p(h_i; w_i^h, \theta, \lambda) \tag{3.7}$$

$$\cdot \prod_i \left(\frac{(1-p_m)^{c_i}}{\sum_l (1-(1-p_m)^{h_l}) p(h_l; w_i^c, \theta, \lambda)}\right) p(c_i; w_i^c, \theta, \lambda) \tag{3.8}$$

Where we use the trick for each p_m of defining is as

$$p_m = \frac{a_m \cdot N_{\text{hotspots}}}{\lambda \cdot \text{Genome Length}} \tag{3.9}$$

which is a somewhat intuitive relation between these two quantities, and reduces the number of variables we infer from 4 to 3 for each motif category. This helps substantially with optimization speed.

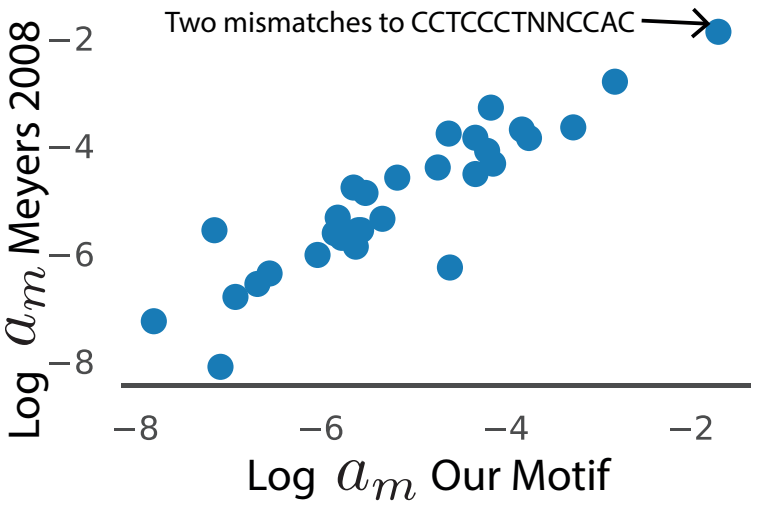

Figure 3.11: The strong agreement between our a_m's vs a_m's reported in [Myers et al., 2008] for the categories common to both approaches that yielded non-zero effect sizes. Note that there are 31 points here vs the 46 in [Myers et al., 2008] due to this filtering.

3.6.0.2 Motif Categories

We used a similar partitioning of motifs into categories of variable effect size as in [Myers et al., 2008]. There is a slight discrepancy between our motif categories and the originally published ones, as we collapsed the last 5 categories in [Myers et al., 2008] (supplementary table 4.) which contain various instances of repeats outside of the explicitly labelled motifs (L2, AluY, AluSc, AluSg, THE1A, THE1B, THE1C, THE1D).

These categories were absorbed into the first 7 categories, which originally contained instances of the motif of interest in unique DNA. In our approach, these are now considered to be instances of motif outside of otherwise declared repeats, be they unique DNA or other repeats. As a consequence, we are left with a total of 41 separate categories, as opposed to the 46 used originally. Nevertheless, the inferred a_m's for our motifs vs the original partitions were highly correlated (see fig. 3.11). For a full description of these motifs, see appendix B.

In addition to the categories already discussed previously we also devised a new set of categories to test our newly discovered motif. Interestingly, the newly discovered motif had a substantially different repeat context than the previously discovered motif. Specifically, there was very little presence of this motif in all of the ALU and THE families, so these classes of repeat context were not considered relevant for determining the active motif burden of this motif. We therefore had a total of 12 new categories of motif to test for, consisting of perturbations of the discovered motif. Note that these motifs could overlap with previously identified ones, so we controlled for this to avoid double counting. For a list of the new motif categories and their calculated effect size see appendix B.

3.6.0.3 Optimizing

To fit this model we used scipy's optim.minimize function, parallelized across the motifs using the python multiprocessing module. We set the boundaries for the 3 parameters a_m, λ, θ at $(1E-4, 0.95), (1E-7, 0.01), (1E-7, 0.01)$ respectively to prevent underflow.

In addition, we used two other tricks aid in the optimization:

- Though taking the log likelihood makes the optimization easier, there are still some difficult to evaluate terms inside the logarithms that can't be simplified. Of principal difficulty is the coefficient in the negative binomial distribution used as the prior on motif counts. The issue is that the individual gamma functions

in this coefficient can easily diverge, even though the ratio is tractable. To solve this we use the identity

$$\frac{a}{b} = \exp(\ln a - \ln b) \tag{3.10}$$

in conjunction with scipy's gammaln function.

- The sum over all motif presences in the denominator to infinity was prematurely truncated at 50 terms. Heuristic experimentation indicated that this didn't impact the stability of the results substantially (data not shown).

3.6.0.4 Results

We ran the above optimization procedure on the previously discussed motif categories (41 representing the original motifs and 12 reflecting the new motif) to determine a_m in each case. The exact results of this analysis are presented for each motif category in B.

We then combined the a_m's across motif classes to determine the importance of the motif in determining hotspot location. To do this we first calculate the probabilities that a given hotspot j has an active motif in category m, q_m^j. By Bayes theorem this quantity is given by

$$q_m^j = \frac{a_m \frac{1-(1-p_m)^{h_m}}{\sum_l (1-(1-p_m)^{h_l}) p(h_l; w_i^c, \theta, \lambda)}}{a_m \frac{1-(1-p_m)^{h_m}}{\sum_l (1-(1-p_m)^{h_l}) p(h_l; w_i^c, \theta, \lambda)} + (1-a_m) \frac{(1-p_m)^{h_m}}{\sum_l (1-p_m)^{h_l} p(h_l; w_i^c, \theta, \lambda)}} \tag{3.11}$$

These values are then combined across all motif classes for each hotspot to give the probability that there is at least one active motif within that hotspot, under the assumption of independence. Then the probability of any hotspot containing at least one active motif is:

$$Q_j = 1 - \prod_m (1 - q_m^j) \tag{3.12}$$

So one can simply take the mean of this value to find the estimate of the fraction of hotspots that will contain at least one active motif. Note that we calculated this

value for a number of different partitions of the data: Firstly we calculated this for just the categories corresponding to the known motif, as outlined above, achieving an estimate of 38.4%, which is comparable but slightly lower than previously reported (41%), reflecting the slightly different datasets. Secondly, we just calculated this value for just the categories corresponding to the novel motif, yielding a value of 14.9%. Finally, we performed this sum over all 53 motif classes. This resulted in a value of 47.2%, which is a substantially higher than previously estimated. This indicates that this binding motif corresponds to an important pathway in hotspot localization, that was previously undiscovered on this dataset. A visual representation of these results is presented in figure 3.12.

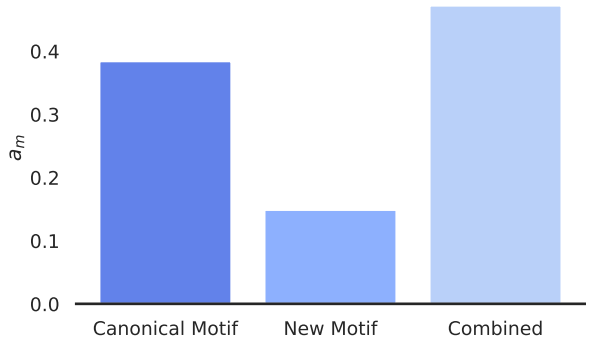

Figure 3.12: The calculated total a_m achieved by combining the individual attributions across all categories in both the canonical and novel motif. Note that these two categories sum to less than our overall estimate of the proportion of hotspots that have an active motif.

3.6.1 Retrofitting existing literature models

Thanks to Ryan Schenck who implemented the original 'Basset' as a rotation project with the group.

As a different test of the equivariant network's ability to improve results in this domain, we retrofitted two existing published approaches to incorporate this symmetry. The first of these models is Basset [Kelley et al., 2016], which attempts to predict ac-

cessibility from sequence using a convolutional network acting on sequences of length 1000. The second approach is DanQ[Quang and Xie, 2016], which also attempts to predict accessibility in addition to transcription factor binding and histone modification using a convolutional-LSTM topology.

For both of these models we made the following changes:

1. Required reverse-complement equivariance in any convolutional or bidirectional-LSTM layer

2. Replaced any dropout layers with equivariant MC dropout layers. At evaluation-time these are sampled 50 times.

Additionally, in DanQ we applied the reverse-complement max pooling before the final fully connected layer, guaranteeing global equivariance. For the Basset model, we did not do this as the pooling for the upper layers didn't evenly divide the sequence, making global equivariance impossible.

Ryan implemented a faithful version of the Basset model as per the paper, and observed it to achieve the stated accuracy results using the ADAM optimizer with lr=0.0001. The DanQ model required modification from the optimization regime discussed in the paper. Specifically, we could not recover the stated test loss when using the specified optimizer (rmsprop with lr=0.001, $\rho = 0.9$, ϵ=None, decay=0.0). We used the Adam optimizer, which has gained in popularity since the the paper was published) and performed a hyperparameter search over the ϵ parameter. This hyperparameter search was hierarchical: We first performed a search over a logarithmic space of $\epsilon = 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-}2$ and then a linear search between 10^{-4} and 10^{-3} running each trial for at most 3 epochs. We were able to recapitulate the stated loss with $\epsilon = 2e^{-4}$. All other Adam parameters were kept at their default values.

3.6.1.1 Results

The results here were mixed: For the Basset model, we observed an improvement in test loss on the hold out data, figure 3.13 left. For the DanQ model however, we did

not see a similar advantage to this methodology. We trained 3 topologies, one only replacing dropout with MC dropout (MC only), one performing only items 1 and 2 above (Partial Eq), and one performing items 1,2 and performing RC-pooling (Fully Eq). The results of these trials are presented in figure 3.13 right. We are unable to demonstrate an improvement in test accuracy for either of the equivariant topolgies, and only a minor improvement for the MC only. It is difficult to reason further about these small effect sizes, as it impractical to build convergence distributions here as these models take around 2 weeks to train to convergence.

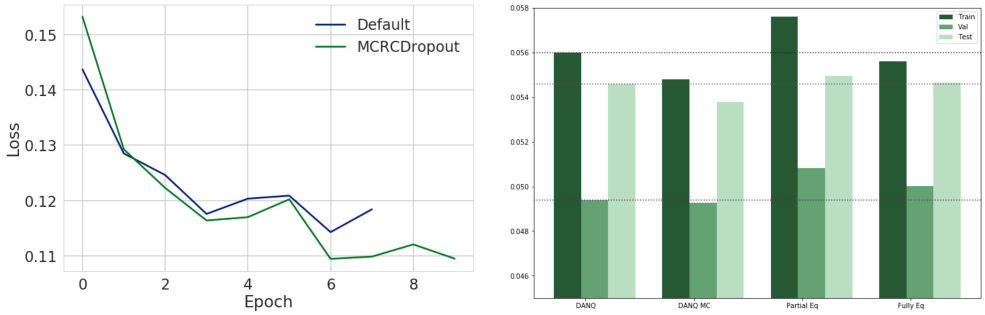

Figure 3.13: **Left** Test Loss calculated at the end of each epoch for our implementation of Basset, and the Basset modified with as per operations 1 and 2 above. **Right:** Train, validation and test loss values for the 4 DanQ models.

Chapter 4

Predicting Splice Site Locations from Genomic Sequence

4.1 Introduction

In this chapter, we consider the problem of modelling alternative splicing in humans using a novel deep learning approach that operates directly on genomic sequence. Here, we consider only splicing in bulk tissues and not stratified by tissue type, seeking to accurately model the core regulatory machinery that is common to all cells. We did however make substantial attempts to generalize this to tissue specific predictions. Our efforts, and the problems we faced, are briefly discussed in the 'further work' section.

First, we describe the approach at a high level and review the construction of the datasets used in this work. We discuss design considerations that lead us to make the decisions we have and give a detailed report of model selection and training. We then discuss the resulting models and the accuracy of the predictions, compared with recent competitive approaches, showing that we achieve state of the art performance at quantifying alternative splicing. We analyse the characteristic features of our approach and strengths and weaknesses in comparison to literature.

We then address our other aim: To determine to what degree we can interpret

our model to predict the effect of previously unseen mutations on human splicing. Here, we look not only at the conserved splice junction dinucleotides, but also more distal intronic mutation that is traditionally hard to interpret using bioinformatic techniques. We discuss the characteristics of mutations that we predict to have a large impact on splicing and show that they are biologically conserved and are under purifying selection in human populations, are enriched in disease populations, and validate directly on RNA-seq data, giving us confidence in the validity of our approach.

Finally, we discuss two areas that still pose a challenge in this field and our progress towards these solutions: The incompatibility of our predictions and traditional splicing QTLs, and the prediction of tissue-specific effects. We conclude with suggestions for further work in this project.

4.2 Predicting Splice Junction Locations and Quantifying Alternative Splicing

This section outlines the approach we take in order to predict and understand mRNA splicing. We present two similar approaches that both attempt to generate splicing prediction directly from genomic sequence and no other epigenetic input variables using adaptations on convolutional neural networks.

4.2.1 The Percent Spliced In ψ Distribution

To model splicing, it is important to first have a way to quantify transcript diversity. The ψ distribution described below, is such a way to do this and is our learning objective. This distribution was originally defined with respect to a particular transcript event [Venables et al., 2008, Katz et al., 2010] such as optional exon skipping, such that

$$\psi_i^{\text{exon}} = \frac{n_i}{n_i + n_i^{'}} \tag{4.1}$$

where n_i are the number of transcripts that support this event and $n_i^{'}$ and the number of transcripts that do not support it.

Whilst this is reasonable, we use a slightly modified definition as used in [Bretschneider et al., 2018] that does not define the transcript inclusion event type and merely calculates the distribution over all possible events for a given junction. Specifically, for each donor junction j with genomic position p_j we calculate the matrix

$$\psi_i^{(j)} = \frac{\#\text{isoforms supporting acceptor at position } p_i \text{ with donor at } p_j}{\#\text{isoforms supporting donor pos } p_j} \tag{4.2}$$

$\forall p_i \in [p_j + 1 \dots p_j + N + 1)$ for a user specified N. A schematic of this definition is presented in figure 4.1

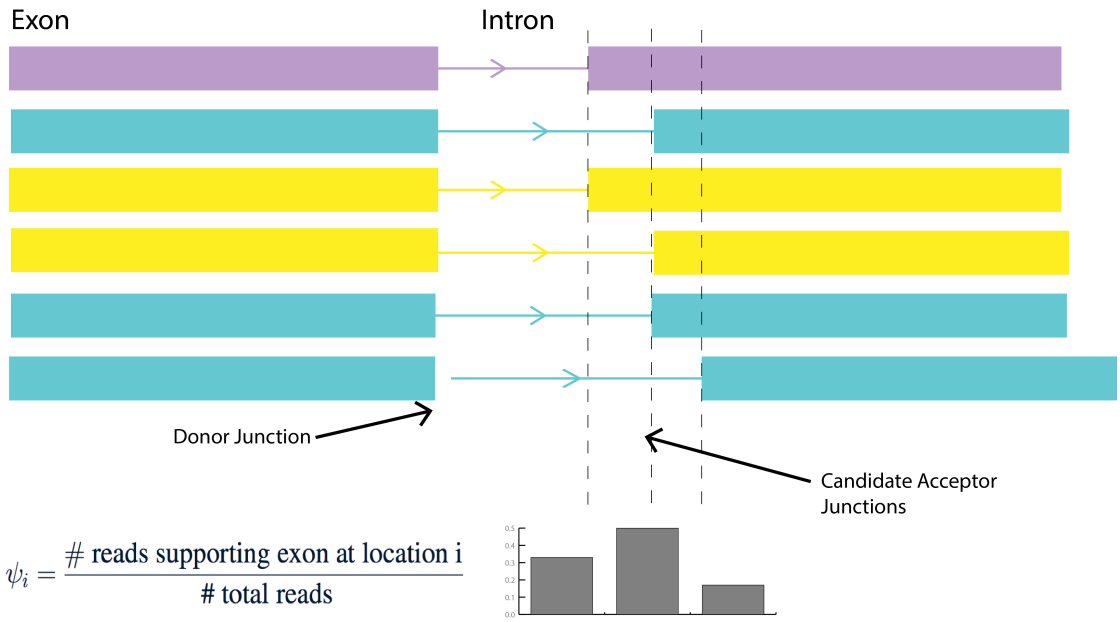

Figure 4.1: A typical calculation of the ψ distribution for a single donor junction

What this approach allows over those previous methods is the unbiased annotation over both exon skipping and alternative acceptor locations, with the ability to perform large scale annotations of the genome.

4.3 Training Dataset

The models are trained using a set of junctions derived from GTEx [GTEx Consortium and others, 2015]. This cohort contains RNA-seq data from 11688 unique samples, comprising 53 unique tissues types. For ease of comparison, we use the junction calls generated in [Bretschneider et al., 2018] which were created in the following way:

- FASTQ files were aligned with the HISAT2 aligner [Kim et al., 2015]
- For each donor junction site obtained from these alignments, scan for acceptor locations that are supported by at least 2 samples in at least 2 tissue types and obtain the number of reads seen across all tissues.
- For each read count, apply the Bayesian Bootstrap method in [Xiong et al.,

2016] to account for positional biases

From these junction calls and associated read counts, we construct the ψ distribution for each donor site by normalizing read counts for all acceptor positions associated with this donor. We require each observed acceptor to fall within 50kb of the donor, accounting for around 70% of the 170,000 total junctions in this dataset. The remaining 30% of donor distributions are discarded. Note that this procedure makes the assumption that splicing is guaranteed to happen in this window

For these ψ distributions we constructed two training datasets in order to test two different model topologies. For both, we partitioned these distributions into two groups by removing chromosomes 10, 20, 2 and X for testing, and keeping the remaining autosomes for training. From these test chromosomes, we removed all genes that had paralogs, defined by the ensemble paralog list, in training chromosomes to reduce the possibility of overfitting (though we experimented without this procedure and found only a very minor effect here, data not shown). For the training chromosomes, we performed a random partitioning with a split of (90%, 10%) into training and validation data used in the hyperparameter search.

From here, for training set 1 we stored 2 genomic sequences for each donor junction selected from the human reference genome hg19, ignoring any individual sample variation that may contribute to the distributions derived from GTEX data. One sequence of length 85bp centred around the donor junction was stored, in addition to a sequence of length 50kb starting at the donor junction and proceeding in the direction of transcription for each gene, such that there is an overlap between this sequence and the second half of the donor sequence. For dataset 2, we simply had a contiguous sequence of total length 57500bp, with an identical 50kb downstream of the donor and a 7500bp of upstream bases. Note that for both datasets, the strandedness of the gene was determined and the matching sequence stored.

4.4 Erhythroid RNA-seq

In addition to the above, we also generated our own dataset as an orthogonal validation of the model. This was constructed as follows:

Thanks to Damien Downes for performing the sequencing as described below. The following paragraph was written entirely by him to describe this procedure. Erythroid RNA-seq. CD34+ haematopoietic progenitor stem cells from three healthy donors (two males, one female) were isolated and differentiated into erythroid cells as described (Scott et al 2019, https://doi.org/10.1101/744367). 5x106 cells were fixed in 1 ml TRI-reagent (Sigma), snap frozen and stored at -80C for less than one year. RNA was extracted by addition of 0.1 ml 1-bromo-3-chloropropane, pipette mixing and separation in a Phase Lock gel Heavy tube (5Prime) and then precipitation with 1 μl of GlycoBlue and an equal volume (500 μl) isopropanol and centrifugation (10 min, 12,000 rcf, 4C). The RNA pellet was washed with 75% ethanol, resuspended in DEPC-treated water, and stored at -80C for less than one year. Total RNA was treated with Turbo DNAse (Invitrogen) at 25C for 60 min, then RNA was separated using phenol-chloroform isoamylalcohol and a PhaseLock Light-gel tube (5Prime). DNase treated RNA was precipitated at -80C overnight with sodium acetate, glycoblue, and 75 % ethanol, before centrifugation (12,000 rcf, 4C), 75 % ethanol wash and resuspension in DEPC-treated water. Globin and rRNA sequences were depleted from up to 5 μg of treated RNA using Globin-Zero Gold (Illumina), before PolyA selection with NEBNext Poly(A) mRNA Magnetic Isolation module (New England Biolabs), and indexing with NEBNext Ultra Directional RNA Library Prep Kit for Illumina (New England Biolabs) following the manufacturers' instructions with fragmentation to 180bp. RNA-seq libraries were quantified by qPCR (KAPA) prior to sequencing on the NextSeq platform (Illumina) with 150 bp paired-end reads (300-cycles total).

4.4.1 Ψ Distribution Construction

We used OLego [Wu et al., 2013] to map the reads for the 3 samples generated by the sequencing described above to our reference genome hg19. We used the same parameter for the allowed levenshtein distance (the number of single character events required to transform one string into another) between the reads and the reference as SpliceAI [Jaganathan et al., 2019] (-M 4). OLego operates entirely in a reference agnostic way using a regression model combining features such as local splice motif score and intron length to discriminate between ambiguous alignments. Due to this it is slower than some competitor aligners (such as STAR [Dobin et al., 2013]), but for our purposes this didn't pose an issue.

Upon generating alignment files, we then use the Leafcutter Cluster [Li et al., 2018] package to generate junction counts for the samples. We used the parameters (-m 1 -l 500000) for this, requiring 1 split read to support a junction and a maximum intron length of 0.5mb. Upon generating junction counts for each samples, we then combined them, and constructed Ψ distributions. We did this by filtering junctions that had donor site common to our previous dataset only, and normalized read counts for all acceptors corresponding to the same donor.

4.5 Empirical analysis of the training ψ distribution

We looked directly at the learning target ψ distribution, for the GTEX derived dataset in 4.3 stratified by tissue type, in order to understand the positional dependence of the acceptor junction locations (defined as all sites where the value of $\psi > 0.01$) relative to the splice junctions. Figure 4.2 top shows that there is a strong enrichment of of true splice acceptor junctions across all tissues within 10kb of the donor junction, representing the bulk of all introns. We then asked whether nearby acceptor junctions are favoured with respect to their donor i.e. for donor junctions that are spliced

# acceptor locations	# ψ distributions	Mean ψ first acceptor	Mean ψ not first acceptor	Ranksums p-value
2	12637	0.56	0.44	3.8e-123
3	1647	0.46	0.27	7.2e-37
4	197	0.35	0.21	0.019
5	10	0.83	0.044	1.2e-06

Table 4.1: Calculated median inclusion ratios for first putative acceptor vs subsequent acceptors

alternatively to 2 acceptor locations, do they prefer on average to be spliced to the closer junction? To answer this, we look at each junction individually and find that there is indeed a positional dependence within junctions (figure 4.2 bottom right). We found statistically significant evidence that the first putative acceptor junction location carries higher probability of being utilized (table 4.1). This is consistent with the 'window of opportunity' model for splicing (section 1.4.4.3), as this will be the acceptor site that is first visited by Pol II, which is a cofactor in spliceosome recruitment, yielding a greater splicing probability as splicing proceeds cotranscriptionally (see section 1.4.4.1). An outline of this is illustrated in figure 4.3.

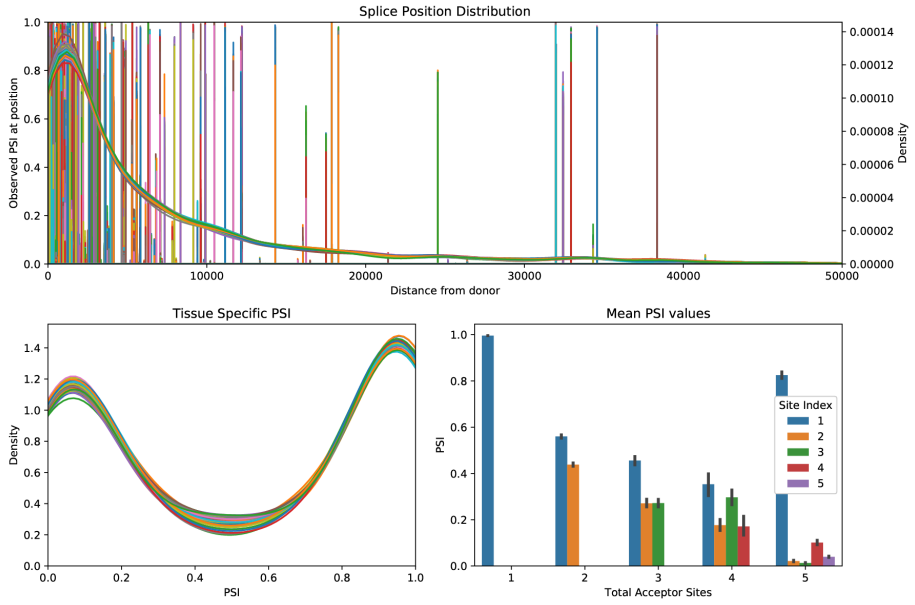

Figure 4.2: **Top:** For the training ψ distribution, a plot of the non-zero acceptor locations and their intensities relative to their respective donor positions. **Left:** For alternatively spliced junctions (defined as having more than 1 site with $\psi_i > 0.01$) we plot this distribution of nonzero ψ_i values. **Right:** We stratify junctions by number of putative acceptors and plot the mean value of each junction inclusion by index (sorted such that index 1 has position closest to the donor junction.) We find statistically significant inflation of mean inclusion rate for index 1 sites relative to sites that have index > 1.

4.6 Modelling Approaches

In this section, we provide details of the two models implemented to attempt to predict a ψ distribution from a section of genomic sequences taken from a gene. Both of these models rely heavily upon dilated convolutional layers [Yu and Koltun, 2015] to combine sequence context over large distances. For both of these models we use an ELU activation function after the convolutional layers.

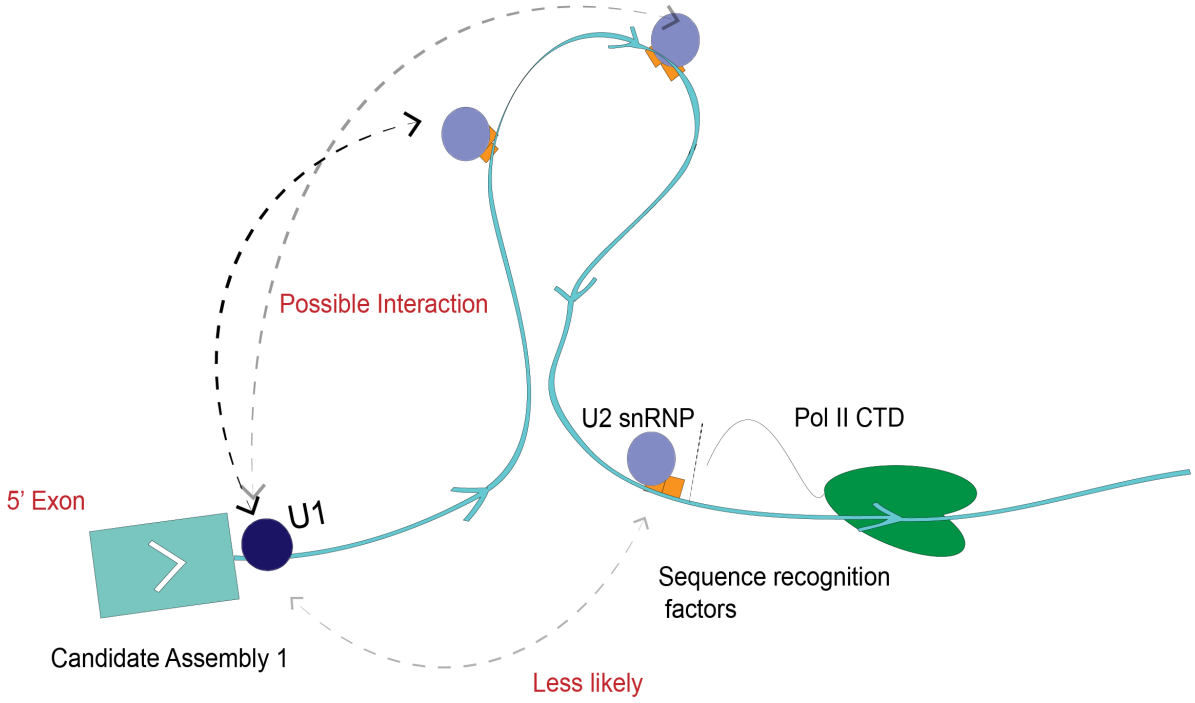

Figure 4.3: Shorter junctions have possible kinetic advantage

4.6.1 Model Design Approach 1: Conditional Donor

The first model topology is designed to attempt to model the interaction of the spliceosome across introns by having two distinct sequence inputs, one for the donor sequence and one for the acceptor sequence, and combining them.

- **Acceptor Sequence processing**: We process a contiguous 50kb sequence starting downstream of the donor junction, which is one hot encoded. We have an initial convolutional layer containing n_f filters ($n_f \in [40, 50, 100]$) of length $l_f \in [30, 40]$. We then have a batch-normalization layer, and a dropout layer with dropout probability $p < 0.25$. We then allow $n_c \in [6, 7, 8]$ dilated convolutional layers whose parameters are governed by a small number of meta parameters: Specifically, we define 'skew', $0.9 < s_k < 1.08$ and 'base', $b \in [50, 100]$ such that the number of convolutional filters in the i'th dilated convolutional layer is determined by $\text{int}(n_i = b \cdot s_k^i)$. The width of these layers is determined by a parameter $w \in [4, 6, 8, 10]$. The dilational rate is fixed at $d_i = 2^i$, and each

of these layers followed by an additional batch-normalization layer.

- **Donor Sequence processing**: We also input a separate 85bp of sequence around the donor junction that we processed using a series of up to 4 convolutional layers (governed by parameter la of filter length 24 for the first layer and 5 for every subsequent layer, with number of filters $na_i \in (8, 12, 24)$ chosen for each layer. Again, batch - normalization followed each of these layers. We then applied a global spatial max - pool, reducing the output to a $[1 \times na_{la}]$ vector.

- **Combining and predicting Ψ distribution**:We then combined the layers processing acceptor and donor sequence in the following way: At each spatial position, we concatenated the $[1 \times na_{la}]$ output from the donor sequence and the filters (note that this number is variable depending on the topology being trained) from the acceptor dilated convolutional layers, effectively copying the donor vector at every position. We then applied a 1 dimensional length 1 convolutional layer to this, with a choice of $r \in [5, 10, 20, 50]$ filters. We finally applied a convolutional layer with 1 filter and length $o \in [5, 10, 30]$, and then applied a softmax layer to the output to ensure the distribution is normalized.

4.6.2 Model Design Approach 2: Spatially Aware Convolutions

In addition to the above, we also tried a different approach where instead of using 2 separate inputs to deal with the acceptor and donor sequences individually, we modelled the entire junction in a single section and applied a fully convolutional model to this. In contrast to the approach above, instead of a 50kb sequence downstream of the donor site, and a 85bp window around it, we here used a stretch of sequence 50kb downstream and 7500bp upstream of the donor site.

4.6.2.1 Spatially aware convolutions

In order to generate a degree of positional awareness when using a fully convolutional network, we perform a slight modification to the convolution layer. Specifically, a typical convolutional layer acts on tensor X_{ij} to give

$$C(X)_{ij} = f\left[\sum_{m=1}^{N_{f-1}} \sum_{n=1}^{f_l} X_{m,j+n-1} W_{mni}\right] \tag{4.3}$$

with f_l the filter length in the layer and N_{f-1} is the number of filters in the previous layer (or the number of rows in the output of the previous layer in the case that the previous layer is not a convolutional layer). The adaptation we make to this layer is to make the weights themselves an explicit function of position. This is equivalent to making the weight tensor W_{mni} 4 dimensional and adjusting the above sum so the output becomes:

$$C'(X)_{ij} = f\left[\sum_{m=1}^{N_{f-1}} \sum_{n=1}^{f_l} X_{m,j+n-1} W_{mni,j+n-1}\right] \tag{4.4}$$

The obvious downside of this formulation is that the huge increase in number of parameters means that it offers nothing over a simple fully connected layer. What we do to get around this issue is constrain the way in which the tensor w_{mnij} can vary in the j'th dimension. Specifically, we allow the factorization

$$w_{mnij} = w^{'}_{mni} f(j) \tag{4.5}$$

with $f(j)$ a scalar function of position. For the function $f(j)$, we make the simple choice of $f(j) = e^{-\beta j} + c$, which has the distinct advantages of having a small number of parameters, having a constrained difference between neighbouring positions. These parameters were initialized $\beta = 0$, $c = 0$, so that they initially leave operation totally unchanged.

Using the above layer, we parameterized a model using the following procedure:

- **Input Layer** We input the stretch of contiguous sequence of length 57500 and have an initial convolutional layer that has a binary b_1 variable that allows the

optional inclusion of spatially dependent convolutions as described above. The number of filters are parameterized again ($n_f \in [40, 50, 100]$) with filter lengths $l_f \in [30, 40]$. We then have a batch-normalization layer.

- **Dilational Layers** As before we draw a skew variable s_k from $0.9 < s_k < 1.08$ and 'base', $b \in [50, 100]$ such that the number of convolutional filters in the i'th subsequent layers is given by $n_i = b \cdot s_k^i$. The width of these layers is determined by $w \in [4, 6, 8, 10]$. The dilational rate is fixed at $d_i = 2^i$, and each of these layers is again followed by an additional batch-normalization layer. We parameterize the number of these layers with $n_c \in [8, 9, 10]$

- **Clipping and output layers** The output length of each of the subsequent layers is 57500. We clip the first 7500 bases of this, and have a softmax function for the remaining 50000.

4.6.2.2 Relation to separable fully connected layers

Recently, the idea of 'separable fully connected layers' was proposed to address the issues associated with fully dense layers in genomics [Alexandari et al., 2017], specifically, the number of parameters and lack of biological relevance. This idea of the separable fully connected layer is to allow the more efficient encoding of spatial patterns for motifs amalgamated after convolutional layers and before fully connected layers. The authors originally conceived them to operate identically to a fully connected layer (without an activation function) where $FC(x) = \sum_j w_{ij} x_j$ except for the factorization of the tensor w_{ij} of shape (number of fully connected outputs n_o, number filters in previous convolutional layer n_c $\times$ length of previous convolutional layer l). Specifically, they define a 3 dimensional tensor s_{kij} as the outer product of two 2D tensors (i.e. $s_{kij} = u_{ki} \bigotimes v_{kj}$). Written explicitly, this gives

$$s_{kij} = \begin{bmatrix} u_{k1}v_{k1} & u_{k1}v_{k2} & u_{k1}v_{k3} & \dots & u_{kl}v_{kl} \\ u_{k2}v_{k1} & u_{k2}v_{k2} & u_{k2}v_{k3} & \dots & u_{k2}v_{kl} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ u_{kn_c}v_{k1} & u_{kn_c}v_{k2} & u_{kn_c}v_{k3} & \dots & u_{kn_c}v_{kl} \end{bmatrix} \tag{4.6}$$

The value of the weight tensor is then given by

$$w_{km} = \text{Flatten}(s_{kij})_m \tag{4.7}$$

where Flatten is an operation that converts the 2D tensor argument to a vector by concatenating rows, that we then take the m'th coordinate of. The approach here can be seen as somewhat analagous except designed to be implemented before convolutional layers for use in fully convolutional networks. If we imagine instead of matrix multiplication layer before a convolutional layer but instead an elementwise multiplication of a single slice s^k from above (that we will now refer to as $s_{ij} = u_i \bigotimes v_j$) then we have

$$C(s \circ x) = C'(x) \tag{4.8}$$

as per equation 4.4 with $w'_{mnij} = w_{mni}v_j u_m$. In the setup described above we actually let $u_m = 1$, and therefore have only the very simple modification of the weight tensor in 4.5. A criticism of this is that this schema only allows 1 possible spatial dependency, but empirically this is not a problem as the network is able to use this one positional dependence in conjunction with the non-linearity to learn unique spatial distributions for all filters (shown in 4.13).

4.6.3 Hyperparameters and Training

In order to find the best topology associated with each high-level approach we performed independent hyperparameter searches. We implemented parameterized versions of the model in keras and used the package hyperas. This is a wrapper around the popular hyperopt [Bergstra et al., 2013], which attempts to improve model optimization by making more sophisticated parameter choices than random (see introduction

for an more complete discussion). We used a uniform distribution over the parameter priors discussed below, except for categorical variables.

We ran a total of 50 trials for each topology, running over the training data only a total of 1 epoch (one single pass through the data) on each trial due to computational constraints. Each epoch took approximately 4 hours to run on an NVIDIA p100 graphics card. Upon recording the validation accuracy for each topology for each of the two approaches, we trained to convergence, using ADAM [Kingma and Ba, 2014], performing early stopping regularization with a patience parameter of 1.

The search space of parameters for model 1 is presented in table 4.2, and model 2 presented in table 4.3, and the final model schematics are given in appendix C.

Parameter Name	Description	Possible Values
n_f	Number of input filters for the acceptor sequence	$40, 50, 100$
l_f	Filter lengths for acceptor sequence convolutional layer	30,40
p	Dropout probability after the first convolutional layer	< 0.25
n_c	Number of dilated convolutional layers	$6, 7, 8$
s_k	'skew' used in calculating number of filters in dilational layers	$0.9 < s_k < 1.08$
b	'base' used in calculating number of filters in dilational layers	$50, 100$
w	width of filters in dilated convolutional layers	$4, 6, 8, 10$
r	Filters in penultimate convolutional layer	$5, 10, 20, 50$
o	Width of convolutional filters in output layer	$5, 10, 30$

Table 4.2: Hyperparamers for determining conditional donor topology

4.6.4 Final Model Results

We present the final train, validation and test accuracies and losses of the above models in tables 4.4 and 4.5 respectively. The performance is very high, with the fully convolutional model able to correctly identify the most probable peak in 83% of cases on holdout data. The split donor acceptor model also performs well although admittedly worse. For all subsequent analysis we will use the fully convolutional model unless otherwise stated.

Parameter Name	Description	Possible Values
n_f	Number of input filters for the acceptor sequence	$40, 50, 100$
l_f	Filter lengths for acceptor sequence convolutional layer	30,40
b_1	Allow the inclusion of spatial dependence in first layer	True, False
n_c	Number of dilated convolutional layers	$8, 9, 10$
s_k	'skew' used in calculating number of filters in dilational layers	$0.9 < s_k < 1.08$
b	'base' used in calculating number of filters in dilational layers	$50, 80$
w	width of filters in dilated convolutional layers	$4, 6, 8, 10$
o	Width of convolutional filters in output layer	$5, 10, 30$

Table 4.3: Hyperparamers for determining Spatially Aware Convolutions topology

Model	Train	Validation	Test
Split Donor-Acceptor	0.81	0.68	0.75
Fully Convolutional	0.86	0.84	0.83

Table 4.4: Train, Validation and Test accuracies for the two topologies

As a further test of robustness, we ran the model to attempt to predict the ψ distribution curated from the erhythroid data described above. This achieved a reasonable accuracy of 64% on the holdout data. Whilst this is lower than the data we used it is a reasonable results given:

- This is a distribution trained on bulk tissue being used to predict a single tissue
- We used a different pipeline and sequencing protocol to GTEX, which may come with specific biases
- There was lower coverage in this data than the model training data

We now turn our attention to a biological analysis of the resultant fully convolutional model.

Model	Train	Validation	Test
Split Donor-Acceptor	0.81	1.72	0.96
Fully Convolutional	0.60	0.68	0.69

Table 4.5: Train, Validation and Test Losses for the two topologies

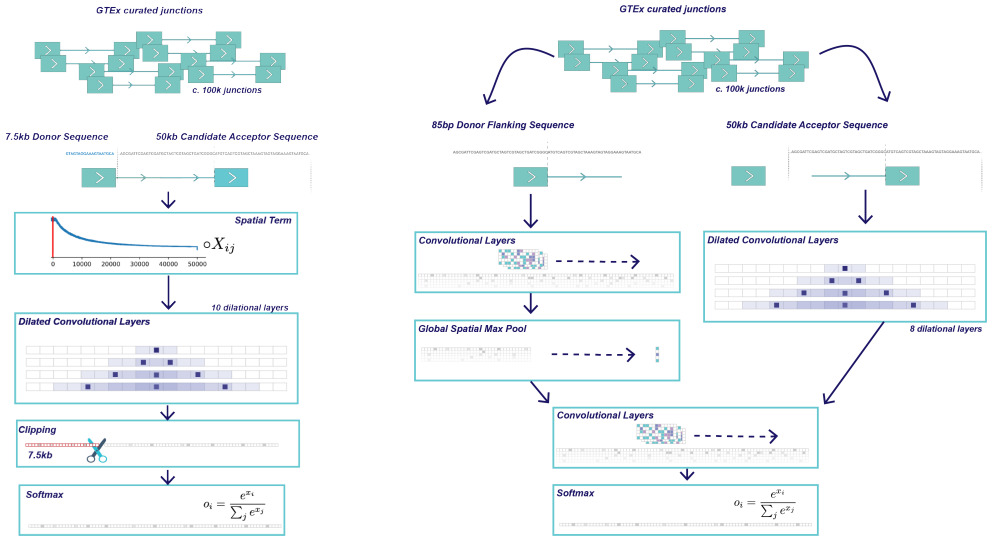

Figure 4.4: **Left:** Schematic of fully convolutional approach, **Right:** Schematic of split donor approach

4.7 Characteristics of Final Model

In this section we discuss the usage and characteristics of the final trained model as defined in 4.6.4, both to test the quality of its predictions against literature methods, and to evaluate how the model makes predictions by looking at 'saliency' and 'mutagenesis' scores it produces.

We discuss the exact forms of the saliency and mutagenesis scores in section 4.7.2, but for now it suffices to say that they both give a measure of the forecast splicing sensitivity that the model has to changes in a particular base, with the saliency scores used to annotate large scale genomic windows, and mutagenesis scores used for individual SNPs.

We performed a large scale saliency score annotation of the genome on the hold

out chromosomes to determine the biological relevance of this metric (section 4.7.3). We observed the expected saliency inflation at known splice junctions (figure 4.7). This is reasonable, as any mutation here is likely to have significant consequences.

Further to this, we investigated a broad distal inflation of saliency around the junctions. Particularly, this appeared to extend strongly into the surrounding exons (figure 4.7). This raises the question to what degree this model is learning otherwise meaningless coding sequence bias to define junctions as opposed to learning, for example, binding sites for splice enhancers or regulators. Admittedly, this is a difficult problem to address, as sequence content differences appear to have a significant effect on exon definition, and changing the GC content in introns surrounding splice sites has been shown to impact exon inclusion [Amit et al., 2012].

To test the robustness of this model in spliced RNA that isn't upstream of coding sequence, we use the set of noncoding RNA's (lincRNAs) published in [Jaganathan et al., 2019] to evaluate to what degree protein-coding sequence content drives prediction accuracy. Generating predictions on this collection of 1047 junctions reveals a drop in accuracy to 69%, which is still a robust result, especially as the positional dependence of our model is optimized for coding intron length, which has been observed to be shorter than lincRNA intron length [Derrien et al., 2012].

We found strong evidence that the model was implicitly searching for the next exon (section 4.7.4) as a whole as opposed to just the acceptor site. This is somewhat expected, as [Berget, 1995] noted that exon definition is at least partially determined by the pairing and relative location of an upstream acceptor site and its downstream donor site. It appears that the model finds this information particularly useful in determining exon inclusion ratio, and we found strong evidence that the model was more sensitive to destruction of the downstream donor site than it was to destruction of either upstream or downstream controls (figure 4.8 top right). This is a gratifying result that the model appears to recapitulate purely from data the phenomenon reported in [Talerico and Berget, 1990] that the rate of intron removal was lowered by a factor of 20 by the destruction of the downstream donor site. Indeed, in our in-silico

experiments, we see acceptor junctions that had $> 95\%$ splicing probability according to the model drop to $< 1\%$ when the downstream donor was destroyed (data not shown). We found that this effect was dependent on the length of the subsequent exon and that over 300 base pairs this was negligible (figure 4.8 bottom left).

Performing a direct interrogation of the model weights in the first layer, yielded matches to known RNA binding proteins (section 4.7.6), indicating that the model has learned some of the complex regulatory mechanisms that govern this process. We further observe that these mechanisms have varying degrees of spatial significance (section 4.7.7), indicating it has learned a tradeoff between local sequence optimality vs probabilistic inclusion ratios as a function of distance.

We finally compared our model to the literature models spliceAI [Jaganathan et al., 2019] and MaxEntScan [Yeo and Burge, 2004], to compare the extent to which we correctly predict the ψ distribution and demonstrate state-of-the-art performance at this task (section 4.7.8).

4.7.1 Typical Model Usage

We embed this model within a command line utility to allow its use, both at predicting junction inclusion ratios at junction sites, but also in providing annotation as to which bases in the genome are important in determining splice junction location. Specifically, the functionality allows us to:

- Produce wig files containing junction inclusion ratios for all junctions overlapping a user specified genomic locus.
- Produce wig files containing 'saliency scores' for junctions overlapping a user genomic site.
- Provide 'mutagenesis scores' for a list of user specified SNPs.

An example of the usage of the tool is presented in figure 4.5, with a schematic analysis of exons 5 and 6 of the CSRP2BP on chromosome 20. We see that the

model correctly identifies the two well-known acceptor sites for exon 5, and only the constitutive junction for exon 6. One can also see the saliency scores plotted below these predictions, allowing an interpretation of the model's output. Here, we see high saliency surrounding not only the canonical binding motifs and known splice altering SNP rs 80113248, but also more distally, indicating that the model is picking up a degree of more sophisticated splicing regulation.

Figure 4.5: Model provides base-sensitive splicing positions for genomic regions of up to 50kb and reports base saliency scores, which illustrated the importance of each base in making the final prediction.

4.7.2 Saliency Scores vs Mutagenesis Scores

We use two distinct methods to calculate the saliency and mutagenesis scores due to computational considerations. Saliency scores, which are used to annotate larger stretches of the genome, are calculated as the gradient of the junction-wise cost function with respect to the input. Specifically,

$$S_{ij} = \sum_{k=1}^{4} |\frac{\partial}{\partial x_{ik}} C(y^j, \hat{y}^j)| \tag{4.9}$$

where i is the position relative to the start of the overlapping sequence (in this case 7500 base pairs before the donor site), and j is the index of the junction of that overlaps the given genomic site, and C is the cross-entropy loss function used to fit this model. The vectors y^j and $\hat{y}^j$ are chosen to be the following: y^j is the default prediction of the ψ distribution for this junction under the trained model and is treated as a constant. $\hat{y}^j$ is the output of the model that is being differentiated. Consequently, under this scheme, if the differential of the base doesn't change the output then $S_{ij} = 0$. Note that we sum over j as x is a one-hot encoded vector of 4 bases.

Note that this defines a saliency score for each base for a particular junction. Whilst this is useful for interpretation, when looking to provide large scale analysis of the genome it can be cumbersome. Consequently, in the analysis below, we calculate scores in absolute genomic positions. For forward strand junctions (reverse strand junctions are defined analogously) if $O_p = [p_1, p_2...]$ is the set of all junction start locations (defined as 7500 base pairs 5' of the donor site) overlapping at position p and

$$S'_p = \max_{k=1,2..|O_p|} S_{p-p_k,k} \tag{4.10}$$

For the mutagenesis scores, used to provide information about how SNPs from a list (of length much less than the genome length) alter splicing, we use a different approach. Specifically, for each of the j overlapping splice junctions, we calculate the output ψ distribution (given again by $\hat{y}^j$) for both the reference and given alternative

allele. Let $\hat{y}^j_{\text{ref}}$ and $\hat{y}^j_{\text{ref}}$ be the predicted ψ distributions for the model corresponding to the alt and reference alleles for the given SNP and junction j. For this SNP, we then have the 'mutagenesis score'

$$M_{\text{SNP}} = \max_j |C(\hat{y}^j_{\text{ref}}, \hat{y}^j_{\text{alt}})| \tag{4.11}$$

with C a distance measure between the two distributions. We experimented with a number of forms of C (data not shown), and chose C to be the KL-divergence between these distributions.

The reason for using these two evaluation functions is purely computational. In an ideal world, we would use the mutagenesis scores only, as the saliency represents only infinitesimal changes and does not correspond to a genuine genomic sequence. For large scale annotation, the mutagenesis scores in this form are unfortunately unfeasible to calculate, as for n scores we need $\approx (N \times$ mean overlapping junctions) evaluations of the network to calculate them. We experimented with the correlation between these scores empirically and found a correlation between the two, but the saliency scores were qualitatively 'blurred' around salient bases and were less suited to detecting specific splice altering bases (for an example, see figure 4.6).

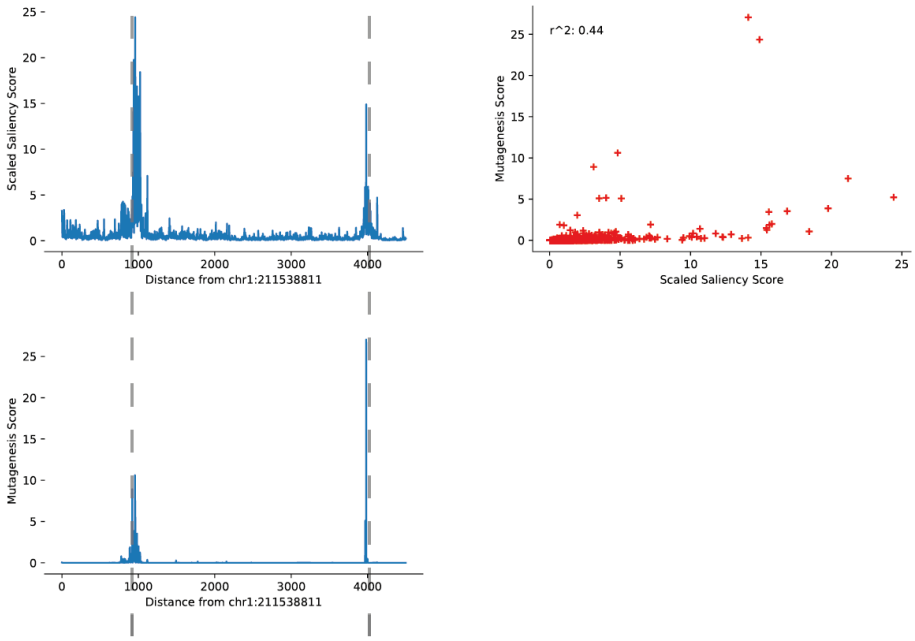

Figure 4.6: Comparison of saliency and mutagenesis scores for donor junction at position 211538811. We observe this effect of 'blurring in the saliency scores, making them less useful for fine-scale interpretation. **Right:** Correlation between these scores for this genomic interval

4.7.2.1 Mutagenesis Querying System

In order to generate mutagenesis scores for a set of query SNPs, we use the following approach: Construct an index for the sequence data stored in hdf5 format. The data structure used for this is the interval tree. This is a ubiquitous structure in genomics, and allows positional querying in $O(\log(N))$ time, with N the number of intervals stored (in this case splice junctions). We build a unique tree for each contig, and store intervals parameterized by donor location, final acceptor location (+ padding), and store an index into our sequence table as metadata to avoid replicating this. We further store the strandedness for each sequence. This index is cached after its initial construction using python cpickle, as construction is an $O(n \log n)$ operation. For

each query SNP, we then use this index to return all overlapping junctions, allowing us to calculate mutagenesis scores as described in 4.7.2.

4.7.3 Saliency Scores are Increased at Known Splice-Sites and in Exons

We inspected the degree to which saliency are enriched around known splice junctions. To this end, we used locations of known donor and acceptor junctions in GENCODE V24lift37 and calculated the mean scores bulked across all junctions. The results presented in figure 4.7, illustrate the strong enrichment in the two canonical splice bases for both donor an acceptor junctions, which is sensible, as a loss of these bases is likely to be catastrophic for splicing.

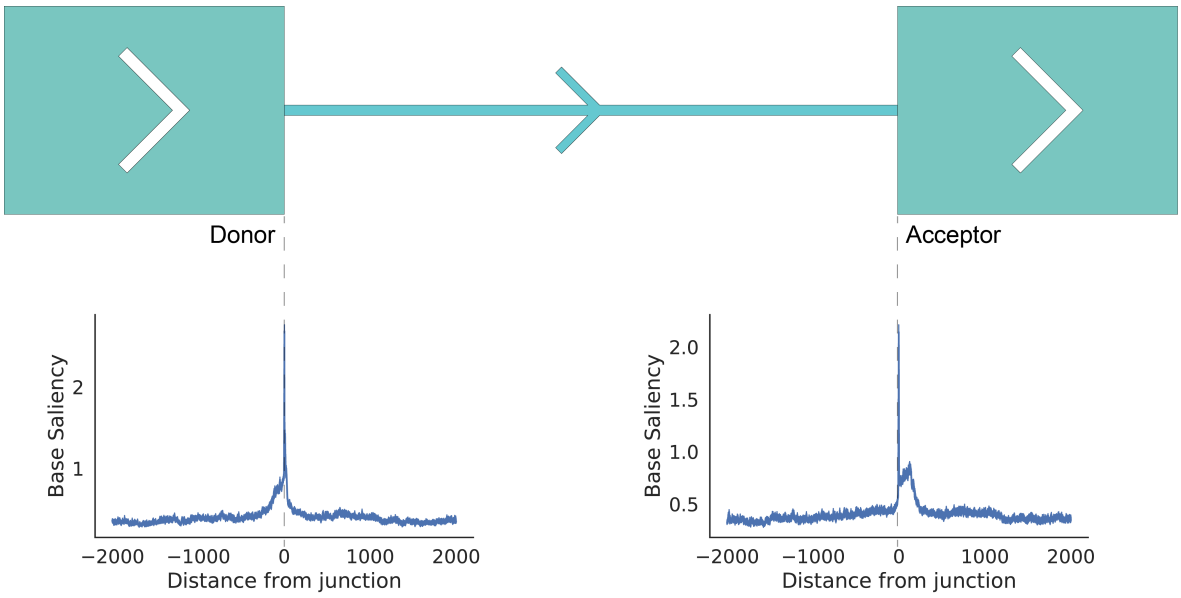

Figure 4.7: Mean base saliency for splice junctions defined in GENCODE V24lift37 saw a substantial spike around known acceptor and donor sites, with a broad enrichment around 150bp into the surrounding exons.

4.7.4 The Implicit Modelling of Exon Definition

We investigated the phenomenon that when trying to predict an inclusion ratio for a specific acceptor, it appeared to ascribe substantial weight to the next downstream donor and thus appeared to be modelling the entirety of the next exon, despite this not being part of the training data. Figure 4.8(top left) illustrates this common situation in the MYLK gene. For donor junction at position 12367817, we try to find its corresponding acceptor location, putting a high probability on location 12366275, which corresponds with GENCODE annotations. Upon inspecting the local basewise saliency we see, as expected, a large peak at and around this location, including the canonical splice acceptor site. Further to this, we also see a strong saliency peak at location 123366070, which is the annotated exon end location.

We then established that this dependence on a downstream donor was a statistically significant effect. For our test data, we located for each junction the location of the principal acceptor location. For locations that were also tagged in GENCODE V24lift37, we then used this database to find the location of the subsequent donor site and discarded any examples where this donor was not contained within the test sequence window. Here, we performed three distinct transformations on three independent copies of the sequence:

- We randomly permuted the 10 bases within the 5 bp of the downstream donor site. This leaves sequence x_p.
- We randomly permuted the 10 bases ending 5bp upstream of the downstream donor site. This leaves sequence x_{up}
- We random permuted the 10 bases beginning 5bp downstream of the downstream donor site. x_{dp}

Upon doing this, we were left with 4 similar sequences (including the original): $x, x_p, x_{up}x_{dp}$ that we generated 4 predictions with the model $(\hat{y^j}, \hat{y}^j_p, \hat{y}^j_{up}, \hat{y}^j_{dp})$ for each junction j. We then calculate 3 junction-wise mutagenesis scores similar to those in

equation 4.11:

$$M_j^p = C(\hat{y}^j, \hat{y}_p^j) \tag{4.12}$$

$$M_j^{up} = C(\hat{y}^j, \hat{y}_{up}^j) \tag{4.13}$$

$$M_j^{dp} = C(\hat{y}^j, \hat{y}_{dp}^j) \tag{4.14}$$

We plot the distribution of these scores in figure 4.8(top right) and find a significantly greater mutagenesis score for the case that the bases are permuted within the subsequent downstream site than either of the two controls.

We then further looked to see to what extent the length of this exon determined the potency of this phenomenon. We calculate a 'damage score' representing the loss in predicted splicing probability occurs during the permutation of these bases:

$$d_j = \hat{y}^j \left[\text{argmax } y^j\right] - \hat{y}_p^j \left[\text{argmax } y^j\right] \tag{4.15}$$

This analysis is presented in figure 4.8(bottom left) and illustrates that there is a larger effect for smaller exons; Junctions over 300bp receiving little change in predicted inclusion when their downstream donor is destroyed. This could be seen as evidence that the stabilizing reaction between the downstream U1 and upstream spliceosome elements can only operate at distances less than this.

Performing a somewhat opposite procedure, we added a copy of each of these downstream donor junctions exactly 150bp downstream of the acceptor junction location, with the expectation that this would increase the probability of splicing in longer exons that were predicted to splice with less than probability 1 (figure 4.8(bottom right)). Whilst this appeared to happen slightly, as evidenced by the increased presence of negative damage scores in the bottom right of this figure, this proved to be a weak effect. This indicates that the presence of this 10bp downstream motif is not enough, and it has to be surrounded by the correct sequence context to improve the probability of splicing to the target acceptor site.

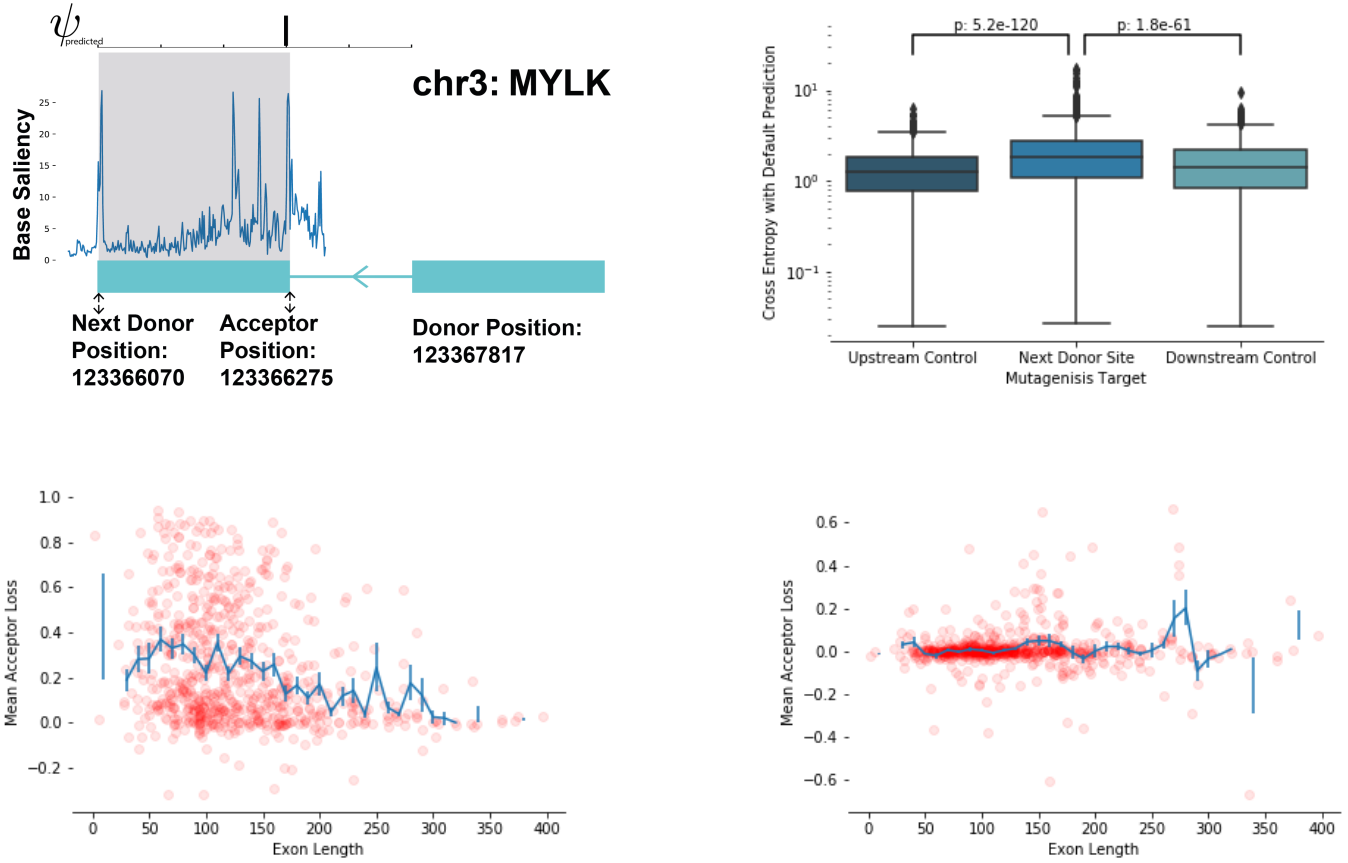

Figure 4.8: **Top Left:** A representative example of the phenomenon of downstream donor importance. For exon at chr3:12336275 we see saliency peaks not only at acceptor but also at subsequent donor location. **Top Right:** Junction-wise permutation mutagenesis scores are enriched in downstream exon relative to control sequences. **Bottom Left:** Acceptor loss under permutation as calculated in 4.15 plotted as function of next exon length. **Bottom Right:** Acceptor loss under addition of downstream donor motif 150bp from acceptor plotted as function of next exon length.

4.7.5 The Output ψ distribution

We evaluated the predictive distribution on test chromosomes and found it to be well calibrated. Figure 4.9 (left) shows the distribution of true ψ values for three stratifications of predicted acceptor sites into low confidence ($0.01 <$ predicted inclusion value < 0.1), medium confidence ($0.1 <$ predicted inclusion value < 0.9) and high confidence $0.9 <$ predicted inclusion value).

What we did see (figure 4.9 right) was a slight under-reporting of very high predicted ψ values relative to the null, which is explained by the fact that the model is not able to perfectly disregard sites that appear to be locally optimal.

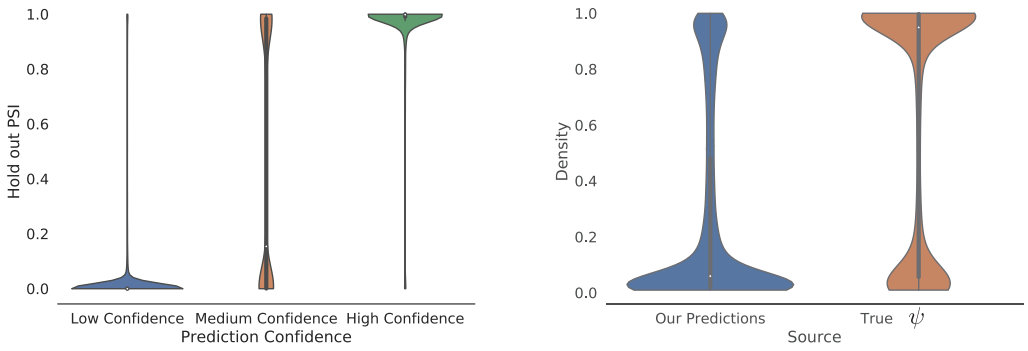

Figure 4.9: **Left:**. True inclusion ratios on test data for sites the model predicts to be low confidence acceptors, medium confidence acceptor and high confidence acceptors. **Right:** Comparison of the predicted inclusion ratio distribution vs the true inclusion ratio distribution.

4.7.6 Biologically Salient Motifs

From the trained model, we analysed the weights to understand what sequence features are important in making splicing decisions. We looked at the weights only in the first convolutional layer and performed the following procedure (similar to that used in [Alipanahi et al., 2015a]) to convert them into sequence logos. Firstly, we created a new model using only the first convolutional layer of the trained model, in-

cluding the trained weights. The output of this model is a tensor of shape $[50, 57500]$ (representing the number of filters and the length of the input sequence). On test data, for each of the 50 filters, we find the sequence corresponding to the maximum activation as long as that activation is greater than the elu(0) (our choice of activation function at zero). We used these sequences to build classical sequence logos for further querying. The results of this procedure are displayed in figure 4.11.

We then used TOMTOM [Gupta et al., 2007] to evaluate these motifs in the context of known literature. We searched the database RNAcompete [Ray et al., 2013] to search for sequence logos of known RNA-binding proteins. The manual inspection of these motifs reveals a large degeneracy between them (especially the donor and acceptor sequences), but we nevertheless returned a large number of matches against motifs of well-known splice impacting proteins.

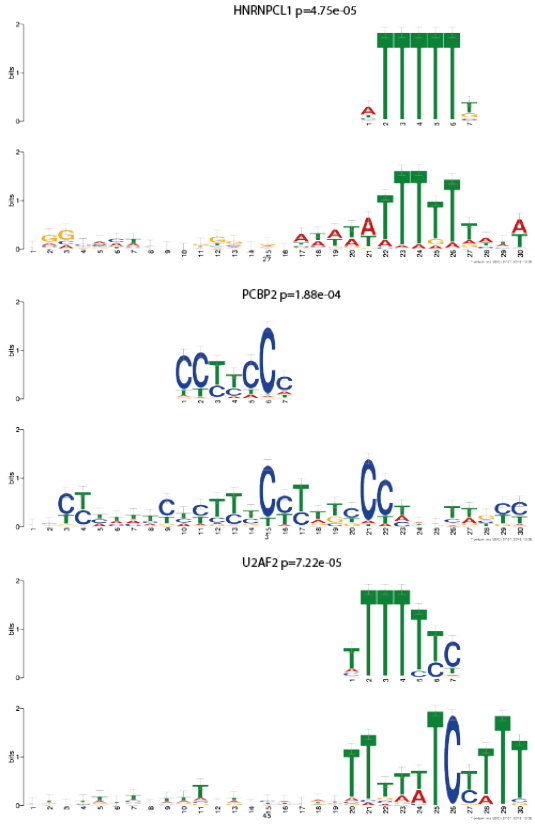

Figure 4.10: A selection of representative motifs generated from the model weights by the procedure above and their matches in against the RNAcompete [Ray et al., 2013] RNA binding protein database using TOMTOM [Gupta et al., 2007].

Figure 4.10 shows 3 illustrative matches of the learned model weights to known motifs, corresponding to 3 known RNA binding proteins hnRNPCL1, PCBP2 and U2AF. U2AF is a well-known part of the minor spliceosome that binds in a sequence-specific manner in the early phase of spliceosome recruitment at the acceptor junction. Further, PCBP2 is highly homologous to known splice factor PCBP1 and has recently been indicated as an oncogenic splice factor itself [Guo and Jia, 2019]. We saw further matches to other known splicing factors with a full list of the TOMTOM matches in appendix D.

Qualitatively we find these motifs similar to those reported in COSSMO [Bretschneider et al., 2018], with the difference being that they were somewhat longer on average, with slightly less significant p-values to these known RNA binding proteins. A pos-

sible reason for this derives from the different aims of the two methods. COSSMO, despite using a very similar set of peaks attempts only to discriminate between a small number (typically 50) possible splice junctions based predominantly on the local sequence structure of them. Consequently, it can not learn the importance of large scale sequence context and therefore does not adapt to do so. This makes the interpretation of our network more challenging, as it tries to balance the recognition of large scale context with the recognition of sequence-specific factors.

4.7.7 Motif Activations

For each of the motifs generated in section 4.7.6, we plotted the mean activations of these surrounding the donor site for each distribution (figures 4.12, 4.13). Note that these motif activations correspond to the same motifs as each other and as those in figure 4.11. Looking locally, we see some interesting characteristics of these activations. Specifically, a number of them appear to be searching for GC content which facilities genome twisting and thus impacts nucleosome occupancy [Olson et al., 1998], which is known to be correlated exon location [Schwartz et al., 2009]. In (row, column notation), these are activations (1,1) and (9,3) in figure 4.12. Interestingly, although motif (1,2) has a large quantity of AT content, there is minimal recognition of the difference in exonic vs intronic sequence content.

Figure 4.11: Filters learned by the model in layer 1

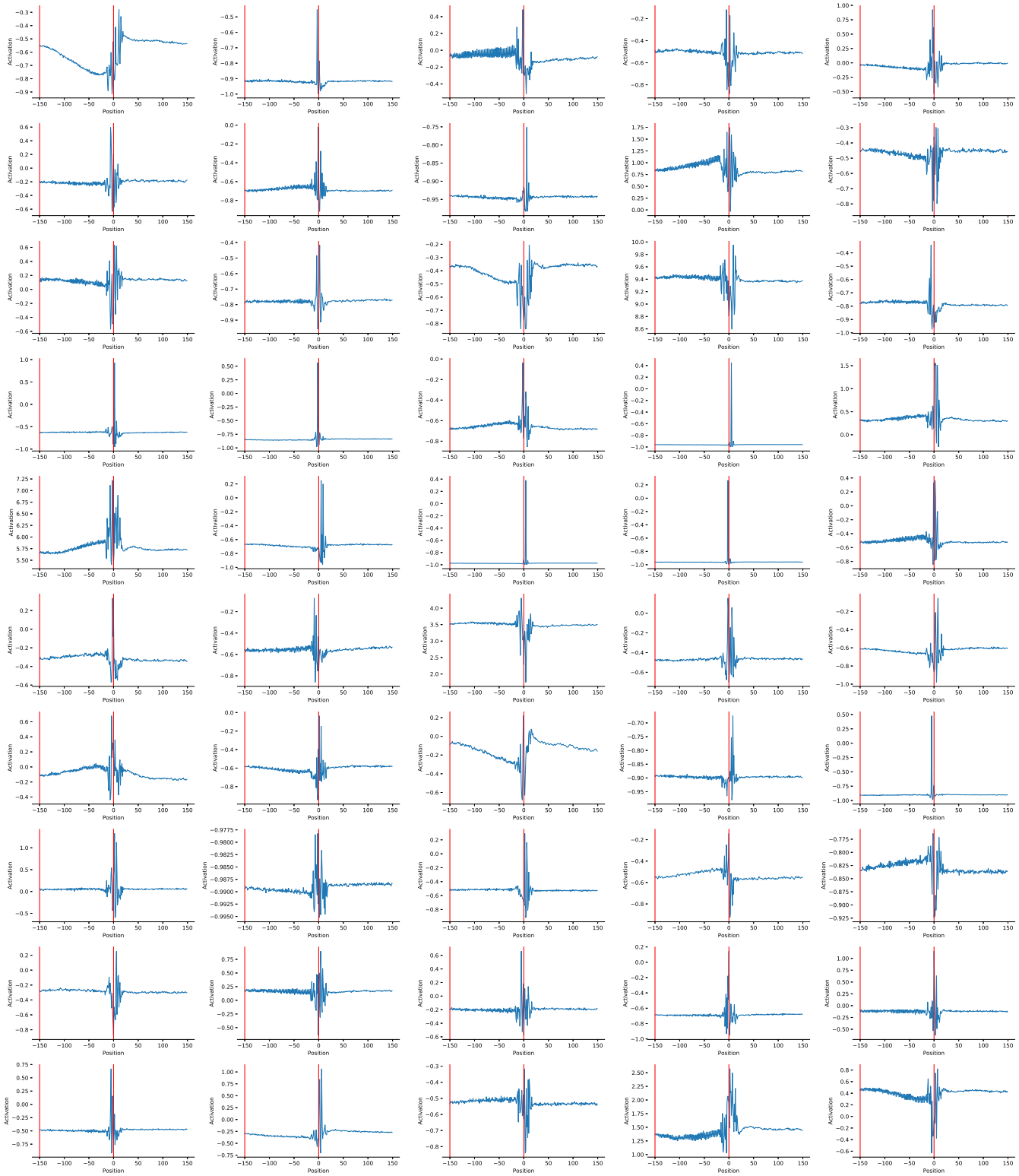

Figure 4.12: Mean activation by position for filters in 4.11 centred around donor site

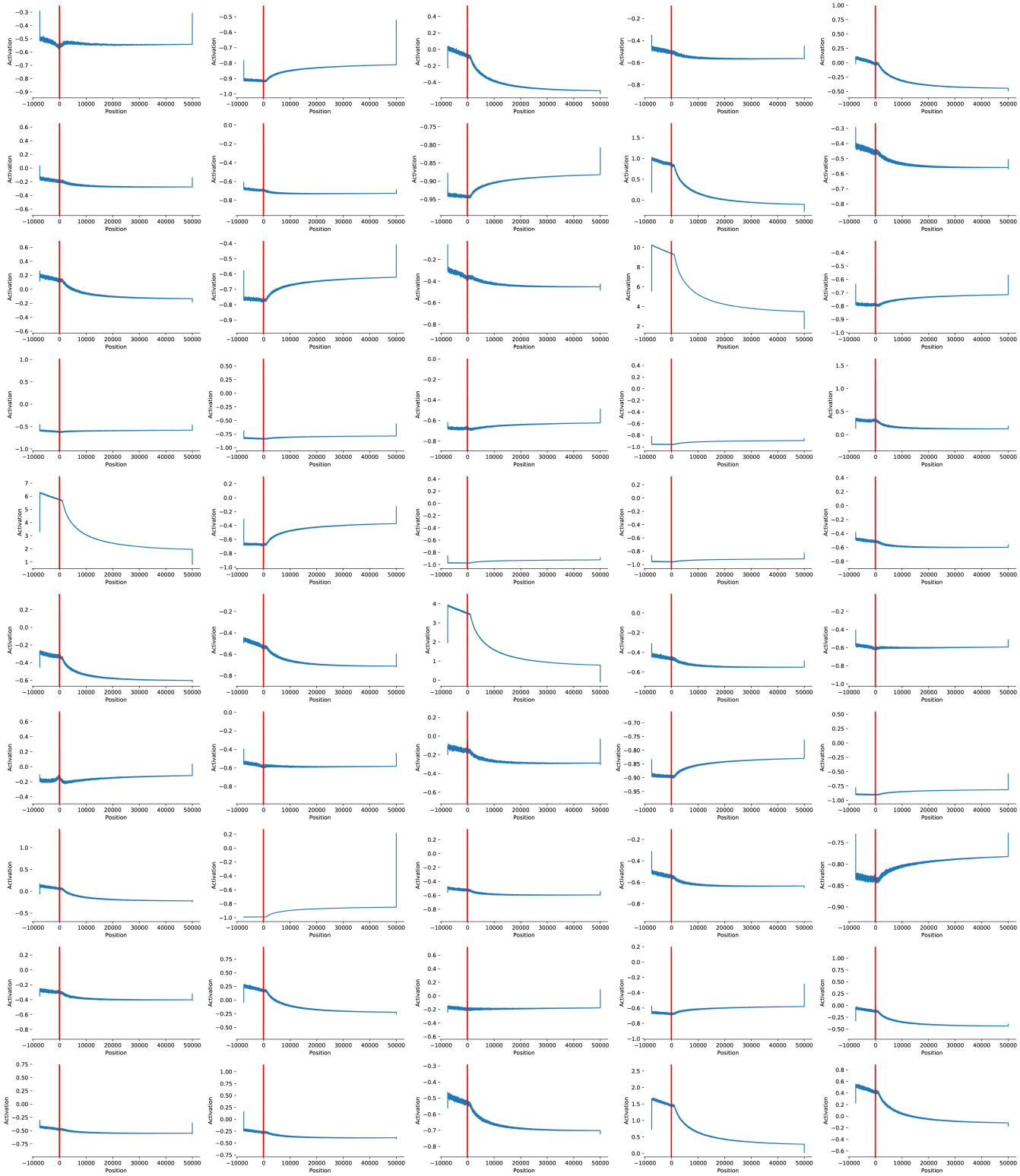

Figure 4.13: Mean activation by position for filters in 4.11 centred around donor site observed at large scale show spatial dependence is different for different motifs.

Looking at the large scale activations in figure 4.13, we see immediately the impact of adding the spatial relationship to the convolutional filter in the model: There is now a clear spatial importance to the motifs, that allows the model to reason about relative importance of seeing varying motifs at different distances from the donor splice junction. One interesting feature about these activations is how they yield a heterogeneous spatial dependence. This implies that certain motifs have been learned to have a very strong spatial dependence ((5,1), (5,2) and (6,3)) whereas others are much less impacted (4,4). This is difficult to interpret more directly, as not all of these motifs have convincing matches in 4.7.6, but it appears that the model is attempting to balance less optimal sequence configurations for splicing with knowledge about typical intron lengths. For examples, motif (4,4) has 5 high information content bases so presumably yields an important splice signal regardless of distance from donor junction, whereas there are a number of possible matches to ((5,1), (5,2) and (6,3)), so sub-optimal motif matches could be permitted if they are favoured spatially.

4.7.8 Comparison with Literature Methods

We performed a series of tests to compare the results of our model to existing tools in a series of splice prediction tasks. The tools that we chose to benchmark were MaxEntScan, a ubiquitous splice-prediction tool that takes into account short sequence context, and SpliceAI, a state of the art deep learning tool published recently.

Both of these tools attempt to answer a slightly different question to us in that they attempt to flag up all splice junctions, as opposed to just acceptor - donor pairs as we do. Consequently, it is necessary to transform the output of these tools into a comparable format to compare them. We perform the following steps for each of the tools listed.

4.7.8.1 MaxEntScan preparation

MaxentScan takes a sequence context of 23 base pairs around acceptor junctions, with 20bp of that upstream of the junction, and the remaining 3bp in the downstream exon. To generate predictions for every position, we, therefore, split each of our benchmarking examples up into overlapping windows starting from the donor position until we reach the end of the junction, and predict for each of these separately. In our 13,010 test junctions, this procedure yields a total of 140,098,849 MaxEntScan predictions. Upon, arriving at these predictions, we then transform them such that each base has an independent probability of being a splice site. This procedure takes the MaxEntScan outputs o_i where:

$$o_i = \log_2 \left(\frac{p(\text{splice site})}{p(\text{not a splice site})} \right) \tag{4.16}$$

and so we set

$$o_i' = \frac{2^{o_i}}{2^{o_i} + 1} = p(\text{splice site}) \tag{4.17}$$

To assure that all junctions are the same length, we then pad the remaining bases corresponding to padding with small positive $\epsilon = 1e^{-8}$.

4.7.8.2 SpliceAI preparation

SpliceAI in its final form is an ensemble method: The authors trained 5 models with identical topology, and average predictions from them at inference time. For a fair comparison with our method, we simply select one of these individually trained models (spliceai_1 in the GitHub repository) and compare this with our approach. The output of SpliceAI is already formatted to give the probability that each base is either a splice donor, splice acceptor, or a null site, so we can simply apply the transformation outlined in the following section to the predicted splice acceptor sites.

We note that there is overlap between our holdout chromosomes and SpliceAI's training chromosomes and as such, there is the prospect of overfitting (though of course SpliceAI wasn't trained to predict the ψ distribution so much as flag all pu-

tative splice sites. It is possible that these results therefore slightly favour SpliceAI, although the authors did note that they didn't notice any substantial overfitting.

4.7.8.3 Fitting output to softmax simplex

Once we have arrived at the unconditional base-wise splice site probabilities for each of the desired methods, we then transform them to a simplex by defining a model for each that is a single parameterized softmax layer, with scaling parameter b_{scale}. Then, for each of the two approaches above, the matrix o'_{ij} is probability of a splice site at position j in junction i the normalized splice distribution is

$$\psi_{ij}^{\text{Model}} = \frac{\exp(b^{\text{Model}} o_{ij})}{\sum_j \exp(b^{\text{Model}} o_{ij})} \tag{4.18}$$

We then fit this model for parameter b for each model, attempting to minimize the crossentropy loss between the predictions and the true ψ on the holdout data. Note that in effect we are overfitting this model here, but as there is only 1 parameter and this just serves to benefit the literature models this is highly unlikely to effect the conclusions drawn from this analysis. This inference procedure was performed using the scipy-optimimize-minimize function in python using an initial value of $b = 1$ and bounds $[-100, 100]$

Upon obtaining these distributions for each of the methods, we computed cross entropy loss with our splicing derived truth distribution for our holdout data. We stratified the predictions twofold:

- Junction length: Create 3 distinct classes of acceptor junctions: Short ($<$ 1000bp), Medium ($1000 - 7500$bp) and Long (7500bp +).
- Number of true acceptor sites: 1, 2, > 2

The results of this calculation are presented in table 4.6

A criticism of this approach that could explain the relative underperformance of SpliceAI, which has been demonstrated to be the current state of the art tool for splice site identification, is that SpliceAI could be predicting a large number of

Algorithm	Acceptor Sites	$l < 1000$ (Accuracy, Loss)	$1000 \leq l < 7500$ (Accuracy, Loss)	$7500 \leq l$ (Accuracy, Loss)
MaxEntScan	1	0.26, 3.10	0.19, 3.81	0.14, 4.80
SpliceAI	1	0.61,1.77	0.64, 1.72	0.68, 1.63
OurMethod	1	**0.92, 0.30**	**0.91, 0.36**	**0.79, 0.87**
MaxEntScan	2	0.17, 3.98	0.14, 4.25	0.08, 4.87
SpliceAI	2	0.43, 3.42	0.44, 3.01	0.46, 2.92
Our Method	2	**0.81, 0.87**	**0.79, 0.85**	**0.73, 1.04**
MaxEntScan	> 2	0.13, 4.88	0.12, 4.72	0.07, 5.44
SpliceAI	> 2	0.34, 5.63	0.31, 5.11	0.29, 5.25
OurMethod	> 2	**0.58, 1.74**	**0.64, 1.65**	**0.54, 2.07**

Table 4.6: For the test data stratified by number of acceptor sites with probability > 0.01 and total junction length (l) we calculated loss scores for each of the three models, with distributions prepared as described. The pairs reported in the table are (probability of predicting the most probable peak, cross entropy loss between predicted ψ distribution and true ψ distribution.)

additional candidate deep intronic splice sites, and is getting penalized for doing so. This doesn't appear to be the case, as after cataloguing the sites predicted by our model and SpliceAI (> 0.01 inclusion value) to be junctions, we observed comparable numbers (in fact spliceAI predicted around 20% fewer).

An explanation of this was found to be that our approach is indeed able to assign more reliable donor junction specific inclusion rates than SpliceAI. To test this we performed the following procedure, using the SpliceAI scores before the underwent normalization as per section 4.7.8.2. We for each donor junction we found that the predicted inclusion ratio for the first non-zero (> 0.01) acceptor site in the truth set correlated well with the true value, and performed only slightly worse than our approach (figure 4.15 top). For the last nonzero acceptor site, we found a substantial increase in SpliceAI scores relative to the true value (figure 4.15 bottom). This is not unreasonable based on the original purpose of SpliceAI, as these sites could easily be strong acceptors for downstream introns, but only weak acceptors for the junction in question. However, it does illustrate the ability of our approach to predict junction-specific inclusion ratios by deprioritizing strong candidate acceptor sites that are not relevant to the junction in question.

We also introduced a threshold-based metric to demonstrate these findings visually. Figure 4.14 illustrates the percentage of junctions that are within a given cross-entropy threshold of the truth for each of these approaches. This presentation backs up the above interpretation; For all junction length and acceptor partitions, SpliceAI has a greater number of very accurate distributions that MaxEntScan, but also has a reasonable number of confidently wrong predictions coming from the prediction of correct acceptor sites that are spliced strongly to other donors, but only weakly to the junction in question.

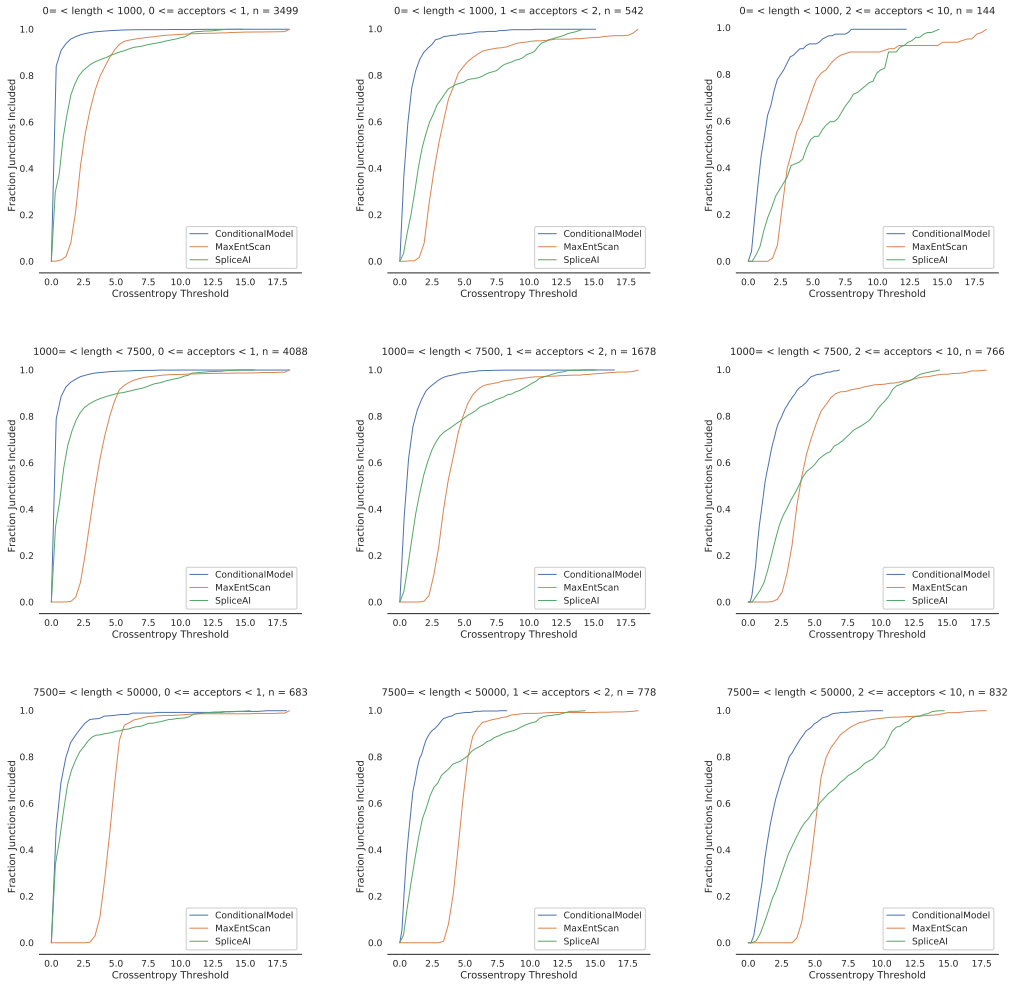

Figure 4.14: We plot cumulative distributions for the three models benchmarked to quantify what percentage of the calculated ψ distributions for each junction (Y axis) fall within a given threshold of their cross entropy calculated with the true ψ distribution on hold out data (X axis).

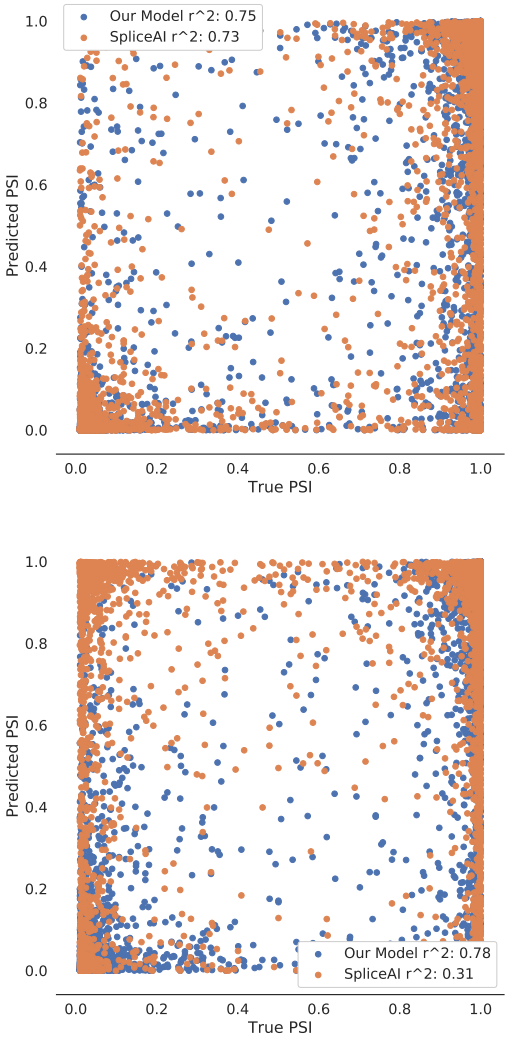

Figure 4.15: **Top:** Correlation between unnormalized SpliceAI scores and true inclusion ratio for first nonzero acceptor site. **Bottom:** Correlation between unnormalized SpliceAI scores and true inlusion ratio for last nonzero acceptor site.

4.8 Validation on Mutation Databases

4.8.1 Biological Datasets - HGMD

We curated an enrichment-control dataset using HGMD (The Human Gene Mutation Database). This is a curated catalogue of known disease-causing mutations. From this database, there are a total of 77,272 mutations that overlap with our junction intervals to test for predicted splice-altering potential. Though these are predominantly near exons, there are 454 examples of mutations $> 100bp$ away from known exons that overlap with introns $< 50kb$, as required by our model.

As a control set, we created a matched number of high-frequency variants in DbSNP (with allele frequency $> 1\%$) that we assume to be splice conserving. In reality, we will also include sites that do impact splicing in a phenotypically benign way, but we ignore this possibility.

To generate this control set with as minimal positional bias as possible, we perform the following procedure: For each HGMD site, we perform a local search in DbSNP within a 1000bp window of the site to find the closest polymorphism that has not already been included in the control set. This procedure produced a total of 71,646 sites.

4.8.2 Biological Dataset - ExAC

To explore the extent to which the splice-altering sites are under natural selection, we use a dataset from the human Exome Aggregation Consortium (ExAC) [Lek et al., 2016]. This is a repository of variation for 60,706 individuals, and therefore provides allele frequency resolution of up to 8.2×10^{-6}. Due to the excessive size of this data, it was not possible to perform a mutagenesis experiment for every reported site. We consequently used only chromosomes 10, 20 and 2, yielding a total of 1,380,337 sites.

4.8.3 Transcript Models

To be able to calculate the model performance on deep intronic variants, we used 3 transcript models to define the location of the introns.

- Principal Transcript Model: GENCODE annotations defined as 'Principal' by the APPRIS catalogue [Rodriguez et al., 2012]. This amounted to a total of 19, 956 unique transcripts.
- Basic: The complete set of GENCODE basic annotations. This was examined in [Frankish et al., 2015] and was found to be similar to RefSeq [O'Leary et al., 2015], containing a total of 84,817 unique transcripts.
- Comprehensive: The GENCODE comprehensive transcript list containing a total of 181,716 unique transcripts.

4.8.4 Biological Dataset GTEx - Whole Genome Sequence

An additional resource published with GTEx [GTEx Consortium and others, 2015] is the whole genome sequencing of 149 individuals that can be paired with the RNA sequencing used to build the model. Using this resource, we directly tested the predictions of our model in section 4.8.8.1.

4.8.5 Deep Intronic Splice Machinery is Under Purifying Selection

We calculated mutagenesis scores for the SNPs in ExAC above and evaluated the extent to which predicted splice altering SNPs are selected against in this population. We performed a comparison of ExAC alternative allele frequency between SNPs that have splice mutagenesis score > 1 and those that have splice mutagenesis score < 0.001. What we find is a statistically significant reduction in allele frequency for

the high splice scores vs low scores (median allele frequency 0.00019 vs 0.00055, p-value 4×10^{-16} with Mann-Whitney U test, figure 4.16 bottom). As a view over a longer evolutionary time horizon, we observed the relationship between these SNPs and PhyloP 100-way scores [Pollard et al., 2010]. We found a similar but weaker effect for comparison between the phylopP scores of SNPs with mutagenesis scores > 1 vs those with scores < 0.001 (p-value 0.0001 with Mann-Whitney U test, median phyloP scores 0.87 vs 0.58, figure 4.16 top right), suggesting conservation between mammalian species.

This result is unsurprising, as this data contains the canonical splice sites, which we know are strongly conserved. However, we also see the same effect for SNPs $>$ 7bp away from known splice junctions, indicating that more subtle mechanisms such as branch points and splice regulatory factors are detected (median allele frequency 0.0004 vs 0.0008, p-value 4×10^{-16} figure 4.16 top left). Interestingly, we were not able to discriminate between these SNPs using the PhyloP scores (p-value 0.57 with Mann-Whitney U test, median phyloP scores -0.015 vs -0.0054), indicating much weaker conservation of the intronic sequence between species.

4.8.6 Predicted Splice Altering Sites are Enriched in Disease Populations

We examined the extent to which we can discriminate between known disease-causing mutations and likely benign SNPs using our splice mutagenesis scores. To this end, we use our control set discussed in section 4.8.1 and calculated splice mutagenesis scores for both sets.

Unsurprisingly, despite our best efforts to find SNPs that are as near as possible in location to the HGMD mutations, we discover a substantially different location profile between these two datasets. Examining this bias with the 3 transcript models defined in section 4.8.3 we discovered, a depletion in intronic sites in HGMD relative to the control set. There are several reasons for this, notably the lower probability

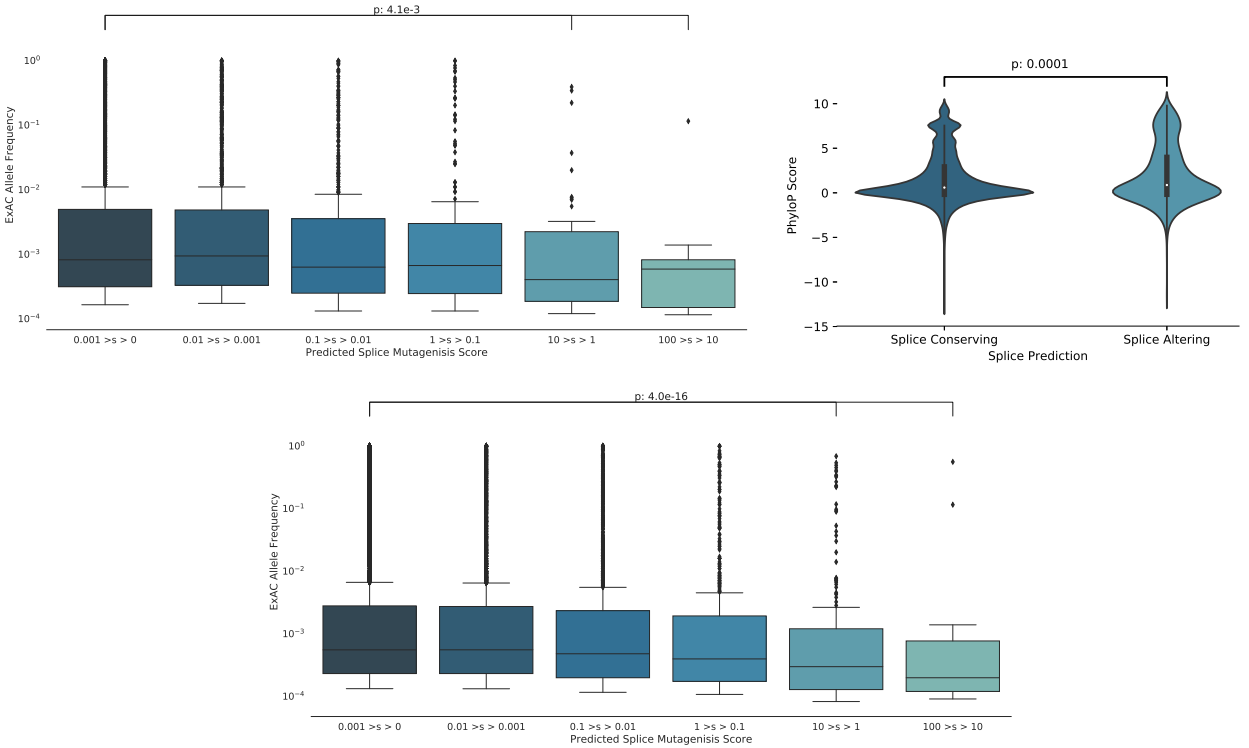

Figure 4.16: For ExAC SNPs on chromosomes 10,20,2 we found that there was a significant negative relationship between MAF and splice mutagenesis score. **Top left:** Plot of the relationship between splice-score and Exac MAF for sites > 7bp from known exons. **Top Right:** Comparison between PhyloP scores for predicted splice-altering vs splice-conserving sites. **Bottom:** Plot of the relationship between splice-score and Exac MAF for all sites.

of exonic polymorphisms rising to allele frequency > 1%, and the probable under-reporting of pathogenic intronic mutations in HGMD. As illustrations of the severity of this problem, we find only 9.5% of HGMD mutations occurring in intronic regions vs a total of 57% of control SNPs when we use the most comprehensive transcript model.

To account for this bias, we partition the sites into 4 categories and compare each one individually. These categories are exonic (i.e in an exon in every transcript in the transcript model), intronic (i.e in an intronic in every transcript in the transcript model), within partially included exons (denoted intronic, exonic) and other for sites that do not fall into any of the above categories such as snps that are in intergenic or promotor regions. We see in figure 4.18 that there is a very large enrichment in splice altering sites in the intronic HGMD vs intronic control over all 3 transcript models. This is unsurprising, as this category contains numerous mutations in known splice dinucleotides. What is more interesting is the observation of a small but significant enrichment of splice altering sites within exons and optionally included exons in HGMD. This implies that splicing disruption could pose an additional pathogenic effect, for exonic mutations beyond typical coding sequence changes.

4.8.7 12% of HGMD Deep Intronic Variants are Explained by Splicing Defects

We further examined this mutagenesis score enrichment for deep intronic sites to establish to what degree we are able to discriminate between pathogenic and benign variants far from annotated exons. Progressively filtering out SNPs that are within a distance of 0,1,2,5,10, 25 and 100 base pairs of known splice junctions (figure 4.19 bottom left). As expected, we did see an initial steep loss of splice impact prediction as we excluded canonical splice site mutations which make up a substantial fraction of the HGMD dataset (figure 4.19 top left), but observed a persistent signal after 25bp, that didn't substantially degrade with greater distance.

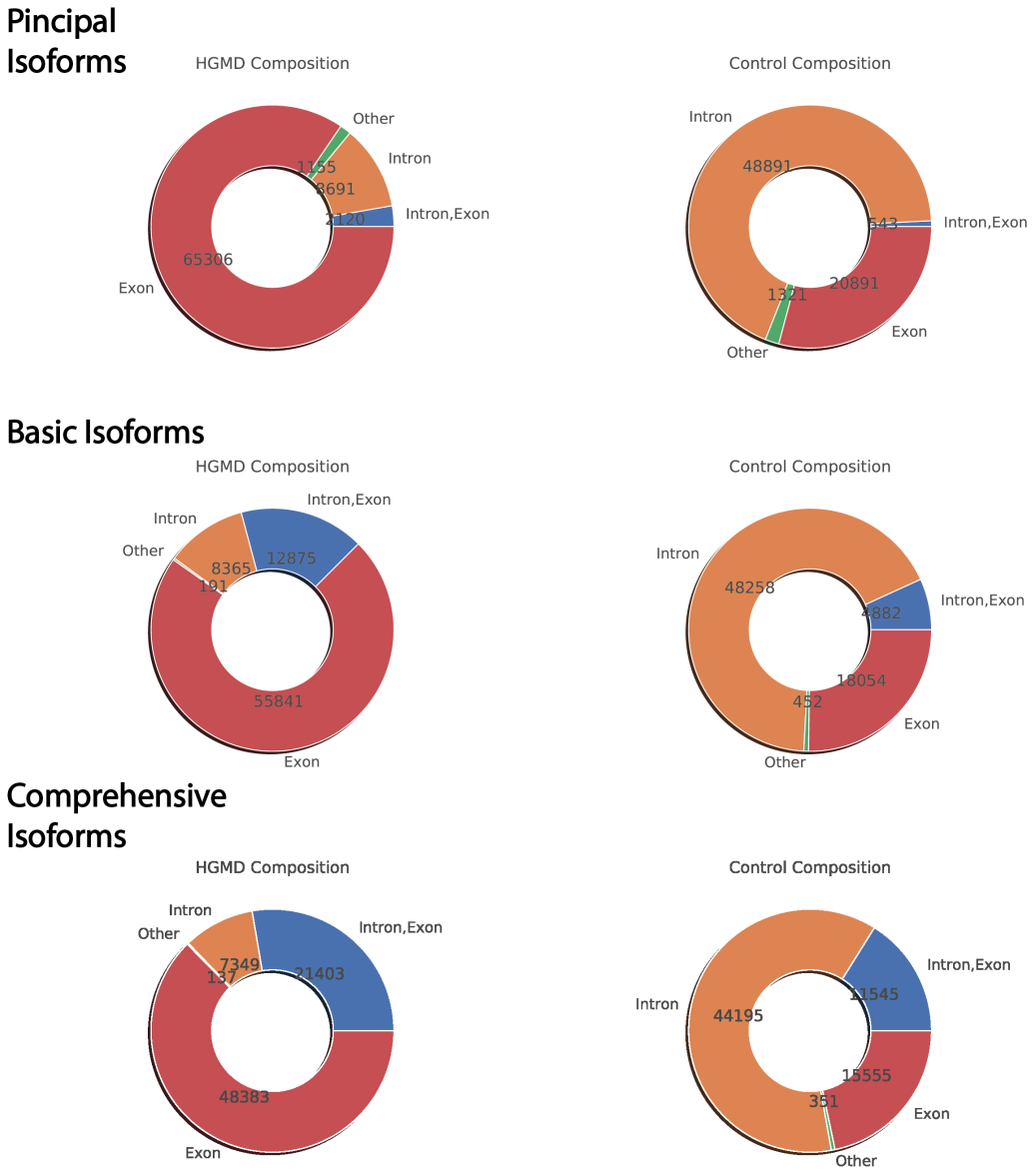

Figure 4.17: Analysis of SNP locations of the HGMD vs matched control datasets reveal a substantial inflation of exon-located SNPs as defined by the transcripts in V24lift37. As we add more transcripts, a progressively greater number of SNPs are in alternatively splice exons.

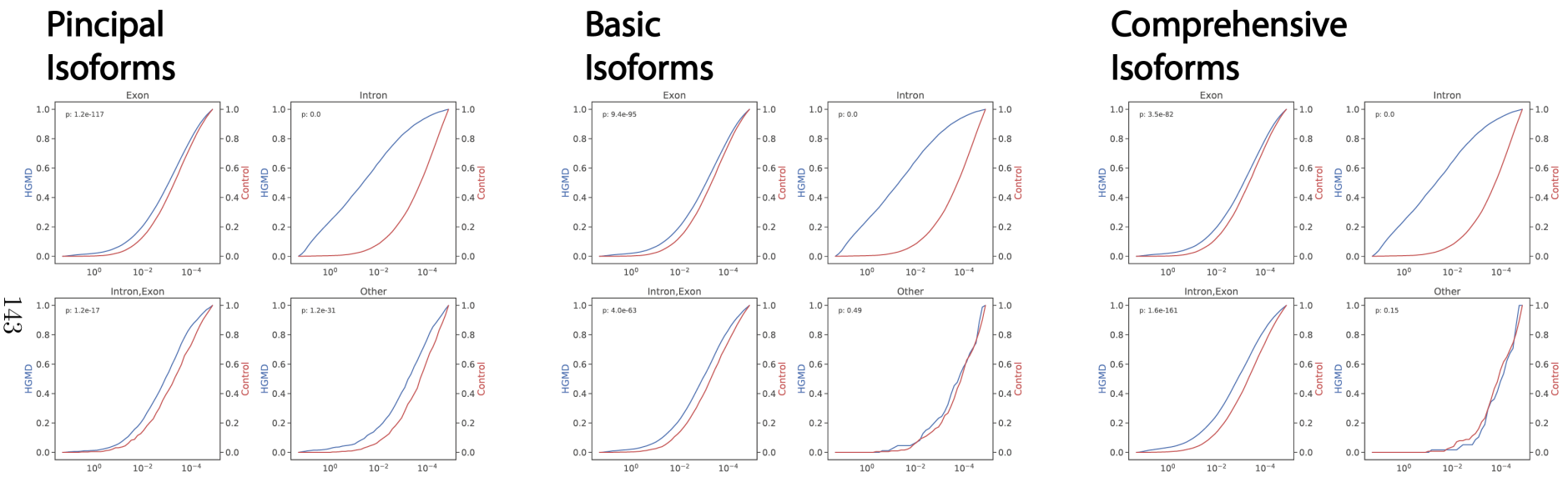

Figure 4.18: Cumulative distributions for splice mutagenesis scores calculated for HGMD and DbSNP control dataset for the 4 discussed sequence contexts across 3 transcript models

We evaluated the strength of our predictive abilities to discriminate between these sets of deep intronic sites (454 HGMD vs $\approx$ 27000 control) by calculating precision recall curves scores assigned to the two groups for our method in addition SpliceAI and MaxEntScan (figure 4.19 top right).

To generate the SpliceAI mutagenesis scores, we trained 5 models using the code provided with the paper and used the median performing model. We calculated a mutagenesis score for SNP i of

$$S_k = \sum_{ij} |\hat{y}_{ij}^{\text{ref}_k} - \hat{y}_{ij}^{\text{alt}_k}| \tag{4.19}$$

where $\hat{y}_{ij}^{\text{ref}_k}$ is the output distribution for a 10kb sequence window centered on the SNP in question with the reference allele.

To generate the MaxEntScan scores, we calculated

$$S_k = \max(\Delta D_k, \Delta A_k) \tag{4.20}$$

where ΔD_k is the change in the output of the donor model between the alt and ref alleles for the k'th snp, and ΔA_k is an analogous score for the acceptor model. Note that to generate each of these Δ scores we run the model a total of 23 times for the acceptor model and 9 times for the donor model with each run centered on a different base as each model takes input sequences of those lengths. We then take the maximum of these runs in each case.

We find only a weak ability to discriminate for all methods, though SpliceAI outperforms our approach (AUPRC 0.18 vs 0.094) in this task. This is perhaps somewhat unsurprising, as this approach explicitly models donor site positions, whereas we only implicitly model it.

We performed two further comparisons (figure 4.20) to test our approach to SpliceAI. Firstly, required the algorithms to discriminate between intronic HGMD sites that have been explicitly demonstrated to impact splicing, vs the remaining intronic HGMD sites. In this comparison, SpliceAI outperformed our method, although this was only a marginal gain (figure 4.20, top). We also discriminated between this

set of HGMD sites that have been validated to impact splicing against our original DbSNP control set, and find qualitatively similar scores to figure 4.19 top right, with weak (though, as expected, improved) AUPR curves for both methods, although again SpliceAI yielded superior results in this task (figure 4.20, bottom).

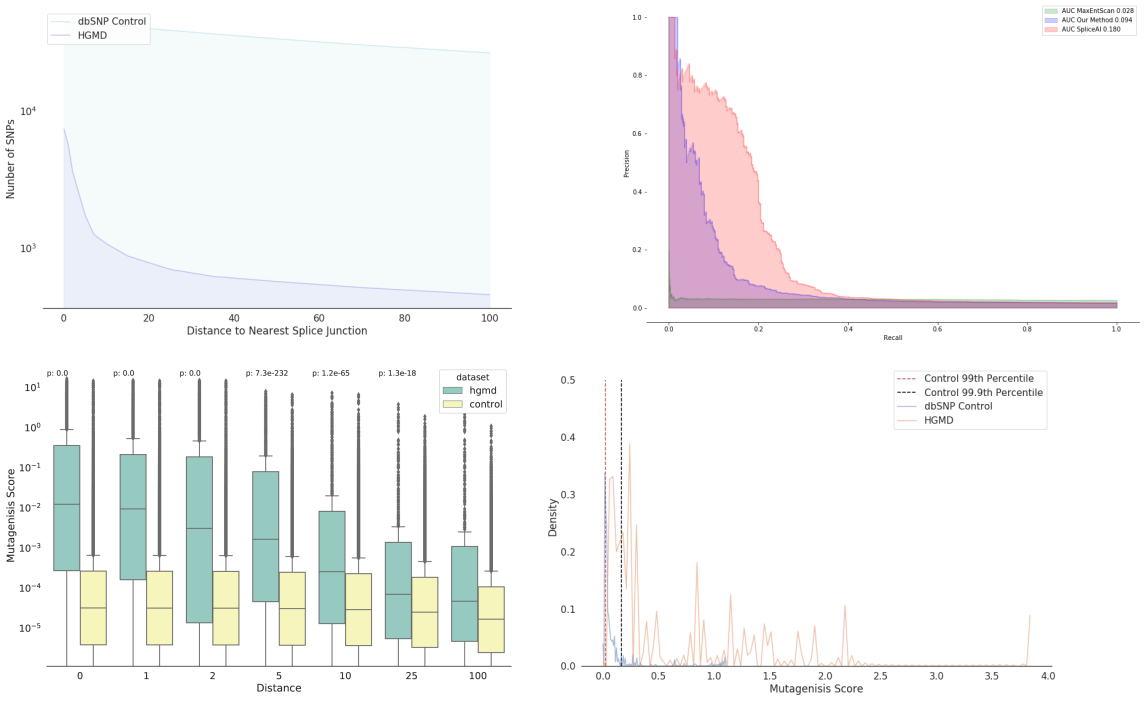

Figure 4.19: **Top Left:** Mutation density as a function of distance from known exons. **Top Right:** Precision-Recall comparison for deep intronic sites between our method, SlpiceAI and MaxEntScan. **Bottom Left:** Comparison of mutagenesis scores for HGMD and our control dataset as a function of distance from nearest exon. **Bottom Right:** Mutagenesis score density for deep intronic sites for control and HGMD.

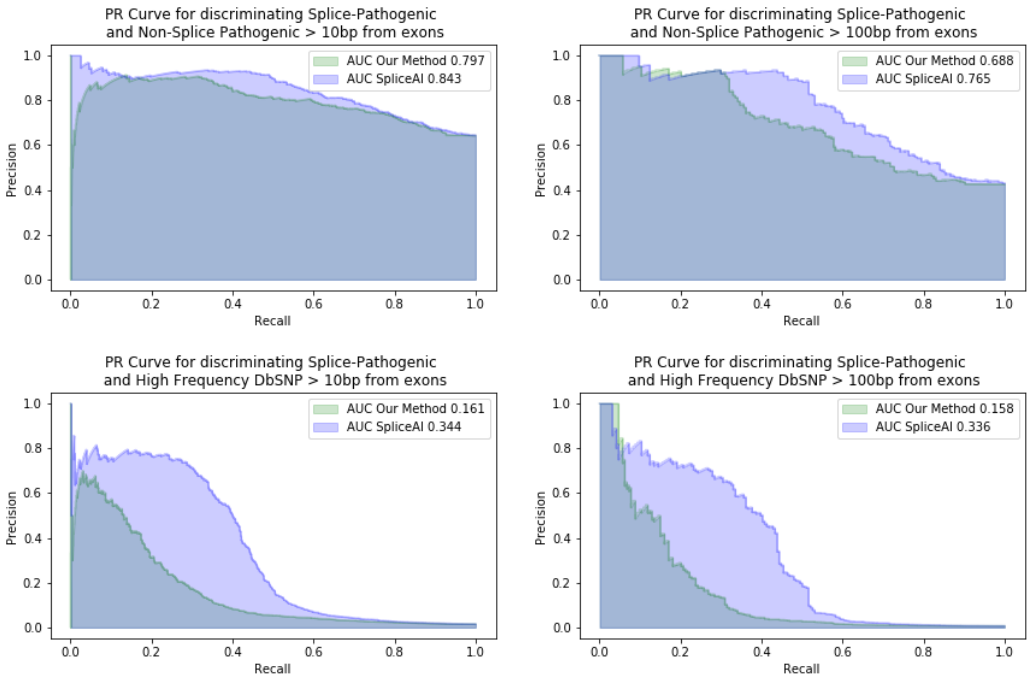

Figure 4.20: **Top:** Precision recall curves for discriminating intronic HGMD sites shown to impact splicing in literature vs the remaining intronic intronic HGMD sites. **Bottom:** Precision recall curves for discriminating intronic HGMD sites shown to impact splicing in literature vs common DbSNP control set

We then examined at the distributions of our mutagenesis scores for the HGMD sites vs the control sites (figure 4.19 bottom right). We find a significant inflation in mutagenesis scores in the HGMD, with 56 of the 454 HGMD sites (12.3%) recording splice altering scores at the 99th percentile in the control population. Further to this, 52 of these sites appeared in the set of the 194 HGMD sites that are considered splice variants, with validated splicing impact, meaning that we were able to identify 27% of these. A comprehensive list of the sites that had greater predicted splice alterning potential than the 99.9th percentile of the null distribution is tabulated in Appendix E.

4.8.8 4 Validated Cryptic Splice Site Case Studies

We looked further at a small number of the 56 predicted splice altering variants in section 4.8.7 to establish what degree our predicted ψ distribution for the alternative SNP matched what has been seen in literature. We present briefly 4 examples where our predicted splice altering deep intronic mutations have been shown in vitro to yield our predicted splicing effect.

Mutation 1: BRCA2 Intron 12

Mutations in the BRCA genes have been a known risk factor for breast cancer (the etymology of the name is BReast CAncer) since the early 90's [Narod and Foulkes, 2004]. BRCA2 is a known tumour suppressor gene [Narod and Foulkes, 2004], and therefore it is particularly important to characterise mutations that are likely to impact its product. We predict that a deep intronic SNP `Chr13 32919384T>G` causes the retention of a cryptic exon beginning 95bp upstream. The mechanism for this is improving a donor site in the middle of intron 12, that then appears to pair with a previously unused upstream acceptor. In carriers of this mutation, we predict this upstream site will be used with 80% probability (figure 4.21).

In addition to being predicted by us, this mutation has both appeared in vivo and subsequent laboratory assays [Anczuków et al., 2012]. Particularly, two patients presented at the referral of a genetic counsellor to the Mazoyer lab in Lyon. They had a family history of breast cancer and both carried the mutation. Reverse transcriptase PCR illustrated the inclusion of this exon, and site-directed mutagenesis found that in the presence of this SNP the dominant transcript contained the cryptic exon, with the wild-type haplotype a barely detectable product.

The authors note in supplemental figure 1 that there are a great number of donor and acceptor motifs predicted by MaxEntScan[Yeo and Burge, 2004] and SPNN [Reese et al., 1997], so it is a clear validation of our method that a minor modification to one of them (the site in question changes from GTAAAAGgtatttca to GTAAAAGgtat**g**tca) gives such a clear result.

Mutation 2: ATP7 Intron 16

Occipital Horn Syndrome is a mild variant of Menkes disease: broadly defined as an X-linked disorder of copper metabolism. In 1994 [Kaler et al., 1994] showed that this syndrome could be caused by splicing defects at known donor junctions leading to exon skipping in the ATP7A gene, motivating greater exploration of deep intronic sites that are likely to perturb splicing similarly.

We predict that deep intronic mutation `ChrX 77287843C>G` leads an additional exon inclusion 68bp upstream with 70% probability (figure 4.21) by the mechanism of improving a downstream donor site analogously to `Chr13 32919384T>G`. This exact mutation was observed in patient 3 of the 2014 study [Yasmeen et al., 2014], where the authors confirm that this exonisation does indeed occur although this exon contains a premature stop codon after 33 amino acids leading to transcript decay via NMD.

Mutation 3: MTM1 Intron 7

We predict that the SNP `ChrX 149808833A>G` in the MTM1 gene leads to an exon inclusion event with 70% probability (figure 4.21) by improvement of an inactive splice junction. This has been validated in [Tosch et al., 2010] and show that it causes centronuclear (myotubular) myopathy, a disease developing any time between early childhood and adulthood characterised by muscle weakness and wasting.

Mutation 4: MTR Intron 3

Methionine synthase, produced by the MTR gene, catalyzes the necessary conversion of amino acid homocysteine to methionine in the presence of a vitamin B12 related compound methylcobalamin and enzyme methionine synthase reductase [Kräutler, 2009]. Methionine synthase deficiency consequently inhibits this pathway, and can result in devastating clinical symptoms such as neonatal seizures, cerebral atrophy, and blindness.

We predict an intron retention event caused by the utilization of a novel acceptor junction 165bp upstream of the constitutive acceptor junction in intron 3 of the MTR gene as the result of snp `Chr1 236971838 A>G`. This is predicted to be a strong junction with 70% acceptor probability (figure 4.21 bottom right) for the donor at

236969534, and contains a stop codon 9 bp into the additional retained sequence.

This mutation has been observed in vivo in [Wilson et al., 1998] to cause significant loss of transcript presence due to NMD in two siblings suffering from MS deficiency and presenting with neonatal seizures.

4.8.8.1 Direct Validation on RNA-seq Data

We then looked to validate our predictions directly on RNA-seq junction calls using GTEx.

We partitioned the HGMD SNPs that we previously analysed into splice-altering (having a δ score > 0.1) and splice-conserving (having a δ scores $\leq 1 \times 10^{-8}$). We then filtered the splice conserving SNPs such that on any individual contig we did not include more conserving SNPs than splice altering SNPs, to achieve an approximately balanced dataset. This process was achieved by sampling at random without replacement. After this, we were left with a total of 6735 splice altering SNPs genome-wide and 6644 in the splice conserving category. From here, we used the whole genome sequencing data available for the GTEx cohort to further reduce the SNP sets to subsets that were polymorphic in these individuals.

Upon doing this, we found a drastic reduction in the number of both splice-altering and splice-conserving sites, to a total of 185 and 421 respectively. Strikingly, we found a substantial depletion in splice altering sites that are polymorphic in this dataset relative to the splice conserving sites (table 4.7). This is perhaps surprising as all of the candidate SNPs here are annotated as disease-causing, implying that splice altering SNPs are conserved even against this stringent background.

For the sites that were polymorphic in our sample population, we compared how the read distributions in junctions overlapping these sites were different in carriers of these mutations vs carriers of the reference allele. To do this, for every SNP in each of the datasets, we used the GTEx junctions called in [Jaganathan et al., 2019] and returned the read counts in overlapping junctions for the carriers and the non-carriers. For both carriers and reference samples, we pool across cell types and then

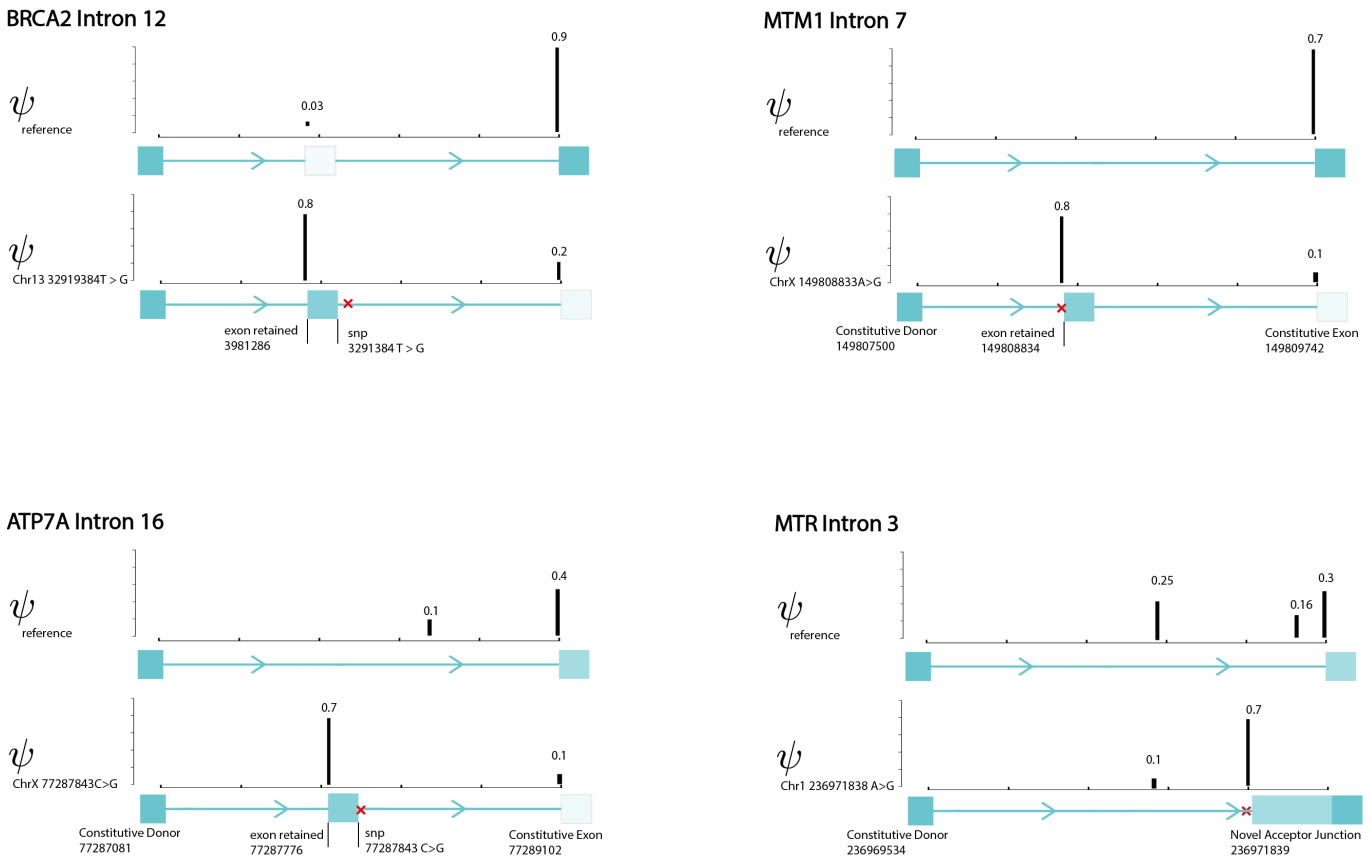

Figure 4.21: Validated pathogenic predicted cryptic splice Sites: **Top Left:** Mutation in cryptic donor site causes exonisation in BRCA2 intron 12 **Top Right:** Mutation in cryptic acceptor site causes exon inclusion in MTM1 intron 7 **Bottom Left:** Mutation in cryptic donor site causes exon inclusion in ATP7A intron 16 **Bottom Right:** Mutation in MTR intron 3 causes novel acceptor site 165bp downstream of known acceptor.

	Polymorphism State		
SNPset	Polymorphic	Monomorphic	Total SNPs
Splice-Altering	185	6,550	6,735
Splice-Conserving	421	6,223	6,644
Total by Polymorphism State	606	12,773	13,379

Table 4.7: Depletion in hgmd splice altering SNPs polymorphic in GTEx cohort relative to hgmd splice conserving GTEx SNPs monomorphic in cohort significant with $p = 2.7 \times 10^{-23}$ (contingency table)

discard any sites that had fewer than 30 reads in total in either the carrier or reference sets. These sets were initially very noisy, with a large number (up to 20) of junctions overlapping the SNP in question containing mostly 1 read each. To clean this up, we removed junctions that didn't carry at least 10% of the total reads overlapping in either the carrier or reference set.

We calculate two empirical read distributions for each SNP in the splice altering and splice conserving sets separately, with

$$c_{ij} = \frac{\#\text{Reads in overlapping junction } j}{\sum_{k=1,2,N_i} \#\text{Reads in overlapping junction } k} \tag{4.21}$$

being the 'carrier distribution', summing up all reads from samples that carry the i'th alternative allele and N_i being the number of junctions that overlap the i'th genomic locus. We similarly define r_{ij}, calculated for all samples that carry the reference allele. Note that the indices of the splice conserving and splice altering snps are independent, so we are ultimately left with:

$c_{ij}^{\text{Conserving}}$ = Fraction of reads from carrier samples in jth junction for splice conserving SNP i

$r_{ij}^{\text{Conserving}}$ = Fraction of reads from reference samples in jth junction for splice conserving SNP i

c_{ij}^{Altering} = Fraction of reads from carrier samples in jth junction for splice altering SNP i

r_{ij}^{Altering} = Fraction of reads from reference samples in jth junction for splice altering SNP i

Using these quantities we calculated a distance metric:

$$d_i^{\text{Conserving}} = \max_j(|c_{ij}^{\text{Conserving}} - r_{ij}^{\text{Conserving}}|) \tag{4.22}$$

$$d_i^{\text{Altering}} = \max_j(|c_{ij}^{\text{Altering}} - r_{ij}^{\text{Altering}}|) \tag{4.23}$$

To assess statistical significance the observed deviation for each snps, we performed a simple sampling procedure as follows:

- For each snp find the number of carriers and reference samples in GTEx n_c, n_r.
- Repeat 5000 times: Partition the GTEx read counts for overlapping junctions into 2 groups of sizes n_c, n_r. If sampled read counts pass filtering procedure above, calculate distances scores d.
- If there are more than 4000 successful samples, calculate bootstrap p value (as per Davison and Hinkley (1997), Bootstrap Methods and their Application)

We plot QQ plots for the two sets of SNPs obtained by this procedure, and see a substantial inflation in the p-values obtained for the SNPs predicted to cause splice disruption 4.22. For this set of SNPs, we see a median p-value of 0.012, and 9 of the 20 sites having a p-value of less than 0.01. Interestingly, we also see an (albeit lesser) inflation of p-values for the SNPs predicted to be splice-conversing, with 6 of the 32 sites yielding a p-value of less than 0.01. This is possibly due to imperfect forecasting, where SNPs that do impact splicing have been incorrectly assigned low mutagenesis scores by the model.

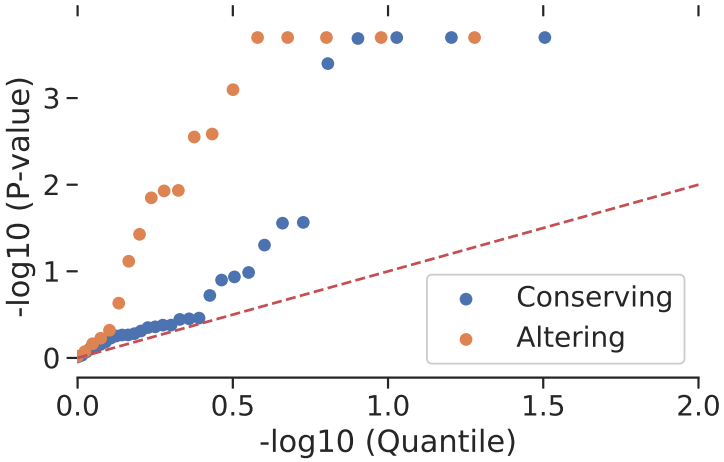

Figure 4.22: QQ plots for the P-values obtained for carrier-control disruption for the predicted splice-conserving HGMD vs predicted splice-altering HGMD SNPs.

4.9 The Curious Case of the Splice-QTL

Further to the previous biological validation, we also tested the degree to which we see enrichment in the number of SNPs predicted to affect splicing vs a control set. Splice QTLs are a method of quantifying splicing, so a concordance with our mutagenesis scores is to be expected. Curiously, we do not see an enrichment of predicted splice altering scores in our spliceQTL set (section 4.9.2), and briefly discuss the possible causes of this phenomenon.

A possible cause of this is simply that the salient splice determining sQTL characteristics are not learnable from sequence. We demonstrate that this is not true in section 4.9.3, where we can discriminate between these sets of SNPs using features learned from sequence.

We conclude with a discussion of the predictive features used in section 4.9.3 that are statistically significant and how to combine them with the approached we have outlined so far in this chapter.

4.9.1 Dataset - Battle et. al

We use the dataset in Battle et. al [Battle et al., 2014], with splice QTLs (sQTLs) calculated from 922 individuals. These sQTLs are calculated using transcript ratio as a quantitative trait. Specifically, each of the 922 individuals had their transcriptome sequenced and aligned such that the expression levels of relevant transcripts can be calculated. Withing a given gene, for the ith transcript and the jth individual the expression level is given as e_{ji}, and the calculated splicing phenotype is

$$\frac{e_{ji}}{\sum_k e_{jk}}$$

This metric was then associated with the measured SNPs for each individual within 1mb of the relevant transcription start site. This procedure generated a total of 2851 splice altering SNPs, of which 1801 overlapped with our junctions. As for the HGMD dataset in 4.8.1, we constructed a matched DbSNP control to test these for

enrichment for sites with predicted high splice mutagenesis score.

4.9.2 Loci With High Splice Scores are not Enriched Among sQTLs

We compared the enrichment for high splice scores for 3 models: Our two models from section 4.6.4 and SpliceAI, using the calculated splice mutagenesis scores from all 3 to calculate a ROC curve for discriminating between the sQTL and control datasets. We found some initial signs of enrichment in the early retrieval domain, where all 3 models appeared to show moderate enrichment for sQTLs (figure 4.23 top left), which corroborates the result presented in (figure 4.23 top right). Here, we used the published 'high confidence' splice altering SNPs from SpliceAI and looked to see if a greater percentage was included in the test or control datasets which we find to be statistically significant using a chi-squared test. However, looking at the entire ROC curve, we find minimal global enrichment for high splice scores, with all ROCs < 0.51.

4.9.3 sQTLs are Predictable Using Learned Epigenetic Features

To test the hypothesis that these contain signal that is not learnable from sequence we devised a simple linear model using features derived from sequence as input. Specifically, we calculate scores for the SNPs by

$$y \sim \beta x$$

with a logistic link function, and the response variable y given by

$$y = 1, \text{ If SNP is sQTL}$$

$$y = 0, \text{ If SNP is Control}$$

Here, the features x are the mutagenesis scores calculated from [Zhou and Troyanskaya, 2015b] for each SNP. These include a variety of accessibility, transcription factor binding and histone modification predictions. We fit the model on 75% of the SNPs, and use the remaining 25% as test data. We found that this lead to significantly greater predictability (figure 4.23 bottom left), even when the sQTLs and control SNPs were restricted to being solely deep-intronic. Figure 4.23 bottom right, shows the distributions of 100 bootstrapped ROC curves. We filtered SNPs that were progressively further from known exons and sampled at random 75% of the data for training as above.

The above findings are somewhat surprising. It is difficult to reconcile the fact that our model has a validated $\approx$ 82% predictive accuracy for the next acceptor site but cannot discriminate between sQTLs and null sites. We know that it is possible to discriminate between these two sets from epigenetic features learned from sequence in section 4.9.2 (albeit slightly), so the model is either unable to or chooses not to learn these features for the purposes of predicting the ψ distribution. To interpret this further, we calculate the significant features via an ANOVA (table 1.4.5), which appear to be predominantly transcription factors or other compounds associated with transcriptional regulation. We hypothesize that this could be due to a large component of the splicing determination captured by the sQTLs is governed by more distal regulation of elongation such as choice of promotor [Tasic et al., 2002] or enhancer [Kadener et al., 2002], and that the isoform ratio is changing in the sQTLs due to these effects as opposed to change in local exon definition or the spliceosome and related splice enhancers/silencers. If this is the case, then it seems possible that the model could be improved by adding additional sequence content around the transcription start site.

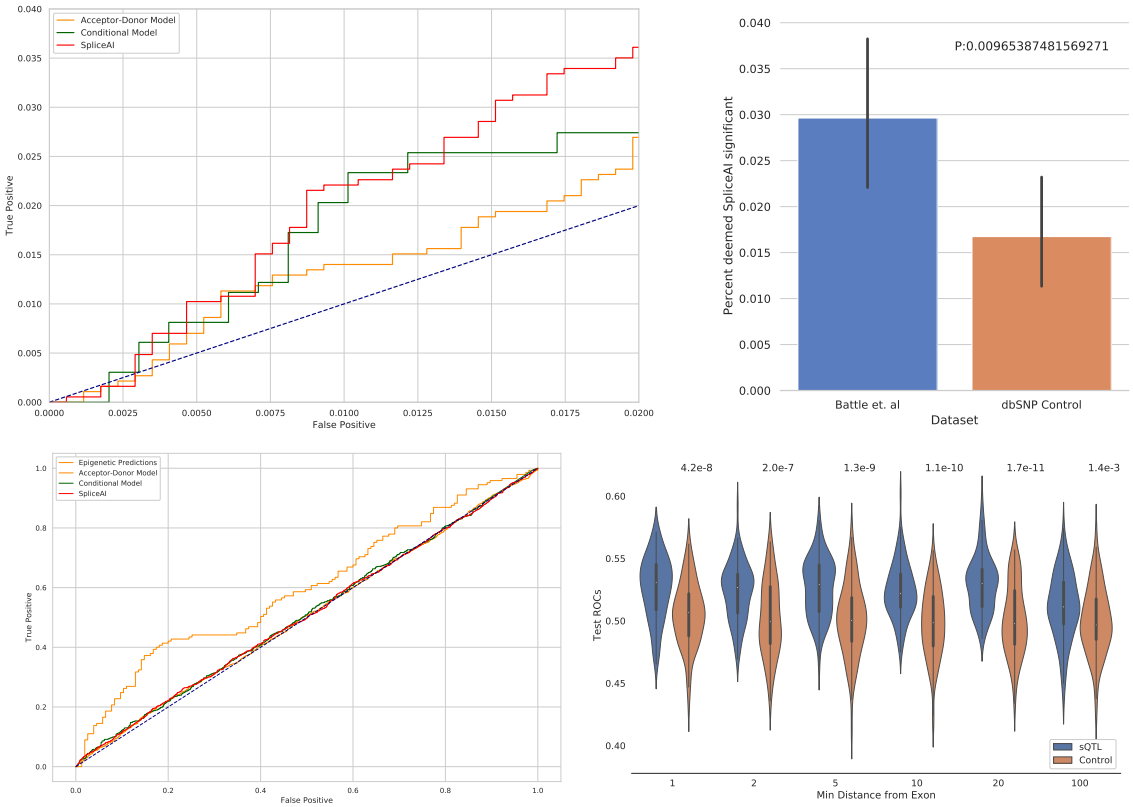

Figure 4.23: **Top (Left):** ROC curves for the 3 models calculated from splice mutagensis scores on the Battle et al. sqtls SNPs and matched DpSNP control. **Top (Right):** Fraction of SNPs present in the published SpliceAI precomputed set for the two datasets, corresponding to a single point on the calculated ROC curve. Here we observed an approximated two-fold enrichment for sqtl SNPs in the spliceAI significant set vs the control SNPs. **Bottom (Left):** Full ROC curves for the 3 methods reveal poor explanatory power of these models in comparison to the predictions derived from the linear model using DeepSEA features. **Bottom (Right):** ROCs for epigenetic features consistently elevated above random even when fitted only on intronic SNPs.

Feature	bf corrected value
c-MYC	$4.85 \times 10^{-}5$
SP2	$4.98 \times 10^{-}5$
DNase	0.00082
THAP1	0.005
Pol2	0.011
FOSL1	0.011
CTCF	0.014
CTCFL	0.015
GABP	0.023
TAF1	0.031

Figure 4.24: Significant p-values after Bonferroni correction derived for the DeepSea[Zhou and Troyanskaya, 2015b] mutagenesis scores for a variety of sequence determined epigenetic marks between the sqtls SNPs in Battle et al. and the position-matched DbSNP control set.

4.10 Discussion

In this chapter, we have devised and developed a novel method for predicting exon inclusion by using a dilated convolutional network with positionally dependent convolutional weights to model an empirically derived *psi* distribution. We show that this method is state of the art at predicting this distribution, which is a necessary step on the path to predicting entire isoforms.

We have incorporated this algorithm into a package that allows large scale and fine-scale interpretation of the genome in terms of splicing relevance, in addition to allowing a prediction of the effect of previously unobserved mutations. We validated these predictions by checking for conservation in splice altering bases, including bases deep within introns, in addition to directly using GTEx junction data. Additionally, we compared our approach to two literature methods and found it to be state-of-the-art at the task of predicting the basewise ψ distribution.

We then used this model to study known pathogenic mutations deep within introns, and understand to what extent we are able to predict validated HGMD pathogenic cryptic splice events. We found strong inflation of predicted deep intronic splice causing mutations vs a matched DbSNP set and found 27% of these sites exhibited splice alteration at the 99th percentile in this null set. We observed that our approach performed substantially better than the well known MaxEntScan [Yeo and Burge, 2004] at discriminating between these sets, but slightly worse than the recently published SpliceAI [Jaganathan et al., 2019], which neglects inclusion rates and attempts to flag all donor an acceptor sites. We note that no method performs with high AUPRC on this problem, suggesting that there are still substantial gains to be made both here, and also in the interpretation of published splicing QTLs.

On looking at our predictions we found explicit corroboration with literature results for deep intronic events in 4 genes, BRCA2, MTR, MTM1 and ATP7A where we predict activation of cryptic splice sites that are validated experimentally. For each of these events, we provide easily interpretable sparse distributions for the acceptor

location probability indicating this could be a valuable tool to clinicians.

4.10.1 Future Work

There are two clear ways to extend this work. The first being whole isoform quantification and the second being tissue-specific splicing predictions. With these two improvements, we would be substantially closer to an full in silico cellular model.

In terms of progress towards tissue specific splicing, we have made a substantial effort as part of this research with limited success. Specifically we have tried 3 high level approaches:

- Create models that predict ψ distributions for the 53 unique tissues in GTEx simulatenously as a 53×57500 tensor.
- Create models that take in a 'tissue code' vector as an additional input and try to predict the ψ distributions specific to that tissue
- Train independent models unique to each tissue

Qualitatively, the issues we face are similar across approaches and can be characterized as overfitting and underdispersion (specifically, our predictions are very close to the bulk tissue distribution). An indirect cause of both of these phenomena is the observation that only a small proportion of junctions are spliced in a tissue-specific way (only between 7-21 % of sQTLs were observed to be tissue-specific in [GTEx Consortium and others, 2015]), and so consequently there is a much smaller dataset of relevant training data whilst trying to capture a much richer phenomenon. The battle at this point is to determine to what extent it is important to learn more general machinery from the bulk tissue via a transfer learning [Pan and Yang, 2009] regimen. The broad tradeoff is: including lots of non-tissue specific junctions at the cost of all predictions looking very similar vs including a small number of non-tissue specific junctions and overfitting due to insufficient data. The fact that neither COSSMO [Bretschneider et al., 2018] or SpliceAI [Jaganathan et al., 2019] were able to model this convincingly indicates that this is still very much an open problem.

In terms of whole isoform quantification, the correct approach theoretically is to attempt to model the isoform graph directly. This has clear interpretability and utility benefits over any current literature approaches but comes with its own challenges. Firstly, one has to construct a thorough 'splicing graph' [Heber et al., 2002] which in itself a non-trivial challenge for longer exons. Then one needs to model this computationally, which again is challenging due to the scale of the genome. Machine learning techniques that are applicable to this problem are graph convolutional networks [Henaff et al., 2015], but these are certainly not powerful enough to scale to modelling a gene-scale dense graph, and consequently to make further progress it will currently be necessary to make certain sparsening approximations such as only allow connections between sites that have hamming distance ≤ 1 to theoretical donor and acceptor motifs.

Chapter 5

Conclusion

Recent advances in statistical methodology, specifically the construction and training of highly complex deep learning models, combined with novel biological assays have presented us with a phenomenal opportunity to build models of the cell to predict certain crucial pathways with unprecedented accuracy.

In this work, we first explored adapting ANNs to be equivariant to reverse complement symmetry. We showed that this yields improved predictive accuracy, and improves representational stability. We showed that this can be a powerful method of motif discovery in comparison to existing approaches, and then used this to investigate motifs that determine whether or not a sequence is likely to be contained within a recombination hotspot. We further developed models that achieved state of the art performance at predicting splice-sites and show they allow valuable interpretation of deep-intronic mutations that impact splicing. To finish this work, we discuss how this fits in with the larger challenges and goals of the field and issues that could prevent these approaches being adopted into existing clinical pipelines.

5.1 Progress Towards Full in Silico Cellular Computation

The models for splicing we have presented represent only a small part of the mechanisms of the cell. Ideally, what we would like is the ability to predict the protein composition, in a tissue-specific manner, of a cell given its genome, age and external stimuli. It is not clear the best way to construct this model, and the field is currently a tapestry of piece-wise incremental improvements to predictions of the constituent parts of this problem.

This pragmatic, attritional approach is not without its merits; New tools and methods become available to clinical bioinformaticians as they are developed, and this, in turn, trickles down to improved diagnoses for real patients. Indeed, a number of the case studies of predicted cryptic splice-sites that we validated in literature were discovered as a result of this collaborative pipeline with the previous generation of tools.

This, however, might not be the optimal approach. Trends in deep learning literature favour training models end-to-end over breaking the problem up, as this allows the model to perform all of the featurization, possible allowing information to be pooled across these sub-problems. Certainly, given the dependence on Pol II dynamics, a complete model for splicing should be a model for expression also.

5.2 Improvements in Model Interpretation

Philosophically, it is important to decide to what extent we are prepared to lose interpretability of our models to achieve superior accuracy. In the broader field of machine learning [Selvaraju et al., 2017], argue that interpretability of deep learning models is important for at least two reasons: To build trust in the model's predictions, and to learn from the model. Both of these are vitally important if one wants to implement these approaches in a clinical pipeline.

In this work, most of the model interpretation is performed either by looking at which parts of the input are important for determining the model prediction, or by looking at the first layer of weights. This is reasonable, but neglects any interpretation of the interaction between regulatory elements, at least in a way that can be interpretable by humans. The field of model interpretation is an active area of research, with numerous recent advances [Shrikumar et al., 2017b, Selvaraju et al., 2017], but finding the best way of summarizing networks to explain their predictions in a way that humans can understand is still a challenging problem.

5.3 Fundamental Roadblocks

What powerful modelling tools can not do is differentiate causality from correlation. For example, in the work in chapter 2, we aimed to discriminate between recombination hotspots and coldspots. Due to biased gene conversion [Duret and Galtier, 2009], we know that GC content is predictive of this. Indeed, we found that not normalizing for GC content increased our predictive accuracy by around 10% (data not shown), through the learning of an uninteresting correlation. To make progress in this field, we must attempt to be cognizant of the underlying biological models to the extent we can be, and view un-validated, opaque predictions sceptically.

Appendix A

Chapter 2 Network Parameters

A.0.1 Baseline Asymmetric Networks

We performed a hyperparameter search for the best asymmetric topologies for each of the two datasets, not allowing for equivariance or MC dropout layers.

Simulation topology

Layers

- Input 1000 ×4
- Convolutional Layer (12 filters, length 14)
- Elu activation
- Global Spatial Max Pooling
- 2 Output neurons
- Softmax

Regularization

- L2 norm 0

Recombination topology

Layers

- Input 997 ×4
- Convolutional Layer (16 filters, length 30)
- Elu activation
- Spatial Max Pooling (Size 8, Stride 8)
- Convolutional Layer (16 filters, length 4)
- Elu activation
- Global Spatial Max Pooling
- 2 Output neurons
- Softmax

Regularization

- L2 norm 0.0003

Networks obtained for the data augmentation optimization were identical, except that the optimal L2 regularization parameter was 0 for the recombination dataset, and the number of filters was doubled to 32. Note that when we followed the procedure in A.2, we used an equivariant Bayesian network with this 32 filters when comparing against augmented data.

A.0.2 Bayesian Equivariant Networks

For each data set, we performed independent hyperparameter searches for the case of (1) equivariant networks, (2) Bayesian networks, and (3) equivariant Bayesian networks, as described in the main text. Note that for the case of equivariant networks

we kept the same number of filters but tied their parameters to achieve equivariance, effectively halving the number of filters and parameters.

We summarise the results by giving the explicit networks for case (3); the optimal networks for cases (1) and (2) turned out to be the same as those for case (3) except for dropping the relevant layers/features (underlined), and changing "Equivariant MC dropout" into "MC dropout" where relevant.

Simulation topology

Layers

- Input 1000 ×4
- Equivariant Convolutional Layer (12 filters, length 14)
- Elu activation
- Reverse Complement Mean Pool
- Equivariant MC dropout (p = 0.1)
- Global Spatial Max Pooling
- 2 Output neurons
- Softmax

Regularization

- L2 norm 0

Recombination topology

Layers

- Input 997 ×4
- Equivariant Convolutional Layer (16 filters, length 30)
- Elu activation
- Equivariant MC Dropout (p=0.1)
- Spatial Max Pooling (Size 8, Stride 8)
- Equivariant Convolutional Layer (16 filters, length 4)
- Elu activation
- Equivariant MC Dropout (p=0.1)
- Global Spatial Max Pooling
- Reverse Complement Sum Pooling
- 2 Output neurons
- Softmax

Regularization

- L2 norm 0

Note that for both datasets we initialized the output layer as per section 3.8 in the main text.

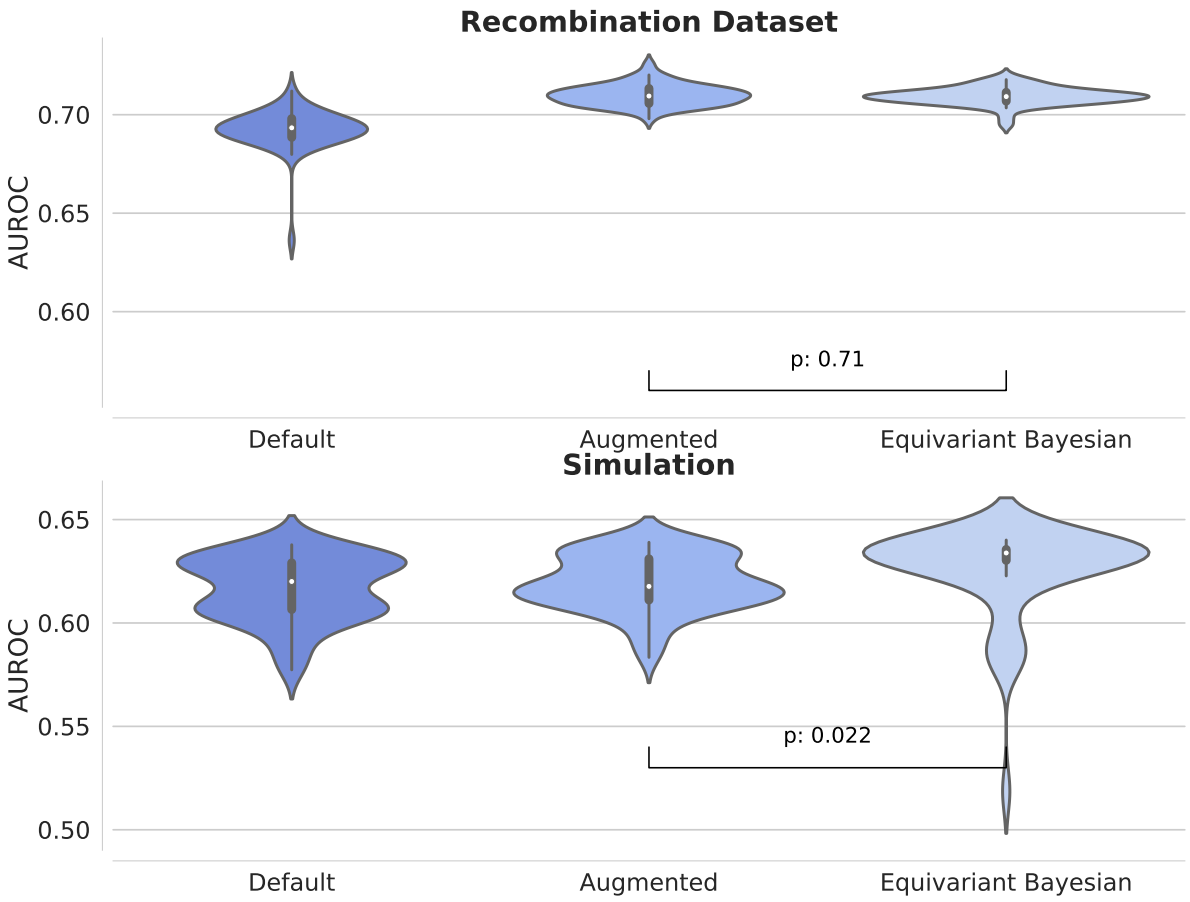

Figure A.1: Comparison between convergence accuracy for best topology optimized on the augmented dataset with MC dropout (augmented) vs the same topology with added equivariance (Equivraint Bayesian) for both simulated and recombination datasets

Appendix B

PRDM9 Motif Effect Sizes

The contexts and effect size for motifs tested in chapter 3. Table B.2 shows the calculated parameter a_m for each of the motif types and sequence contexts for motifs defined in table B.1.

Table B.4 shows a_m calculated for novel motifs defined in table B.3 for the specified sequence contexts.

For both tables the standard errors were calculated by sampling hotspot and coldspot counts with replacement for 10 independent runs.

Motif Type	Sequence Context	a_m	Standard Error
exact_match_canonical	NonRepeat	0.0160	0.000306
one_mismatch_canonical	"NonRepeat"	0.0598	0.000666
two_mismatch_canonical	"NonRepeat"	0.183	0.0179
canonical_prefix_three_mismatch_suffix	"NonRepeat"	0.0156	0.00153
canonical_prefix_four_mismatch_suffix	"NonRepeat"	0.00992	0.000773
exact_match_relaxed_canonical_1	"NonRepeat"	0.0219	0.00100
exact_match_relaxed_canonical_2	"NonRepeat"	0.0236	0.00252
exact_match_canonical	'THE1A'	0.0001	0.0
exact_match_permuted_1	'THE1A'	0.00143	7.88e-05
exact_match_permuted_2	'THE1A'	0.00125	6.67e-05

exact_match_permuted_3	'THE1A'	0.00324	0.000131
all_motifs_except_exact_matches	'THE1A'	0.00289	0.000231
exact_match_canonical	'THE1B'	0.00313	0.000126
exact_match_permuted_1	'THE1B'	0.00485	0.000198
exact_match_permuted_2	'THE1B'	0.0150	0.000247
exact_match_permuted_3	'THE1B'	0.00363	0.000245
all_motifs_except_exact_matches	'THE1B'	0.00881	0.000367
exact_match_canonical	'THE1C'	0.0001	0.0
exact_match_permuted_1	'THE1C'	0.00100	5.30e-05
exact_match_permuted_2	'THE1C'	0.0001	1.69e-05
exact_match_permuted_3	'THE1C'	0.0001	8.93e-05
all_motifs_except_exact_matches	'THE1C'	0.000487	0.000201
exact_match_canonical	'THE1D'	0.0001	0.0
exact_match_permuted_1	'THE1D'	0.00240	7.52e-05
exact_match_permuted_2	'THE1D'	0.000410	6.77e-05
exact_match_permuted_3	'THE1D'	0.000842	0.000200
all_motifs_except_exact_matches	'THE1D'	0.0001	0.000170
exact_match_canonical	'L2'	0.00385	0.000125
one_mismatch_canonical	'L2'	0.0132	0.000396
two_mismatch_canonical	'L2'	0.0381	0.000998
relaxed_motif_1_except_exact_matches	'L2'	0.00374	0.000172
relaxed_motif_2_except_exact_matches	'L2'	0.0101	0.000372
exact_match_semi_relaxed_1	'AluY'	0.0132	0.000471
exact_match_semi_relaxed_2	'AluY'	0.0001	0.0
exact_match_semi_relaxed_1	'AluSc'	0.00403	0.000194
exact_match_semi_relaxed_2	'AluSc'	0.0001	0.0
exact_match_semi_relaxed_1	'AluSg'	0.00569	0.000197
exact_match_semi_relaxed_2	'AluSg'	0.000789	0.000283

exact_match_perturbed_prefix_1	'GA-rich'	0.00355	0.000263
exact_match_perturbed_prefix_2	'GA-rich'	0.00299	0.000287
exact_match_canonical_prefix	'(GGGA)n'	0.0001	8.90E-005

Table B.2: Calculated a_m's for the classical PRDM9 motif in a variety of sequence contexts

Motif Type	Meaning
exact_match_canonical	Regex match to CCTCCCT[A-Z]{2}CCAC
one_mismatch_canonical	One mismatch to CCTCCCT[A-Z]{2}CCAC not previously counted
two_mismatch_canonical	Two mismatches to CCTCCCT[A-Z]{2}CCAC not previously counted
canonical_prefix_three_mismatch_suffix	Exact match to prefix CCTCCCT, up to 3 mismatches to suffix CCAC, not previously counted
canonical_prefix_four_mismatch_suffix	Exact match to prefix CCTCCCT, up to 3 mismatches to suffix CCAC, not previously counted
exact_match_relaxed_canonical_1	Regex match to CC[A-Z]{1}CC[A-Z]{1}T[A-Z]{2}CC[A-Z]{1}C, not previously counted
exact_match_relaxed_canonical_2	Regex match to CC[A-Z]{1}CC[A-Z]{4}CC[A-Z]{1}C, not previously counted
exact_match_permuted_1	Regex match to CCTCCCT[A-Z]{2}CCAT, not previously counted
exact_match_permuted_2	Regex match to CCTCCCT[A-Z]{2}CCAT, not previously counted
exact_match_permuted_3	Regex match to CCTCCCT[A-Z]{2}CCAT, not previously counted
relaxed_motif_1_except_exact_matches	Regex match to CC[A-Z]{1}CC[A-Z]{1}T[A-Z]{2}CC[A-Z]{1}C, but not a match to canonical
relaxed_motif_2_except_exact_matches	Regex match to CC[A-Z]{1}CC[A-Z]{1}T[A-Z]{2}CC[A-Z]{1}C, but not a match to canonical
exact_match_semi_relaxed_1	Regex match to CC[A-Z]{1}CC[A-Z]{4}CC[A-Z]{1}C, not previously counted
exact_match_semi_relaxed_2	Regex match to CC[A-Z]{1}CCTC[A-Z]{2}CC[A-Z]{1}C, not previously counted
exact_match_perturbed_prefix_1	Regex match to CCTCCCTT, not previously counted
exact_match_perturbed_prefix_2	Regex match to CCTCCTT, not previously counted
exact_match_canonical_prefix	Regex match to CCTCCCT, not previously counted

Table B.1: Canonical Motif definitions

Motif Type	Meaning
novel_pre_short_exact_match_4_c	Regex Match to CA[A-Z]{4}T[A-Z]{2}T[A-Z]{7}CCCC, not previously counted
novel_pre_long_exact_match_4_c	Regex Match to CA[A-Z]{4}T[A-Z]{2}T[A-Z]{8}CCCC, not previously counted
novel_pre_short_exact_match_5_c	Regex Match to CA[A-Z]{4}T[A-Z]{2}T[A-Z]{7}CCCCC, not previously counted
novel_pre_long_exact_match_5_c	Regex Match to CA[A-Z]{4}T[A-Z]{2}T[A-Z]{8}CCCCC, not previously counted
novel_pre_short_mismatch_5_c	1 mismatch to CA[A-Z]{4}T[A-Z]{2}T[A-Z]{7}CCCCC, not previously counted
novel_pre_short_mismatch_4_c	1 mismatch to CA[A-Z]{4}T[A-Z]{2}T[A-Z]{7}CCCC, not previously counted

Table B.3: Novel Motif definitions

Motif Type	Sequence Context	a_m	Standard Error
novel_pre_short_exact_match_4_c	NonRepeat	0.0168	0.000523
novel_pre_long_exact_match_4_c	NonRepeat	0.0129	0.000684
novel_pre_short_exact_match_5_c	NonRepeat	0.0202	0.000503
novel_pre_long_exact_match_5_c	NonRepeat	0.0104	0.000383
novel_pre_short_mismatch_5_c	NonRepeat	0.0502	0.00227
novel_pre_short_mismatch_4_c	NonRepeat	0.0287	0.00132
novel_pre_short_exact_match_4_c	L2	0.00294	0.000192
novel_pre_long_exact_match_4_c	L2	0.00125	9.52e-5
novel_pre_short_exact_match_5_c	L2	0.00149	9.05e-5
novel_pre_long_exact_match_5_c	L2	0.00135	8.64e-5
novel_pre_short_mismatch_5_c	L2	0.0114	0.000364
novel_pre_short_mismatch_4_c	L2	0.00412	0.000321

Table B.4: Calculated a_m's for the novel motif in a variety of sequence contexts

Appendix C

Splicing Networks Topology

Networks used in chapter 3 as generated by the keras command model.summary().

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 57500, 4)	0
spatial_term_1 (SpatialTerm)	(None, 57500, 4)	2
conv1d_1 (Conv1D)	(None, 57500, 50)	6050
batch_normalization_1	(Batch (None, 57500, 50)	200
conv1d_2 (Conv1D)	(None, 57500, 84)	42084
batch_normalization_2	(Batch (None, 57500, 84)	336
conv1d_3 (Conv1D)	(None, 57500, 89)	74849
batch_normalization_3	(Batch (None, 57500, 89)	356
conv1d_4 (Conv1D)	(None, 57500, 95)	84645
batch_normalization_4	(Batch (None, 57500, 95)	380
conv1d_5 (Conv1D)	(None, 57500, 100)	95100
batch_normalization_5	(Batch (None, 57500, 100)	400
conv1d_6 (Conv1D)	(None, 57500, 106)	106106
batch_normalization_6	(Batch (None, 57500, 106)	424
conv1d_7 (Conv1D)	(None, 57500, 113)	119893
batch_normalization_7	(Batch (None, 57500, 113)	452

conv1d_8 (Conv1D)	(None, 57500, 119)	134589
batch_normalization_8	(Batch (None, 57500, 119)	476
conv1d_9 (Conv1D)	(None, 57500, 126)	150066
batch_normalization_9	(Batch (None, 57500, 126)	504
conv1d_10 (Conv1D)	(None, 57500, 134)	168974
batch_normalization_10	(Batc (None, 57500, 134)	536
conv1d_11 (Conv1D)	(None, 57500, 1)	1341
batch_normalization_11	(Batc (None, 57500, 1)	4
cropping1d_1 (Cropping1D)	(None, 50000, 1)	0
flatten_1 (Flatten)	(None, 50000)	0
lambda_1 (Lambda)	(None, 50000)	0

Table C.1: Fully Convolutional Network with positionally dependent convolutional filters. This is implemented through the 'spatial term' in layer 1 as expalained in the main text. These correspond to parameters: $f_l = 30, n_f = 50, b_l = 1, n_c = 10, w = 10, o = 10, b = 80, s_k = 1.05$

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None 50000 4)	0
conv1d_5 (Conv1D)	(None 50000 40)	6440
batch_normalization_5 (BatchNor	(None 50000 40)	160
dropout_1 (Dropout)	(None 50000 40)	0
conv1d_6 (Conv1D)	(None 50000 92)	36892
batch_normalization_6 (BatchNor	(None 50000 92)	368
conv1d_7 (Conv1D)	(None 50000 86)	79206
batch_normalization_7 (BatchNor	(None 50000 86)	344

conv1d_8 (Conv1D)	(None 50000 80)	68880
input_2 (InputLayer)	(None 85 4)	0
batch_normalization_8 (BatchNor	(None 50000 80)	320
conv1d_1 (Conv1D)	(None 62 8)	776
conv1d_9 (Conv1D)	(None 50000 74)	59274
batch_normalization_1 (BatchNor	(None 62 8)	32
batch_normalization_9 (BatchNor	(None 50000 74)	296
conv1d_2 (Conv1D)	(None 58 12)	492
conv1d_10 (Conv1D)	(None 50000 69)	51129
batch_normalization_2 (BatchNor	(None 58 12)	48
batch_normalization_10 (BatchNo	(None 50000 69)	276
conv1d_3 (Conv1D)	(None 56 24)	888
conv1d_11 (Conv1D)	(None 50000 64)	44224
batch_normalization_3 (BatchNor	(None 56 24)	96
batch_normalization_11 (BatchNo	(None 50000 64)	256
conv1d_4 (Conv1D)	(None 52 12)	1452
conv1d_12 (Conv1D)	(None 50000 59)	37819
batch_normalization_4 (BatchNor	(None 52 12)	48
batch_normalization_12 (BatchNo	(None 50000 59)	236
max_pooling1d_1 (MaxPooling1D)	(None 1 12)	0
lambda_2 (Lambda)	(None 59 50000)	0
lambda_1 (Lambda)	(None 12 50000)	0
concatenate_1 (Concatenate)	(None 71 50000)	0
lambda_3 (Lambda)	(None 50000 71)	0
conv1d_13 (Conv1D)	(None 50000 20)	1440
conv1d_14 (Conv1D)	(None 50000 1)	201
flatten_1 (Flatten)	(None 50000)	0

lambda_4 (Lambda)	(None 50000)	0

Table C.2: Donor-Acceptor split network topology. This corresponds to parameters: $n_f = 40, l_f = 40, p = 0.07, n_c = 8, b = 100, s_k = 0.92, w = 10, r = 20, o = 10$

Appendix D

Full TOMTOM Motif matches

Matches for the motifs generated from the trained network in chapter 3

Query_ID	Target_ID	Optimal_offset	p-value	E-value	q-value	Overlap	Target_consensus	Orientation
0	RNCMPT00274	-10	3.51865e-05	0.00358903	0.00314734	7	TTTTTTT	+
0	RNCMPT00032	-10	0.000721857	0.0736294	0.021002	7	TTATTTT	+
0	RNCMPT00167	-9	0.00123668	0.126141	0.021002	7	ATTTTTT	+
0	RNCMPT00158	-10	0.00136511	0.139241	0.021002	7	CTTTTTT	+
0	RNCMPT00165	-10	0.00136511	0.139241	0.021002	7	TTTTTTC	+
0	RNCMPT00025	-9	0.0016512	0.168422	0.021002	7	ATTTTTT	+
0	RNCMPT00077	-10	0.00180693	0.184306	0.021002	7	TTTTTTG	+
0	RNCMPT00012	-10	0.00235301	0.240007	0.021002	7	CTTTTTT	+
0	RNCMPT00086	-10	0.00235301	0.240007	0.021002	7	TTTGTTT	+
0	RNCMPT00159	-10	0.00235301	0.240007	0.021002	7	TTTTTTG	+
0	RNCMPT00079	-10	0.00258277	0.263443	0.021002	7	TTTTTTC	+
0	RNCMPT00112	-10	0.00316109	0.322431	0.0235625	7	TTTGTTT	+
0	RNCMPT00117	-10	0.00460607	0.469819	0.0316923	7	TTTGTTT	+
1	RNCMPT00274	-4	5.9337e-05	0.00605237	0.00521446	7	TTTTTTT	+
1	RNCMPT00165	-3	0.000380282	0.0387887	0.0105297	7	TTTTTTC	+
1	RNCMPT00032	-3	0.000489564	0.0499355	0.0105297	7	TTATTTT	+
1	RNCMPT00167	-3	0.000489564	0.0499355	0.0105297	7	ATTTTTT	+
1	RNCMPT00025	-3	0.000655225	0.066833	0.0105297	7	ATTTTTT	+
1	RNCMPT00158	-4	0.000722337	0.0736784	0.0105297	7	CTTTTTT	+
1	RNCMPT00077	-3	0.000871826	0.0889263	0.0105297	7	TTTTTTG	+
1	RNCMPT00079	-4	0.000958563	0.0977734	0.0105297	7	TTTTTTC	+
1	RNCMPT00086	-4	0.00119058	0.121439	0.0107054	7	TTTGTTT	+
1	RNCMPT00012	-4	0.00134002	0.136682	0.0107054	7	CTTTTTT	+
1	RNCMPT00159	-3	0.00134002	0.136682	0.0107054	7	TTTTTTG	+
1	RNCMPT00112	-4	0.00170812	0.174228	0.012509	7	TTTGTTT	+
1	RNCMPT00117	-4	0.00347121	0.354063	0.023465	7	TTTGTTT	+
1	RNCMPT00013	-17	0.00456407	0.465535	0.0286489	7	TAGGTAG	+
1	RNCMPT00268	-4	0.00711355	0.725582	0.0416753	7	CTTTTCT	+
2	RNCMPT00274	-2	0.00086082	0.0878037	0.0705394	7	TTTTTTT	+
2	RNCMPT00084	-16	0.00569022	0.580402	0.102565	7	AACAACA	+
2	RNCMPT00032	-3	0.00629227	0.641811	0.102565	7	TTATTTT	+
2	RNCMPT00167	-2	0.00629227	0.641811	0.102565	7	ATTTTTT	+
2	RNCMPT00033	-20	0.00764199	0.779483	0.102565	7	ACAAACA	+
2	RNCMPT00025	-2	0.00920036	0.938436	0.102565	7	ATTTTTT	+
3	RNCMPT00268	-10	0.000121	0.012342	0.00828126	7	CTTTTCT	+
3	RNCMPT00044	-9	0.000188033	0.0191794	0.00828126	7	CCTTCCC	+
3	RNCMPT00269	-9	0.00054176	0.0552595	0.0159066	7	ACTTTCT	+

3	RNCMPT00079	-9	0.000901742	0.0919777	0.019857	7	TTTTTTC	+
3	RNCMPT00158	-10	0.00145746	0.148661	0.0256754	7	CTTTTTT	+
3	RNCMPT00186	-14	0.00175624	0.179136	0.0257824	7	CCTTTCC	+
3	RNCMPT00012	-10	0.00254402	0.25949	0.0320121	7	CTTTTTT	+
3	RNCMPT00165	-8	0.00334092	0.340774	0.0367847	7	TTTTTTC	+
3	RNCMPT00274	-11	0.00555792	0.566908	0.0543953	7	TTTTTTT	+
5	RNCMPT00158	-23	0.0044872	0.457694	0.259062	7	CTTTTTT	+
5	RNCMPT00012	-23	0.00703858	0.717936	0.259062	7	CTTTTTT	+
5	RNCMPT00268	-23	0.00827475	0.844025	0.259062	7	CTTTTCT	+
6	RNCMPT00047	-4	0.00207446	0.211595	0.210573	7	ACTAACA	+
7	RNCMPT00274	-12	2.08692e-05	0.00212866	0.00185267	7	TTTTTTT	+
7	RNCMPT00165	-15	0.00063653	0.0649261	0.0200181	7	TTTTTTC	+
7	RNCMPT00077	-12	0.000817673	0.0834027	0.0200181	7	TTTTTTG	+
7	RNCMPT00032	-12	0.00121321	0.123747	0.0200181	7	TTATTTT	+
7	RNCMPT00159	-21	0.00121321	0.123747	0.0200181	7	TTTTTTG	+
7	RNCMPT00086	-17	0.00144111	0.146993	0.0200181	7	TTTGTTT	+
7	RNCMPT00079	-15	0.00157844	0.161001	0.0200181	7	TTTTTTC	+
7	RNCMPT00112	-17	0.00206665	0.210798	0.0203853	7	TTTGTTT	+
7	RNCMPT00158	-9	0.00206665	0.210798	0.0203853	7	CTTTTTT	+
7	RNCMPT00167	-13	0.00272546	0.277996	0.0241954	7	ATTTTTT	+
7	RNCMPT00012	-9	0.00363215	0.370479	0.0248035	7	CTTTTTT	+
7	RNCMPT00025	-13	0.00363215	0.370479	0.0248035	7	ATTTTTT	+
7	RNCMPT00117	-17	0.00363215	0.370479	0.0248035	7	TTTGTTT	+
8	RNCMPT00043	-22	0.000461481	0.047071	0.0459138	7	AAAAAAA	+
8	RNCMPT00064	-21	0.00705987	0.720107	0.234134	7	AGAAAAA	+
8	RNCMPT00155	-21	0.00705987	0.720107	0.234134	7	AGAAAAA	+
9	RNCMPT00023	-20	0.000546876	0.0557814	0.0544128	7	TTAGGGA	+
9	RNCMPT00022	-20	0.00121484	0.123914	0.0604367	7	TTAGGGA	+
9	RNCMPT00024	-20	0.00253507	0.258577	0.0840777	7	TTAGGGA	+
10	RNCMPT00274	-7	0.0012469	0.127183	0.0673365	7	TTTTTTT	+
10	RNCMPT00032	-6	0.00155456	0.158565	0.0673365	7	TTATTTT	+
10	RNCMPT00004	-7	0.00299786	0.305782	0.0865693	7	TGTGTGT	+
10	RNCMPT00166	-7	0.00443969	0.452848	0.0961536	7	TGTGTGT	+
10	RNCMPT00086	-7	0.00828085	0.844646	0.115876	7	TTTGTTT	+
10	RNCMPT00112	-5	0.00902981	0.921041	0.115876	7	TTTGTTT	+
11	RNCMPT00026	-16	0.00463509	0.47278	0.470497	7	CCAACCC	+
13	RNCMPT00032	-10	0.000891532	0.0909362	0.0464457	7	TTATTTT	+
13	RNCMPT00086	-9	0.00176882	0.18042	0.0464457	7	TTTGTTT	+
13	RNCMPT00112	-9	0.00214657	0.21895	0.0464457	7	TTTGTTT	+
13	RNCMPT00117	-9	0.0025711	0.262252	0.0464457	7	TTTGTTT	+
13	RNCMPT00274	-9	0.0025711	0.262252	0.0464457	7	TTTTTTT	+
13	RNCMPT00136	-9	0.00750301	0.765307	0.112948	7	TTGGTTT	+
14	RNCMPT00274	-15	0.000301615	0.0307647	0.0198099	7	TTTTTTT	+
14	RNCMPT00032	-15	0.000452798	0.0461854	0.0198099	7	TTATTTT	+
14	RNCMPT00165	-15	0.0014766	0.150614	0.0430676	7	TTTTTTC	+
14	RNCMPT00159	-17	0.00263045	0.268306	0.0573197	7	TTTTTTG	+
14	RNCMPT00077	-15	0.00410352	0.418559	0.0573197	7	TTTTTTG	+
14	RNCMPT00079	-15	0.00410352	0.418559	0.0573197	7	TTTTTTC	+
14	RNCMPT00167	-15	0.00458557	0.467728	0.0573197	7	ATTTTTT	+
14	RNCMPT00025	-15	0.00562178	0.573421	0.0614882	7	ATTTTTT	+
14	RNCMPT00158	-14	0.00736305	0.751031	0.0715852	7	CTTTTTT	+
14	RNCMPT00012	-15	0.00936394	0.955122	0.0819345	7	CTTTTTT	+
17	RNCMPT00268	-20	0.0067332	0.686786	0.387589	7	CTTTTCT	+
18	RNCMPT00117	-16	0.00239596	0.244388	0.0834386	7	TTTGTTT	+
18	RNCMPT00112	-16	0.00287362	0.293109	0.0834386	7	TTTGTTT	+
18	RNCMPT00086	-16	0.00341481	0.348311	0.0834386	7	TTTGTTT	+
18	RNCMPT00166	-16	0.00371799	0.379235	0.0834386	7	TGTGTGT	+
18	RNCMPT00004	-16	0.00440631	0.449443	0.0834386	7	TGTGTGT	+
19	RNCMPT00037	-23	0.00148093	0.151055	0.141041	7	AATCTTG	+

19	RNCMPT00269	-7	0.00863034	0.880295	0.24773	7	ACTTTCT	+
20	RNCMPT00079	-23	0.00097411	0.0993592	0.0369607	7	TTTTTTC	+
20	RNCMPT00158	-22	0.00136138	0.138861	0.0369607	7	CTTTTTT	+
20	RNCMPT00268	-15	0.00186461	0.19019	0.0369607	7	CTTTTCT	+
20	RNCMPT00025	-22	0.00206292	0.210418	0.0369607	7	ATTTTTT	+
20	RNCMPT00167	-22	0.00206292	0.210418	0.0369607	7	ATTTTTT	+
20	RNCMPT00012	-22	0.0040949	0.417679	0.0524049	7	CTTTTTT	+
20	RNCMPT00274	-15	0.0040949	0.417679	0.0524049	7	TTTTTTT	+
20	RNCMPT00165	-23	0.00795883	0.811801	0.0891224	7	TTTTTTC	+
21	RNCMPT00070	-18	0.00180714	0.184328	0.17221	7	GATCAAG	+
22	RNCMPT00079	-9	9.08752e-07	9.26927e-05	8.02654e-05	7	TTTTTTC	+
22	RNCMPT00165	-9	5.38416e-05	0.00549184	0.00237778	7	TTTTTTC	+
22	RNCMPT00274	-8	0.000184038	0.0187719	0.00541838	7	TTTTTTT	+
22	RNCMPT00268	-10	0.00149654	0.152647	0.0309875	7	CTTTTCT	+
22	RNCMPT00032	-8	0.00181621	0.185254	0.0309875	7	TTATTTT	+
22	RNCMPT00158	-8	0.00210501	0.214711	0.0309875	7	CTTTTTT	+
22	RNCMPT00167	-7	0.00290878	0.296696	0.0345977	7	ATTTTTT	+
22	RNCMPT00012	-8	0.00318852	0.325229	0.0345977	7	CTTTTTT	+
22	RNCMPT00077	-8	0.00352538	0.359589	0.0345977	7	TTTTTTG	+
22	RNCMPT00159	-9	0.00393038	0.400899	0.034715	7	TTTTTTG	+
22	RNCMPT00025	-7	0.0043973	0.448525	0.0353083	7	ATTTTTT	+
22	RNCMPT00086	-8	0.0078043	0.796039	0.0530242	7	TTTGTTT	+
22	RNCMPT00112	-8	0.0078043	0.796039	0.0530242	7	TTTGTTT	+
22	RNCMPT00269	-10	0.00865289	0.882595	0.0545904	7	ACTTTCT	+
24	RNCMPT00156	-15	0.00359783	0.366979	0.304468	7	GACAGAT	+
27	RNCMPT00167	-20	4.75239e-05	0.00484744	0.00393077	7	ATTTTTT	+
27	RNCMPT00025	-20	8.90072e-05	0.00907873	0.00393077	7	ATTTTTT	+
27	RNCMPT00274	-20	0.000398792	0.0406768	0.0117411	7	TTTTTTT	+
27	RNCMPT00079	-21	0.00116818	0.119154	0.0206358	7	TTTTTTC	+
27	RNCMPT00165	-21	0.00116818	0.119154	0.0206358	7	TTTTTTC	+
27	RNCMPT00158	-20	0.00265353	0.27066	0.0332921	7	CTTTTTT	+
27	RNCMPT00077	-20	0.00290198	0.296002	0.0332921	7	TTTTTTG	+
27	RNCMPT00032	-21	0.00314778	0.321074	0.0332921	7	TTATTTT	+
27	RNCMPT00012	-20	0.00339235	0.34602	0.0332921	7	CTTTTTT	+
27	RNCMPT00159	-21	0.00412676	0.42093	0.0351889	7	TTTTTTG	+
27	RNCMPT00086	-21	0.00438243	0.447008	0.0351889	7	TTTGTTT	+
27	RNCMPT00112	-21	0.00560116	0.571319	0.0412268	7	TTTGTTT	+
27	RNCMPT00117	-21	0.00951508	0.970538	0.0639216	7	TTTGTTT	+
28	RNCMPT00052	-4	0.0013448	0.13717	0.110867	7	GCGCGGG	+
28	RNCMPT00044	-20	0.00407604	0.415756	0.110867	7	CCTTCCC	+
28	RNCMPT00026	-16	0.00450256	0.459261	0.110867	7	CCAACCC	+
28	RNCMPT00113	-4	0.00450256	0.459261	0.110867	7	GCGCGGG	+
29	RNCMPT00176	-20	0.00832537	0.849187	0.493192	7	TAGTTAG	+
30	RNCMPT00268	-24	0.00208901	0.213079	0.187746	6	CTTTTCT	+
33	RNCMPT00047	-18	0.00311749	0.317984	0.31493	7	ACTAACA	+
34	RNCMPT00153	-12	0.00152701	0.155755	0.144664	7	GAAAACC	+
35	RNCMPT00269	-19	0.00213706	0.21798	0.206886	7	ACTTTCT	+
36	RNCMPT00164	-5	0.00921356	0.939784	0.826908	7	TGTTCGT	+
37	RNCMPT00166	-20	0.00338324	0.345091	0.205002	7	TGTGTGT	+
37	RNCMPT00004	-20	0.00410005	0.418205	0.205002	7	TGTGTGT	+
37	RNCMPT00184	-20	0.00936432	0.95516	0.312144	7	AGTGTGA	+
38	RNCMPT00079	0	4.00001e-05	0.00408001	0.00223548	7	TTTTTTC	+
38	RNCMPT00274	0	5.08764e-05	0.00518939	0.00223548	7	TTTTTTT	+
38	RNCMPT00165	0	9.92126e-05	0.0101197	0.00290623	7	TTTTTTC	+
38	RNCMPT00158	-2	0.000792233	0.0808078	0.0165786	7	CTTTTTT	+
38	RNCMPT00167	0	0.00113986	0.116266	0.0165786	7	ATTTTTT	+
38	RNCMPT00077	0	0.0013865	0.141423	0.0165786	7	TTTTTTG	+
38	RNCMPT00012	-5	0.00153654	0.156728	0.0165786	7	CTTTTTT	+
38	RNCMPT00025	0	0.00170323	0.173729	0.0165786	7	ATTTTTT	+

38	RNCMPT00268	-1	0.00170323	0.173729	0.0165786	7	CTTTTCT	+
38	RNCMPT00159	0	0.00188653	0.192426	0.0165786	7	TTTTTTG	+
38	RNCMPT00032	-5	0.0023104	0.235661	0.0184578	7	TTATTTT	+
38	RNCMPT00086	0	0.00351711	0.358745	0.0257566	7	TTTGTTT	+
38	RNCMPT00112	0	0.00581478	0.593107	0.0364997	7	TTTGTTT	+
38	RNCMPT00269	0	0.00581478	0.593107	0.0364997	7	ACTTTCT	+
38	RNCMPT00117	0	0.00768691	0.784065	0.0450344	7	TTTGTTT	+
39	RNCMPT00274	-10	0.00409812	0.418009	0.115551	7	TTTTTTT	+
39	RNCMPT00159	-11	0.00481696	0.49133	0.115551	7	TTTTTTG	+
39	RNCMPT00165	-10	0.00563481	0.574751	0.115551	7	TTTTTTC	+
39	RNCMPT00025	-9	0.00770136	0.785538	0.115551	7	ATTTTTT	+
39	RNCMPT00167	-9	0.00770136	0.785538	0.115551	7	ATTTTTT	+
39	RNCMPT00077	-11	0.00833026	0.849687	0.115551	7	TTTTTTG	+
39	RNCMPT00079	-10	0.0090144	0.919469	0.115551	7	TTTTTTC	+
40	RNCMPT00274	-11	2.39949e-05	0.00244748	0.00211648	7	TTTTTTT	+
40	RNCMPT00165	-11	0.000796693	0.0812627	0.0137029	7	TTTTTTC	+
40	RNCMPT00032	-11	0.000900892	0.091891	0.0137029	7	TTATTTT	+
40	RNCMPT00159	-13	0.000900892	0.091891	0.0137029	7	TTTTTTG	+
40	RNCMPT00086	-7	0.00108303	0.110469	0.0137029	7	TTTGTTT	+
40	RNCMPT00077	-13	0.00115669	0.117983	0.0137029	7	TTTTTTG	+
40	RNCMPT00079	-11	0.00115669	0.117983	0.0137029	7	TTTTTTC	+
40	RNCMPT00158	-11	0.00124282	0.126767	0.0137029	7	CTTTTTT	+
40	RNCMPT00112	-7	0.00158492	0.161662	0.0152716	7	TTTGTTT	+
40	RNCMPT00167	-11	0.00173275	0.176741	0.0152716	7	ATTTTTT	+
40	RNCMPT00012	-11	0.00190452	0.194261	0.0152716	7	CTTTTTT	+
40	RNCMPT00025	-11	0.00232002	0.236642	0.0170532	7	ATTTTTT	+
40	RNCMPT00117	-7	0.0028687	0.292607	0.0194642	7	TTTGTTT	+
41	RNCMPT00044	-6	0.000394343	0.040223	0.0400288	7	CCTTCCC	+
43	RNCMPT00274	-14	3.84326e-05	0.00392012	0.00341623	7	TTTTTTT	+
43	RNCMPT00159	-16	0.00035591	0.0363028	0.0114893	7	TTTTTTG	+
43	RNCMPT00165	-14	0.000419429	0.0427817	0.0114893	7	TTTTTTC	+
43	RNCMPT00077	-16	0.000646271	0.0659196	0.0114893	7	TTTTTTG	+
43	RNCMPT00079	-16	0.000646271	0.0659196	0.0114893	7	TTTTTTC	+
43	RNCMPT00032	-13	0.00099277	0.101263	0.0120886	7	TTATTTT	+
43	RNCMPT00167	-7	0.00099277	0.101263	0.0120886	7	ATTTTTT	+
43	RNCMPT00025	-7	0.00108797	0.110973	0.0120886	7	ATTTTTT	+
43	RNCMPT00158	-14	0.00129818	0.132414	0.0128215	7	CTTTTTT	+
43	RNCMPT00012	-14	0.00183593	0.187265	0.0163194	7	CTTTTTT	+
43	RNCMPT00086	-15	0.00222479	0.226929	0.0179781	7	TTTGTTT	+
43	RNCMPT00112	-15	0.00291309	0.297135	0.0215784	7	TTTGTTT	+
43	RNCMPT00117	-15	0.00518361	0.528728	0.0354435	7	TTTGTTT	+
43	RNCMPT00268	-15	0.00868444	0.885813	0.0551393	7	CTTTTCT	+
45	RNCMPT00165	-19	6.00382e-05	0.0061239	0.00313302	7	TTTTTTC	+
45	RNCMPT00079	-19	7.22437e-05	0.00736886	0.00313302	7	TTTTTTC	+
45	RNCMPT00274	-18	0.00059969	0.0611684	0.017338	7	TTTTTTT	+
45	RNCMPT00032	-5	0.000822771	0.0839226	0.0178407	7	TTATTTT	+
45	RNCMPT00086	-4	0.00149169	0.152152	0.0258762	7	TTTGTTT	+
45	RNCMPT00159	-19	0.00217885	0.222242	0.029439	7	TTTTTTG	+
45	RNCMPT00112	-4	0.00244023	0.248903	0.029439	7	TTTGTTT	+
45	RNCMPT00117	-4	0.00305473	0.311583	0.029439	7	TTTGTTT	+
45	RNCMPT00268	-20	0.00305473	0.311583	0.029439	7	CTTTTCT	+
45	RNCMPT00167	-23	0.00472752	0.482207	0.041004	7	ATTTTTT	+
45	RNCMPT00077	-19	0.00525286	0.535791	0.0414186	7	TTTTTTG	+
45	RNCMPT00025	-23	0.00651867	0.664905	0.0471162	7	ATTTTTT	+
45	RNCMPT00158	-16	0.00725619	0.740131	0.0484126	7	CTTTTTT	+
45	RNCMPT00012	-16	0.00823918	0.840396	0.0510445	7	CTTTTTT	+
47	RNCMPT00013	-18	0.00173847	0.177323	0.176467	7	TAGGTAG	+
49	RNCMPT00112	-17	0.00500088	0.51009	0.119728	7	TTTGTTT	+
49	RNCMPT00117	-17	0.00602241	0.614286	0.119728	7	TTTGTTT	+

49	RNCMPT00156	-23	0.00659363	0.67255	0.119728	7	GACAGAT	+
49	RNCMPT00177	-17	0.00659363	0.67255	0.119728	7	GTAGTGT	+
49	RNCMPT00176	-14	0.0072128	0.735706	0.119728	7	TAGTTAG	+
49	RNCMPT00086	-17	0.00863147	0.880409	0.119728	7	TTTGTTT	+
49	RNCMPT00041	-14	0.00944187	0.963071	0.119728	7	TAGTTAG	+

Appendix E

HGMD Splice Mutations

The following is a complete list of deep intronic HGMD mutations that are predicted to impact splicing and a level > 99.9th percentile of the null distribution.

chrom	pos	score	ref	alt	phenotype
1	236971838	1.4585497	A	G	Methionine_synthase_deficiency
1	236977232	0.21340674	G	A	Methionine_synthase_deficiency
1	94484002	0.23967469	G	T	Stargardt_disease
1	94483979	0.099143505	C	T	Retinitis_pigmentosa
1	64113966	0.04078549	G	T	Phosphoglucomutase_deficiency
3	136003251	0.025791496	A	G	Propionic_acidaemia
3	193335986	0.030784547	G	A	Optic_atrophy_1
3	180367941	0.11544144	T	C	Primary_ciliary_dyskinesia
4	108945190	0.19937253	G	T	Hypoglycaemia_hyperinsulinaemic
5	35867853	0.07952893	G	A	Immunodeficiency_severe_combined
5	112158423	0.4957204	A	T	Adenomatous_polyposis_coli
5	112115546	0.18397343	G	A	Adenomatous_polyposis_coli
5	112158419	2.173078	C	T	Adenomatous_polyposis_coli
5	42700896	0.022613883	A	G	Growth_hormone_insensitivity
5	60223645	0.02764368	T	C	Cockayne_syndrome

5	7883859	0.03769523	T	C	Homocystinuria
6	131901748	0.05910611	T	C	Hyperargininaemia
6	112381431	0.10338867	G	T	Pseudorheumatoid_dysplasia_progressive
7	117288374	0.050453305	A	G	Cystic_fibrosis
7	117229521	0.07985395	A	G	Cystic_fibrosis
7	117218381	0.30219054	A	G	Cystic_fibrosis
8	103249078	0.11028433	G	C	Altered_p53_binding
9	108368857	0.18771525	G	T	Muscular_dystrophy_Fukuyama
11	108138753	0.10345924	A	G	Ataxia_telangiectasia
12	48379984	0.21462515	C	T	Stickler_syndrome
12	44178047	3.8379965	A	G	IRAK4_deficiency
13	32919384	1.1488519	T	G	Breast_cancer
13	49046098	0.02876997	A	G	Retinoblastoma
13	48937921	1.4977894	T	G	Retinoblastoma
16	2107460	0.8555822	C	T	Tuberous_sclerosis
16	163667	0.059475392	G	A	Reduced_activity
16	50813428	0.39161205	A	G	Brooke-Spiegler_syndrome
17	29661577	0.022499979	A	G	Neurofibromatosis_1
17	29488136	0.048462063	T	G	Neurofibromatosis_1
19	38997317	0.24481273	G	A	Myopathy_congenital

Bibliography

Amr Mohamed Alexandari, Avanti Shrikumar, and Anshul Kundaje. Separable fully connected layers improve deep learning models for genomics. *BioRxiv*, page 146431, 2017.

Babak Alipanahi, Andrew Delong, Matthew T Weirauch, and Brendan J Frey. Predicting the sequence specificities of dna-and rna-binding proteins by deep learning. *Nature biotechnology*, 33(8):831, 2015a.

Babak Alipanahi, Andrew Delong, Matthew T Weirauch, and Brendan J Frey. Predicting the sequence specificities of dna- and rna-binding proteins by deep learning. *Nature Biotechnology*, 33:831 – 838, 2015b. doi: http://dx.doi.org/10.1038/nbt.3300.

Mariano Alló, Valeria Buggiano, Juan P Fededa, Ezequiel Petrillo, Ignacio Schor, Manuel De La Mata, Eneritz Agirre, Mireya Plass, Eduardo Eyras, Sherif Abou Elela, et al. Control of alternative splicing through sirna-mediated transcriptional gene silencing. *Nature structural & molecular biology*, 16(7):717, 2009.

Nicolas Altemose, Nudrat Noor, Emmanuelle Bitoun, Afidalina Tumian, Michael Imbeault, J Ross Chapman, A Radu Aricescu, and Simon R Myers. A map of human prdm9 binding provides evidence for novel behaviors of prdm9 and other zinc-finger proteins in meiosis. *eLife*, 6:e28383, oct 2017. ISSN 2050-084X. doi: 10.7554/eLife.28383. URL `https://doi.org/10.7554/eLife.28383`.

Maayan Amit, Maya Donyo, Dror Hollander, Amir Goren, Eddo Kim, Sahar Gelfman, Galit Lev-Maor, David Burstein, Schraga Schwartz, Benny Postolsky, et al. Differential gc content between exons and introns establishes distinct strategies of splice-site recognition. *Cell reports*, 1(5):543–556, 2012.

Olga Anczuków, Monique Buisson, Mélanie Léoné, Christine Coutanson, Christine Lasset, Alain Calender, Olga M Sinilnikova, and Sylvie Mazoyer. Brca2 deep intronic mutation causing activation of a cryptic exon: opening toward a new preventive therapeutic strategy. *Clinical Cancer Research*, 18(18):4903–4909, 2012.

Robin Andersson, Stefan Enroth, Alvaro Rada-Iglesias, Claes Wadelius, and Jan Komorowski. Nucleosomes are well positioned in exons and carry characteristic histone modifications. *Genome research*, pages gr–092353, 2009.

Gil Ast. How did alternative splicing evolve? *Nature Reviews Genetics*, 5(10):773, 2004.

Catia Attanasio, Armelle David, and Marguerite Neerman-Arbez. Outcome of donor splice site mutations accounting for congenital afibrinogenemia reflects order of intron removal in the fibrinogen alpha gene (fga). *Blood*, 101(5):1851–1856, 2003.

Akinori Awazu. Prediction of nucleosome positioning by the incorporation of frequencies and distributions of three different nucleotide segment lengths into a general pseudo k-tuple nucleotide composition. *Bioinformatics*, 33(1):42–48, 2016.

Yoseph Barash, John A Calarco, Weijun Gao, Qun Pan, Xinchen Wang, Ofer Shai, Benjamin J Blencowe, and Brendan J Frey. Deciphering the splicing code. *Nature*, 465(7294):53, 2010.

Jakub M Bartoszewicz, Anja Seidel, Robert Rentzsch, and Bernhard Y Renard. DeePaC: predicting pathogenic potential of novel DNA with reverse-complement neural networks. *Bioinformatics*, 07 2019. ISSN 1367-4803. doi: 10.1093/bioinformatics/btz541. URL `https://doi.org/10.1093/bioinformatics/btz541`.

Alexis Battle, Sara Mostafavi, Xiaowei Zhu, James B Potash, Myrna M Weissman, Courtney McCormick, Christian D Haudenschild, Kenneth B Beckman, Jianxin Shi, Rui Mei, et al. Characterizing the genetic basis of transcriptome diversity through rna-sequencing of 922 individuals. *Genome research*, 24(1):14–24, 2014.

Göran Baurén and Lars Wieslander. Splicing of balbiani ring 1 gene pre-mrna occurs simultaneously with transcription. *Cell*, 76(1):183–192, 1994.

Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *Journal of Machine Learning Research*, 18(153):1–43, 2018. URL `http://jmlr.org/papers/v18/17-468.html`.

Ilan Ben-Bassat, Benny Chor, and Yaron Orenstein. A deep neural network approach for learning intrinsic protein-rna binding preferences. *Bioinformatics*, 34(17):i638–i646, 2018.

Asa Ben-Hur and William Stafford Noble. Choosing negative examples for the prediction of protein-protein interactions. In *BMC bioinformatics*, volume 7, page S2. BioMed Central, 2006.

Susan M Berget. Exon recognition in vertebrate splicing. *Journal of biological Chemistry*, 270(6):2411–2414, 1995.

James Bergstra, Daniel Yamins, and David Daniel Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. 2013.

James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in neural information processing systems*, pages 2546–2554, 2011.

Ann L Beyer and Yvonne N Osheim. Splice site selection, rate of splicing, and

alternative splicing on nascent transcripts. *Genes & development*, 2(6):754–765, 1988.

Ann L Beyer, Amy H Bouton, and Oscar L Miller Jr. Correlation of hnrnp structure and nascent transcript cleavage. *Cell*, 26(2):155–165, 1981.

Hannes Bretschneider, Shreshth Gandhi, Amit G Deshwar, Khalid Zuberi, and Brendan J Frey. Cossmo: predicting competitive alternative splice site selection using deep learning. *Bioinformatics*, 34(13):i429–i437, 2018.

Richard C Brown and Gerton Lunter. An equivariant bayesian convolutional network predicts recombination hotspots and accurately resolves binding motifs. *Bioinformatics*, 35(13):2177–2184, 2018.

Peter T Buckley, Miler T Lee, Jai-Yoon Sul, Kevin Y Miyashiro, Thomas J Bell, Stephen A Fisher, Junhyong Kim, and James Eberwine. Cytoplasmic intron sequence-retaining transcripts can be dendritically targeted via id element retrotransposons. *Neuron*, 69(5):877–884, 2011.

Chris Burge and Samuel Karlin. Prediction of complete gene structures in human genomic dna. *Journal of molecular biology*, 268(1):78–94, 1997.

Eva Fernández Cáceres and Laurence D Hurst. The evolution, impact and properties of exonic splice enhancers. *Genome biology*, 14(12):R143, 2013.

Luca Cartegni, Jinhua Wang, Zhengwei Zhu, Michael Q Zhang, and Adrian R Krainer. Esefinder: a web resource to identify exonic splicing enhancers. *Nucleic acids research*, 31(13):3568–3571, 2003.

Donghang Cheng, Jocelyn Côté, Salam Shaaban, and Mark T Bedford. The arginine methyltransferase carm1 regulates the coupling of transcription and mrna processing. *Molecular cell*, 25(1):71–83, 2007.

Vicky Cho, Yan Mei, Arleen Sanny, Stephanie Chan, Anselm Enders, Edward M Bertram, Andy Tan, Christopher C Goodnow, and T Daniel Andrews. The rna-binding protein hnrnpll induces a t cell alternative splicing program delineated by differential intron retention in polyadenylated rna. *Genome biology*, 15(1):R26, 2014.

Ramakrishna K Chodavarapu, Suhua Feng, Yana V Bernatavichute, Pao-Yang Chen, Hume Stroud, Yanchun Yu, Jonathan A Hetzel, Frank Kuo, Jin Kim, Shawn J Cokus, et al. Relationship between nucleosome positioning and dna methylation. *Nature*, 466(7304):388, 2010.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Gated feedback recurrent neural networks. In *International Conference on Machine Learning*, pages 2067–2075, 2015.

L Stirling Churchman and Jonathan S Weissman. Nascent transcript sequencing visualizes transcription at nucleotide resolution. *Nature*, 469(7330):368, 2011.

Jean-Michel Claverie and Stephane Audic. The statistical significance of nucleotide position-weight matrix matches. *Bioinformatics*, 12(5):431–439, 1996.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *CoRR*, abs/1511.07289, 2015. URL `http://arxiv.org/abs/1511.07289`.

Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999, 2016a.

Taco S. Cohen and Max Welling. Group equivariant convolutional networks. *CoRR*, abs/1602.07576, 2016b. URL `http://arxiv.org/abs/1602.07576`.

André Corvelo, Martina Hallegger, Christopher WJ Smith, and Eduardo Eyras. Genome-wide association between branch point properties and alternative splicing. *PLoS computational biology*, 6(11):e1001016, 2010.

Rita Das, Jiong Yu, Zuo Zhang, Melanie P Gygi, Adrian R Krainer, Steven P Gygi, and Robin Reed. Sr proteins function in coupling rnap ii transcription to pre-mrna splicing. *Molecular cell*, 26(6):867–881, 2007.

Charles J David, Alex R Boyne, Scott R Millhouse, and James L Manley. The rna polymerase ii c-terminal domain promotes splicing activation through recruitment of a u2af65–prp19 complex. *Genes & development*, 25(9):972–983, 2011.

Manuel de la Mata and Alberto R Kornblihtt. Rna polymerase ii c-terminal domain mediates regulation of alternative splicing by srp20. *Nature Structural and Molecular Biology*, 13(11):973, 2006.

Robert B Denman. Using rnafold to predict the activity of small catalytic rnas. *Biotechniques*, 15(6):1090–1095, 1993.

Thomas Derrien, Rory Johnson, Giovanni Bussotti, Andrea Tanzer, Sarah Djebali, Hagen Tilgner, Gregory Guernec, David Martin, Angelika Merkel, David G Knowles, et al. The gencode v7 catalog of human long noncoding rnas: analysis of their gene structure, evolution, and expression. *Genome research*, 22(9):1775–1789, 2012.

Pawandeep Dhami, Peter Saffrey, Alexander W Bruce, Shane C Dillon, Kelly Chiang, Nicolas Bonhoure, Christoph M Koch, Jackie Bye, Keith James, Nicola S Foad, et al. Complex exon-intron marking by histone modifications is not determined solely by nucleosome distribution. *PloS one*, 5(8):e12339, 2010.

Petr Divina, Andrea Kvitkovicova, Emanuele Buratti, and Igor Vorechovsky. Ab initio prediction of mutation-induced cryptic splice-site activation and exon skipping. *European Journal of Human Genetics*, 17(6):759, 2009.

Alexander Dobin, Carrie A Davis, Felix Schlesinger, Jorg Drenkow, Chris Zaleski, Sonali Jha, Philippe Batut, Mark Chaisson, and Thomas R Gingeras. Star: ultrafast universal rna-seq aligner. *Bioinformatics*, 29(1):15–21, 2013.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12 (Jul):2121–2159, 2011.

Gwendal Dujardin, Celina Lafaille, Manuel de la Mata, Luciano E Marasco, Manuel J Muñoz, Catherine Le Jossic-Corcos, Laurent Corcos, and Alberto R Kornblihtt. How slow rna polymerase ii elongation favors alternative exon skipping. *Molecular cell*, 54(4):683–690, 2014.

Laurent Duret and Nicolas Galtier. Biased gene conversion and the evolution of mammalian genomic landscapes. *Annual review of genomics and human genetics*, 10:285–311, 2009.

Martin Dutertre, Gabriel Sanchez, Marie-Cécile De Cian, Jérôme Barbier, Etienne Dardenne, Lise Gratadou, Gwendal Dujardin, Catherine Le Jossic-Corcos, Laurent Corcos, and Didier Auboeuf. Cotranscriptional exon skipping in the genotoxic stress response. *Nature structural & molecular biology*, 17(11):1358, 2010.

Heidi Dvinge and Robert K Bradley. Widespread intron retention diversifies most cancer transcriptomes. *Genome medicine*, 7(1):45, 2015.

ENCODE Project Consortium and others. An integrated encyclopedia of dna elements in the human genome. *Nature*, 489(7414):57, 2012.

Lucy P Eperon, Ian R Graham, Andrew D Griffiths, and Ian C Eperon. Effects of rna secondary structure on alternative splicing of pre-mrna: is folding limited to a region behind the transcribing rna polymerase? *Cell*, 54(3):393–401, 1988.

Maria Eriksson, W Ted Brown, Leslie B Gordon, Michael W Glynn, Joel Singer, Laura Scott, Michael R Erdos, Christiane M Robbins, Tracy Y Moses, Peter Berglund, et al. Recurrent de novo point mutations in lamin a cause hutchinson–gilford progeria syndrome. *Nature*, 423(6937):293, 2003.

William G Fairbrother, Dirk Holste, Christopher B Burge, and Phillip A Sharp. Single nucleotide polymorphism–based validation of exonic splicing enhancers. *PLoS biology*, 2(9):e268, 2004.

Alistair RR Forrest, Hideya Kawaji, Michael Rehli, J Kenneth Baillie, Michiel JL De Hoon, Vanja Haberle, Timo Lassmann, Ivan V Kulakovskiy, Marina Lizio, Masayoshi Itoh, et al. A promoter-level mammalian expression atlas. *Nature*, 507(7493):462, 2014.

Adam Frankish, Barbara Uszczynska, Graham RS Ritchie, Jose M Gonzalez, Dmitri Pervouchine, Robert Petryszak, Jonathan M Mudge, Nuno Fonseca, Alvis Brazma, Roderic Guigo, et al. Comparison of gencode and refseq gene annotation and the impact of reference geneset on variant effect prediction. *BMC genomics*, 16(8):S2, 2015.

Luke Frankiw, David Baltimore, and Guideng Li. Alternative mrna splicing in cancer immunotherapy. *Nature Reviews Immunology*, page 1, 2019.

Marcel Freund, Corinna Asang, Susanne Kammler, Carolin Konermann, Jörg Krummheuer, Marianne Hipp, Imke Meyer, Wolfram Gierling, Stephan Theiss, Thorsten Preuss, et al. A novel approach to describe a u1 snrna binding site. *Nucleic acids research*, 31(23):6963–6975, 2003.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 1050–1059, 2016. URL `http://jmlr.org/proceedings/papers/v48/gal16.html`.

Pedro Alexandre Favoretto Galante, Noboru Jo Sakabe, Natanja Kirschbaum-Slager, and Sandro José de Souza. Detection and evaluation of intron retention events in the human transcriptome. *Rna*, 10(5):757–765, 2004.

Walter Gilbert. Why genes in pieces? *Nature*, 271(5645):501, 1978.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, pages 249–256, 2010a. URL `http://www.jmlr.org/proceedings/papers/v9/glorot10a.html`.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *JMLR W&CP: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2010)*, volume 9, pages 249–256, May 2010b.

Valerie Goguel, Y Wang, and M Rosbash. Short artificial hairpins sequester splicing signals and inhibit yeast pre-mrna splicing. *Molecular and cellular biology*, 13(11): 6841–6848, 1993.

BRENTON R GRAVELEY, KLEMENS J HERTEL, and TOM Maniatis. The role of u2af 35 and u2af 65 in enhancer-dependent splicing. *Rna*, 7(6):806–818, 2001.

Celeste B Greer, Yoshiaki Tanaka, Yoon Jung Kim, Peng Xie, Michael Q Zhang, In-Hyun Park, and Tae Hoon Kim. Histone deacetylases positively regulate transcription through the elongation machinery. *Cell reports*, 13(7):1444–1455, 2015.

Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232, 2016.

GTEx Consortium and others. The genotype-tissue expression (gtex) pilot analysis: multitissue gene regulation in humans. *Science*, 348(6235):648–660, 2015.

Jihua Guo and Rong Jia. Splicing factor poly (rc)-binding protein 1 is a novel and distinctive tumor suppressor. *Journal of cellular physiology*, 234(1):33–41, 2019.

Shobhit Gupta, John A Stamatoyannopoulos, Timothy L Bailey, and William Stafford Noble. Quantifying similarity between motifs. *Genome biology*, 8(2):R24, 2007.

Hartmut Halfter and Dieter Gallwitz. Impairment of yeast pre-mrna splicing by potential secondary structure-forming sequences near the conserved branchpoint sequence. *Nucleic acids research*, 16(22):10413–10423, 1988.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015. URL `http://arxiv.org/abs/1502.01852`.

Steffen Heber, Max Alekseyev, Sing-Hoi Sze, Haixu Tang, and Pavel A Pevzner. Splicing graphs and est assembly problem. *Bioinformatics*, 18(suppl_1):S181–S188, 2002.

S. Heinz, C. Benner, N. Spann, E. Bertolino, Y. C. Lin, P. Laslo, J. X. Cheng, C. Murre, H. Singh, and C. K. Glass. Simple combinations of lineage-determining transcription factors prime cis-regulatory elements required for macrophage and b cell identities. *Mol Cell*, 38(4):576–589, 2010.

Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

Lydia Herzel, Diana SM Ottoz, Tara Alpert, and Karla M Neugebauer. Splicing and transcription touch base: co-transcriptional spliceosome assembly and function. *Nature reviews Molecular cell biology*, 18(10):637, 2017.

Yutaka Hirose, Roland Tacke, and James L Manley. Phosphorylated rna polymerase ii stimulates pre-mrna splicing. *Genes & development*, 13(10):1234–1239, 1999.

Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

Sepp Hochreiter and Jürgen Schmidhuber. Lstm can solve hard long time lag problems. In *Advances in neural information processing systems*, pages 473–479, 1997.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *International conference on learning and intelligent optimization*, pages 507–523. Springer, 2011.

Camilla Iannone, Andy Pohl, Panagiotis Papasaikas, Daniel Soronellas, Guillermo P Vicent, Miguel Beato, and Juan Valcárcel. Relationship between nucleosome positioning and progesterone-induced alternative splicing in breast cancer cells. *RNA*, 2015.

El Cherif Ibrahim, Matthew M Hims, Noam Shomron, Christopher B Burge, Susan A Slaugenhaupt, and Robin Reed. Weak definition of ikbkap exon 20 leads to aberrant splicing in familial dysautonomia. *Human mutation*, 28(1):41–53, 2007.

International Human Genome Sequencing Consortium and others. Initial sequencing and analysis of the human genome. *nature*, 409(6822):860, 2001.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 448–456. JMLR.org, 2015. URL `http://dl.acm.org/citation.cfm?id=3045118.3045167`.

Joanna Y Ip, Dominic Schmidt, Qun Pan, Arun K Ramani, Andrew G Fraser, Duncan T Odom, and Benjamin J Blencowe. Global impact of rna polymerase ii elongation inhibition on alternative splicing regulation. *Genome research*, 21(3):390–401, 2011.

Laurent Jacob and Jean-Philippe Vert. Protein-ligand interaction prediction: an improved chemogenomics approach. *Bioinformatics*, 24(19):2149–2156, 2008.

Faegheh Jafarifar, Rosemary C Dietrich, James M Hiznay, and Richard A Padgett. Biochemical defects in minor spliceosome function in the developmental disorder mopd i. *RNA*, 20(7):1078–1089, 2014.

Kishore Jaganathan, Sofia Kyriazopoulou Panagiotopoulou, Jeremy F McRae, Siavash Fazel Darbandi, David Knowles, Yang I Li, Jack A Kosmicki, Juan Arbelaez, Wenwu Cui, Grace B Schwartz, et al. Predicting splicing from primary sequence with deep learning. *Cell*, 176(3):535–548, 2019.

Sebastián Kadener, Paula Cramer, Guadalupe Nogués, Demián Cazalla, Manuel de la Mata, Juan Pablo Fededa, Santiago E Werbajh, Anabella Srebrow, and Alberto R Kornblihtt. Antagonistic effects of t-ag and vp16 reveal a role for rna pol ii elongation on alternative splicing. *The EMBO journal*, 20(20):5759–5768, 2001.

Sebastián Kadener, Juan Pablo Fededa, Michael Rosbash, and Alberto R Kornblihtt. Regulation of alternative splicing by a transcriptional enhancer through rna pol ii elongation. *Proceedings of the National Academy of Sciences*, 99(12):8185–8190, 2002.

Stephen G Kaler, Linda K Gallo, Virginia K Proud, Alan K Percy, Yvonne Mark, Neil A Segal, David S Goldstein, Courtney S Holmes, and William A Gahl. Occipital horn syndrome and a mild menkes phenotype associated with splice site mutations at the mnk locus. *Nature genetics*, 8(2):195, 1994.

Yarden Katz, Eric T Wang, Edoardo M Airoldi, and Christopher B Burge. Analysis and design of rna sequencing experiments for identifying isoform regulation. *Nature methods*, 7(12):1009, 2010.

David R Kelley, Jasper Snoek, and John Rinn. Basset: Learning the regulatory code

of the accessible genome with deep convolutional neural networks. *Genome Res*, 26:990 – 999, 2016. doi: 10.1101/gr.200535.115.

O Kessler, YAJUAN Jiang, and LA Chasin. Order of intron removal during splicing of endogenous adenine phosphoribosyltransferase and dihydrofolate reductase pre-mrna. *Molecular and cellular biology*, 13(10):6211–6222, 1993.

Daehwan Kim, Ben Langmead, and Steven L Salzberg. Hisat: a fast spliced aligner with low memory requirements. *Nature methods*, 12(4):357, 2015.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Bernhard Kräutler. Organometallic chemistry of b12 coenzymes. *Metal ions in life sciences*, 6:1–51, 2009.

Rajan Lamichhane, Gerrit M Daubner, Judith Thomas-Crusells, Sigrid D Auweter, Cristina Manatschal, Keyunna S Austin, Oksana Valniuk, Frédéric H-T Allain, and David Rueda. Rna looping by ptb: Evidence using fret and nmr spectroscopy for a role in splicing repression. *Proceedings of the National Academy of Sciences*, 107 (9):4105–4110, 2010.

Jack Lanchantin, Ritambhara Singh, Zeming Lin, and Yanjun Qi. Deep motif: Visualizing genomic sequence classifications. *CoRR*, abs/1605.01133, 2016. URL `http://arxiv.org/abs/1605.01133`.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404, 1990.

Yann LeCun, Fu Jie Huang, Leon Bottou, et al. Learning methods for generic object recognition with invariance to pose and lighting. In *CVPR (2)*, pages 97–104. Citeseer, 2004.

Monkol Lek, Konrad J Karczewski, Eric V Minikel, Kaitlin E Samocha, Eric Banks, Timothy Fennell, Anne H O'Donnell-Luria, James S Ware, Andrew J Hill, Beryl B Cummings, et al. Analysis of protein-coding genetic variation in 60,706 humans. *Nature*, 536(7616):285, 2016.

Christina Leslie, Eleazar Eskin, and William Stafford Noble. The spectrum kernel: A string kernel for svm protein classification. In *Biocomputing 2002*, pages 564–575. World Scientific, 2001.

Christina S Leslie, Eleazar Eskin, Adiel Cohen, Jason Weston, and William Stafford Noble. Mismatch string kernels for discriminative protein classification. *Bioinformatics*, 20(4):467–476, 2004.

Michael KK Leung, Hui Yuan Xiong, Leo J Lee, and Brendan J Frey. Deep learning of the tissue-regulated splicing code. *Bioinformatics*, 30(12):i121–i129, 2014.

Galit Lev-Maor, Amir Goren, Noa Sela, Eddo Kim, Hadas Keren, Adi Doron-Faigenboim, Shelly Leibman-Barak, Tal Pupko, and Gil Ast. The "alternative" choice of constitutive exons throughout evolution. *PLoS genetics*, 3(11):e203, 2007.

Leping Li, Yu Liang, and Robert L Bass. Gapwm: a genetic algorithm method for optimizing a position weight matrix. *Bioinformatics*, 23(10):1188–1194, 2007.

Xiao-Wei Li, Bing-Yu Shi, Qing-Lan Yang, Jie Wu, Hui-Min Wu, Yu-Feng Wang, Zhi-Jiao Wu, Yi-Mei Fan, and Ya-Ping Wang. Epigenetic regulation of cdh1 exon 8 alternative splicing in gastric cancer. *BMC cancer*, 15(1):954, 2015.

Yang I Li, David A Knowles, Jack Humphrey, Alvaro N Barbeira, Scott P Dickinson, Hae Kyung Im, and Jonathan K Pritchard. Annotation-free quantification of rna splicing using leafcutter. *Nature genetics*, 50(1):151, 2018.

Lee P Lim and Christopher B Burge. A computational analysis of sequence features involved in recognition of short introns. *Proceedings of the National Academy of Sciences*, 98(20):11193–11198, 2001.

Sharlene R Lim and Klemens J Hertel. Commitment to splice site pairing coincides with a complex formation. *Molecular cell*, 15(3):477–483, 2004.

Seppo Linnainmaa. The representation of the cumulative rounding error of an algorithm as a taylor expansion of the local rounding errors. *Master's Thesis (in Finnish), Univ. Helsinki*, pages 6–7, 1970.

Hong-Xiang Liu, Michael Zhang, and Adrian R Krainer. Identification of functional exonic splicing enhancer motifs recognized by individual sr proteins. *Genes & development*, 12(13), 1998.

John Lonsdale, Jeffrey Thomas, Mike Salvatore, Rebecca Phillips, Edmund Lo, Saboor Shad, Richard Hasz, Gary Walters, Fernando Garcia, Nancy Young, et al. The genotype-tissue expression (gtex) project. *Nature genetics*, 45(6):580, 2013.

Reini F Luco, Qun Pan, Kaoru Tominaga, Benjamin J Blencowe, Olivia M Pereira-Smith, and Tom Misteli. Regulation of alternative splicing by histone modifications. *Science*, 327(5968):996–1000, 2010.

Fu-Jie Huang Marc'Aurelio Ranzato, Y-Lan Boureau, and Yann LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Proc. Computer Vision and Pattern Recognition Conference (CVPR'07). IEEE Press*, volume 127, 2007.

Ernest Martinez, Vikas B Palhan, Agneta Tjernberg, Elena S Lymar, Armin M Gamper, Tapas K Kundu, Brian T Chait, and Robert G Roeder. Human staga complex is a chromatin-acetylating transcription coactivator that interacts with pre-mrna splicing and dna damage-binding factors in vivo. *Molecular and cellular biology*, 21 (20):6782–6795, 2001.

Susan McVety, Lili Li, Philip H Gordon, George Chong, and William D Foulkes. Disruption of an exon splicing enhancer in exon 3 of mlh1 is the cause of hnpcc in a quebec family. *Journal of medical genetics*, 43(2):153–156, 2006.

D. Mishkin and J. Matas. All you need is a good init. *arXiv preprint arXiv:1511.06422*, November 2015.

Tom Misteli and David L Spector. Rna polymerase ii targets pre-mrna splicing factors to transcription sites in vivo. *Molecular cell*, 3(6):697–705, 1999.

Alberto Moldón and Charles Query. Crossing the exon. *Molecular cell*, 38(2):159–161, 2010.

Chantal F Morel, Mary Ann Thomas, Henian Cao, Caroline H O'neil, J Geoffrey Pickering, William D Foulkes, and Robert A Hegele. A lmna splicing mutation in two sisters with severe dunnigan-type familial partial lipodystrophy type 2. *The Journal of Clinical Endocrinology & Metabolism*, 91(7):2689–2695, 2006.

Antoine Muchir, Gisèle Bonne, Anneke J van der Kooi, Mia van Meegen, Frank Baas, Pieter A Bolhuis, Marianne de Visser, and Ketty Schwartz. Identification of mutations in the gene encoding lamins a/c in autosomal dominant limb girdle muscular dystrophy with atrioventricular conduction disturbances (lgmd1b). *Human molecular genetics*, 9(9):1453–1459, 2000.

Stephanie J Muh, Ruben Hovhannisyan, and Russ P Carstens. A non-sequence-specific double stranded rna structural element regulates splicing of two mutually exclusive exons of fibroblast growth factor receptor 2 (fgfr2). *Journal of Biological Chemistry*, 2002.

Manuel J Muñoz, M Soledad Pérez Santangelo, Maria P Paronetto, Manuel de la Mata, Federico Pelisch, Stéphanie Boireau, Kira Glover-Cutter, Claudia Ben-Dov, Matías Blaustein, Juan J Lozano, et al. Dna damage regulates alternative splicing through inhibition of rna polymerase ii elongation. *Cell*, 137(4):708–720, 2009.

Simon Myers, Colin Freeman, Adam Auton, Peter Donnelly, and Gil McVean. A common sequence motif associated with recombination hot spots and genome instability in humans. *Nature Genetics*, 40:1124–1129, 2008. URL `http://dx.doi.org/10.1038/ng.213`.

Sathiya Pandi Narayanan, Smriti Singh, and Sanjeev Shukla. A saga of cancer epigenetics: linking epigenetics to alternative splicing. *Biochemical Journal*, 474(6): 885–896, 2017.

Steven A Narod and William D Foulkes. Brca1 and brca2: 1994 and beyond. *Nature Reviews Cancer*, 4(9):665, 2004.

Yurii Nesterov. A method for unconstrained convex minimization problem with the rate of convergence o (1/kˆ 2). In *Doklady AN USSR*, volume 269, pages 543–547, 1983.

Guadalupe Nogués, Sebastián Kadener, Paula Cramer, David Bentley, and Alberto R Kornblihtt. Transcriptional activators differ in their abilities to control alternative splicing. *Journal of Biological Chemistry*, 277(45):43110–43114, 2002.

Nuala A O'Leary, Mathew W Wright, J Rodney Brister, Stacy Ciufo, Diana Haddad, Rich McVeigh, Bhanu Rajput, Barbara Robbertse, Brian Smith-White, Danso Ako-Adjei, et al. Reference sequence (refseq) database at ncbi: current status, taxonomic expansion, and functional annotation. *Nucleic acids research*, 44(D1):D733–D745, 2015.

Wilma K Olson, Andrey A Gorin, Xiang-Jun Lu, Lynette M Hock, and Victor B Zhurkin. Dna sequence-dependent deformability deduced from protein–dna crystal complexes. *Proceedings of the National Academy of Sciences*, 95(19):11163–11168, 1998.

JUN Otomo, Shigeo Kure, Tomoko Shiba, Akihiko Karibe, Tsuyoshi Shinozaki, Tetsuo Yagi, Hiroshi Naganuma, Fumiaki Tezuka, Masaetsu Miura, Meiichi Ito, et al.

Electrophysiological and histopathological characteristics of progressive atrioventricular block accompanied by familial dilated cardiomyopathy caused by a novel mutation of lamin a/c gene. *Journal of cardiovascular electrophysiology*, 16(2): 137–145, 2005.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

Tomaso Poggio, Hrushikesh Mhaskar, Lorenzo Rosasco, Brando Miranda, and Qianli Liao. Why and when can deep-but not shallow-networks avoid the curse of dimensionality: a review. *International Journal of Automation and Computing*, 14(5): 503–519, 2017.

Katherine S Pollard, Melissa J Hubisz, Kate R Rosenbloom, and Adam Siepel. Detection of nonneutral substitution rates on mammalian phylogenies. *Genome research*, 20(1):110–121, 2010.

Dan S Prestridge. Predicting pol ii promoter sequences using transcription factor binding sites. *Journal of molecular biology*, 249(5):923–932, 1995.

Daniel Quang and Xiaohui Xie. Danq: a hybrid convolutional and recurrent deep neural network for quantifying the function of dna sequences. *Nucleic Acids Research*, 44(11):e107, 2016. doi: 10.1093/nar/gkw226. URL `http://dx.doi.org/10.1093/nar/gkw226`.

Debashish Ray, Hilal Kazan, Kate B Cook, Matthew T Weirauch, Hamed S Najafabadi, Xiao Li, Serge Gueroussov, Mihai Albu, Hong Zheng, Ally Yang, et al. A compendium of rna-binding motifs for decoding gene regulation. *Nature*, 499 (7457):172, 2013.

Martin G Reese, Frank H Eeckman, David Kulp, and David Haussler. Improved splice site detection in genie. *Journal of computational biology*, 4(3):311–323, 1997.

Bing Ren, François Robert, John J Wyrick, Oscar Aparicio, Ezra G Jennings, Itamar Simon, Julia Zeitlinger, Jörg Schreiber, Nancy Hannett, Elenita Kanin, et al. Genome-wide location and function of dna binding proteins. *Science*, 290(5500): 2306–2309, 2000.

Jose Manuel Rodriguez, Paolo Maietta, Iakes Ezkurdia, Alessandro Pietrelli, Jan-Jaap Wesselink, Gonzalo Lopez, Alfonso Valencia, and Michael L Tress. Appris: annotation of principal and alternative splice isoforms. *Nucleic acids research*, 41 (D1):D110–D117, 2012.

Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

Hiroto Saigo, Jean-Philippe Vert, Nobuhisa Ueda, and Tatsuya Akutsu. Protein homology detection using string alignment kernels. *Bioinformatics*, 20(11):1682–1689, 2004.

Albin Sandelin, Wynand Alkema, Pär Engstrom, Wyeth W. Wasserman, and Boris Lenhard. Jaspar: an open - access database for eukaryotic transcription factor binding profiles. *Nucleic Acids Research*, 32:D91–D94, 2004. doi: 10.1093/nar/gkh012. URL `http://dx.doi.org/10.1093/nar/gkh012`.

Dominik Scherer, Andreas Müller, and Sven Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. In *International conference on artificial neural networks*, pages 92–101. Springer, 2010.

Schraga Schwartz, Eran Meshorer, and Gil Ast. Chromatin organization marks exon-intron structure. *Nature structural & molecular biology*, 16(9):990, 2009.

Katharina Schwarze, James Buchanan, Jilles M Fermont, Helene Dreau, Mark W Tilley, John M Taylor, Pavlos Antoniou, Samantha JL Knight, Carme Camps, Melissa M Pentony, et al. The complete costs of genome sequencing: a microcosting

study in cancer and rare diseases from a single center in the united kingdom. *Genetics in Medicine*, page 1, 2019.

Ulrike Schwarze, Barbra J Starman, and Peter H Byers. Redefinition of exon 7 in the col1a1 gene of type i collagen by an intron 8 splice-donor–site mutation in a form of osteogenesis imperfecta: influence of intron splice order on outcome of splice-site mutation. *The American Journal of Human Genetics*, 65(2):336–344, 1999.

Marina M Scotti and Maurice S Swanson. Rna mis-splicing in disease. *Nature Reviews Genetics*, 17(1):19, 2016.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 618–626, 2017.

Shalini Sharma, Lori A Kohlstaedt, Andrey Damianov, Donald C Rio, and Douglas L Black. Polypyrimidine tract binding protein controls the transition from exon definition to an intron defined spliceosome. *Nature structural & molecular biology*, 15 (2):183, 2008.

Zhen Shen, Wenzheng Bao, and De-Shuang Huang. Recurrent neural network for predicting transcription factor binding sites. *Scientific reports*, 8(1):15270, 2018.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Reverse-complement parameter sharing improves deep learning models for genomics. *bioRxiv*, 2017a. doi: 10.1101/103663. URL `https://www.biorxiv.org/content/early/2017/01/27/103663`.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3145–3153. JMLR. org, 2017b.

E Allen Sickmier, Katherine E Frato, Haihong Shen, Shanthi R Paranawithana, Michael R Green, and Clara L Kielkopf. Structural basis for polypyrimidine tract recognition by the essential pre-mrna splicing factor u2af65. *Molecular cell*, 23(1): 49–59, 2006.

Robert J Sims, Rimma Belotserkovskaya, and Danny Reinberg. Elongation by rna polymerase ii: the short and long of it. *Genes & development*, 18(20):2437–2468, 2004.

DAVID Solnick and SUSANNA I Lee. Amount of rna secondary structure required to induce an alternative splice. *Molecular and cellular biology*, 7(9):3194–3198, 1987.

Rotem Sorek and Gil Ast. Intronic sequences flanking alternatively spliced exons are conserved between human and mouse. *Genome Research*, 13(7):1631–1637, 2003.

Noah Spies, Cydney B Nielsen, Richard A Padgett, and Christopher B Burge. Biased chromatin signatures around polyadenylation sites and exons. *Molecular cell*, 36 (2):245–254, 2009.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014. URL `http://www.cs.toronto.edu/ rsalakhu/papers/srivastava14a.pdf`.

Rodger Staden. Computer methods to locate signals in nucleic acid sequences. 1984.

Charles W Sugnet, Karpagam Srinivasan, Tyson A Clark, Georgeann O'Brien, Melissa S Cline, Hui Wang, Alan Williams, David Kulp, John E Blume, David Haussler, et al. Unusual intron conservation near tissue-regulated exons found by splicing microarrays. *PLoS computational biology*, 2(1):e4, 2006.

Kazuhiko Takahara, Ulrike Schwarze, Yasutada Imamura, Guy G Hoffman, Helga Toriello, Lynne T Smith, Peter H Byers, and Daniel S Greenspan. Order of intron removal influences multiple splice outcomes, including a two-exon skip, in a

col5a1 acceptor-site mutation that results in abnormal pro-α1 (v) n-propeptides and ehlers-danlos syndrome type i. *The American Journal of Human Genetics*, 71 (3):451–465, 2002.

Melissa Talerico and Susan M Berget. Effect of 5'splice site mutations on splicing of the preceding intron. *Molecular and cellular biology*, 10(12):6299–6305, 1990.

Goranka Tanackovic, Adriana Ransijn, Philippe Thibault, Sherif Abou Elela, Roscoe Klinck, Eliot L Berson, Benoit Chabot, and Carlo Rivolta. Prpf mutations are associated with generalized defects in spliceosome formation and pre-mrna splicing in patients with retinitis pigmentosa. *Human molecular genetics*, 20(11):2116–2130, 2011.

Bosiljka Tasic, Christoph E Nabholz, Kristin K Baldwin, Youngwook Kim, Erroll H Rueckert, Scott A Ribich, Paula Cramer, Qiang Wu, Richard Axel, and Tom Maniatis. Promoter choice determines splice site selection in protocadherin α and γ pre-mrna splicing. *Molecular cell*, 10(1):21–33, 2002.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

Valérie Tosch, Nasim Vasli, Christine Kretz, Anne-Sophie Nicot, Claire Gasnier, Nicolas Dondaine, Denis Oriot, Magalie Barth, Hugues Puissant, Norma B Romero, et al. Novel molecular diagnostic approaches for x-linked centronuclear (myotubular) myopathy reveal intronic mutations. *Neuromuscular Disorders*, 20(6):375–381, 2010.

Janne J Turunen, Elina H Niemelä, Bhupendra Verma, and Mikko J Frilander. The significant other: splicing by the minor spliceosome. *Wiley Interdisciplinary Reviews: RNA*, 4(1):61–76, 2013.

Christopher R Vakoc, Sean A Mandat, Benjamin A Olenchock, and Gerd A Blobel. Histone h3 lysine 9 methylation and hp1γ are associated with transcription elongation through mammalian chromatin. *Molecular cell*, 19(3):381–391, 2005.

Julian P Venables, Roscoe Klinck, Anne Bramard, Lyna Inkel, Genevieve Dufresne-Martin, ChuShin Koh, Julien Gervais-Bird, Elvy Lapointe, Ulrike Froehlich, Mathieu Durand, et al. Identification of alternative splicing markers for breast cancer. *Cancer research*, 68(22):9525–9531, 2008.

Zefeng Wang and Christopher B Burge. Splicing regulation: from a parts list of regulatory elements to an integrated splicing code. *Rna*, 14(5):802–813, 2008.

M Bryan Warf and J Andrew Berglund. Role of rna structure in regulating pre-mrna splicing. *Trends in biochemical sciences*, 35(3):169–178, 2010.

M Bryan Warf, Julien V Diegel, Peter H von Hippel, and J Andrew Berglund. The protein factors mbnl1 and u2af65 bind alternative rna structures to regulate splicing. *Proceedings of the National Academy of Sciences*, pages pnas–0900342106, 2009.

Paul J Werbos. Applications of advances in nonlinear sensitivity analysis. In *System modeling and optimization*, pages 762–770. Springer, 1982.

A Wilson, D Leclerc, F Saberi, E Campeau, HY Hwang, B Shane, JA Phillips III, DS Rosenblatt, and RA Gravel. Functionally null mutations in patients with the cblg-variant form of methionine synthase deficiency. *The American Journal of Human Genetics*, 63(2):409–414, 1998.

Daniel E Worrall, Stephan J Garbin, Daniyar Turmukhambetov, and Gabriel J Brostow. Harmonic networks: Deep translation and rotation equivariance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5028–5037, 2017.

Jie Wu, Olga Anczukow, Adrian R Krainer, Michael Q Zhang, and Chaolin Zhang. Olego: fast and sensitive mapping of spliced mrna-seq reads using small seeds. *Nucleic acids research*, 41(10):5149–5163, 2013.

Hui Y Xiong, Babak Alipanahi, Leo J Lee, Hannes Bretschneider, Daniele Merico, Ryan KC Yuen, Yimin Hua, Serge Gueroussov, Hamed S Najafabadi, Timothy R Hughes, et al. The human splicing code reveals new insights into the genetic determinants of disease. *Science*, 347(6218):1254806, 2015.

Hui Yuan Xiong, Leo J Lee, Hannes Bretschneider, Jiexin Gao, Nebojsa Jojic, and Brendan J Frey. Probabilistic estimation of short sequence expression using rna-seq data and the positional bootstrap. *bioRxiv*, page 046474, 2016.

Saiqa Yasmeen, Katrine Lund, Anne De Paepe, Sylvia De Bie, Arvid Heiberg, Joao Silva, Márcia Martins, Tina Skjørringe, and Lisbeth B Møller. Occipital horn syndrome and classical menkes syndrome caused by deep intronic mutations, leading to the activation of atp7a pseudo-exon. *European Journal of Human Genetics*, 22 (4):517, 2014.

Gene Yeo and Christopher B Burge. Maximum entropy modeling of short sequence motifs with applications to rna splicing signals. *Journal of computational biology*, 11(2-3):377–394, 2004.

Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

Hua-Lin Zhou, Guangbin Luo, Jo Ann Wise, and Hua Lou. Regulation of alternative splicing by local histone modifications: potential roles for rna-guided mechanisms. *Nucleic acids research*, 42(2):701–713, 2013.

Jian Zhou and Olga G Troyanskaya. Predicting the effects of noncoding variants with deep learning-based sequence model. *Nature Methods*, 322(12):931 – 934, 2015a. doi: http://dx.doi.org/10.1038/nmeth.3547.

Jian Zhou and Olga G Troyanskaya. Predicting effects of noncoding variants with deep learning–based sequence model. *Nature methods*, 12(10):931, 2015b.