University of Southampton

Faculty of Medicine
School of Human Development and Health
Splicing and Disease

Improving Diagnosis of Genetic Disease through Computational Investigation of Splicing

by

Yaron Strauch
MSc, BSc
ORCiD: 0000-0003-0820-8319

A thesis for the degree of
Doctor of Philosophy

November 2022

University of Southampton

Abstract

Faculty of Medicine
School of Human Development and Health

Doctor of Philosophy

Improving Diagnosis of Genetic Disease through Computational Investigation of Splicing

by Yaron Strauch

Despite an estimate of 50% of pathogenic genomic mutations being related to splicing, this inherently complex mechanism is not yet fully understood. Identifying splice disruption is a complicated expert task requiring manual labour and expensive sequencing. With the emergence of Machine Learning for targeted medicine, modelling splicing computationally allows faster and less expensive analysis and ultimately, treatment. This project curates, analyses, optimises, and utilises Machine Learning datasets and algorithms for splicing related disease using supervised and unsupervised techniques. A clinical dataset of splice disrupting variants is curated, processed, and validated to assess algorithmic predictive performance in clinically relevant data. Predictions are improved by data engineering to include isoforms with lower expressions. Other avenues such as including protein binding sites, incorporating genomic conservation, and semantic encoding of DNA data are explored. CI-SpliceAI, a new algorithm to predict aberrant splicing, is developed and made available to the wider scientific community. Methods of how to explain shallow and deep learning are applied in order to visualise feature contribution of otherwise black-box algorithms to extract new insights about the underlying biological problem.

Contents

List of Figures

List of Tables

Declaration of Authorship

I declare that this thesis and the work presented in it is my own and has been generated by me as the result of my own original research.

I confirm that:

1. This work was done wholly or mainly while in candidature for a research degree at this University;
2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
3. Where I have consulted the published work of others, this is always clearly attributed;
4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
5. I have acknowledged all main sources of help;
6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
7. Parts of this work have been published as: Yaron Strauch, Jenny Lord, Mahesan Niranjan, and Diana Baralle. Ci-spliceai—improving machine learning predictions of disease causing splicing variants using curated alternative splice sites. *PLOS ONE*, 17(6):e0269159, 2022

Signed:.. Date:

Acknowledgements

I want to thank my family and friends who supported me through the hardship of a doctoral program during a global pandemic with an open ear and compassion.

I am also thankful to my supervisors, colleagues, and examiners for providing biological and medical foundations they might have taken for granted, and their patient support when being confronted with tangled and complicated questions.

This research was made possible through funding provided by the *National Institute for Health Research* and the *University of Southampton Institute for Life Sciences*, coordinated through Prof Diana Baralle, which I am very grateful for.

The technological side of the project would not have been possible without the *IRIDIS High Performance Computing Facility*, and associated support services at the University of Southampton.

With the oversight of my supervisors, editorial advice has been sought. No changes of intellectual content were made as a result of this advice.

Chapter 1

Introduction

1.1 Splicing

Ribonucleic Acid (RNA) splicing is a biological process where the *spliceosome*, a complex assembly of different proteins and functional RNAs (Wahl et al., 2009), binds and folds pre-*messenger RNA* (mRNA) to remove *introns*, areas that do not code for protein, and to maintain *exons* that do (Singh and Cooper, 2012). The boundary at the 3′ end of the intron is called the *acceptor site*, the site at the 5′ end is called the *donor*. Whether a region is spliced out or not depends on both *cis* factors, i.e. the nucleotide sequence, and *trans* factors, i.e. protein regulation (Singh and Cooper, 2012).

Figure 1.1 illustrates *constitutive splicing*, which describes the process of removing all intronic regions and retaining all exons. The most important nucleotide motifs occur around the intron-exon boundary and are visualised as frequency diagrams. The clearest motifs are the *consensus sites* AG at the acceptor and GT at the donor site. Introns

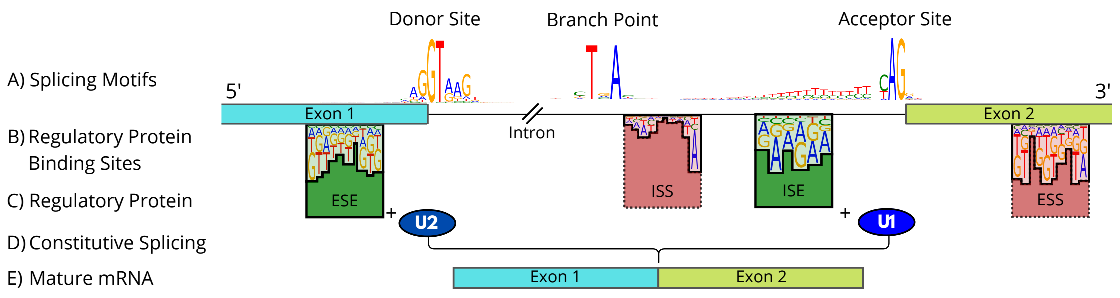


FIGURE 1.1: **Constitutive Splicing.** (a) Frequency diagrams at splice sites reveal nucleotide sequence motifs; GT/AG are the consensus sequences for donor and acceptor sites respectively. (b) Regulatory protein binding sites attract splicing enhancers or silencers (c). (d) In a constitutive splice, all exons are joined together by removing all intronic regions. (e) the resulting mature mRNA codes for the protein.
Branch point sequences in (a) taken from Corvelo et al. (2010); protein binding sequences in (b) taken from Cáceres and Hurst (2013); Venables (2007); Wang et al. (2013, 2004); splice motif sequences generated in this work

often end in a *polypyrimidine tract*, a tail of C/T bases. Another motif, called the *branch point*, is an adenine nucleotide generally located 23 to 27 bases upstream of the acceptor. While 99.24% of introns follow the GT-AG pattern and are spliced by the *U2* spliceosome (*major spliceosome*), the *U12* spliceosome (*minor spliceosome*) also binds to AT-AC in 0.05% of splice sites (Burset et al., 2000; Turunen et al., 2013). Other splice motifs include GC-AG (0.69%), and a remainder of 0.02% of other motifs (Burset et al., 2000).

1.1.1 Alternative Splicing

The constitutive splice that retains all exons and removes all introns is not the default case. In fact, it is believed that 92%-95% of multi-exon genes are expressed through *alternative splicing* (Pan et al., 2008; Wang et al., 2008), which means that the same pre-mRNA can be processed into different mature mRNAs, largely depending on trans factors, potentially resulting in vastly different protein expression. The resulting different mRNA and protein structures are also called *isoforms*.

Figure 1.2 illustrates different ways of alternative splicing. Exons can be skipped, alternated, or mutually excluded. Introns can be retained partially or completely. Combinations of these can result in vastly different isoformic proteins with similar, different, or even opposing functions. (Wang et al., 2008).

Splice sites that are contained within most isoformic expressions are often referred to as *strong* and include very clear consensus motifs, whereas *weak* splice sites refer to alternative junctions with weaker patterns that are only included in a subset of isoforms. Weak splice sites can be strengthened by promoting regulatory binding sites. (Dvinge, 2018).

Regulatory Proteins can bind to RNA motifs to excite or inhibit splice sites in their proximity in both exons and introns (Figure 1.1c); they are therefore termed *Exonic Splicing Enhancers* (ESEs), *Exonic Splicing Silencers* (ESSs), *Intronic Splicing Enhancers* (ISEs), and *Intronic Splicing Silencers* (ISSs). Enhancers, often *serine/arigine-rich* (SR), stabilise the binding of U1, U2 or SF1 while many protein in the *Heterogeneous nuclear ribonucleoprotein* (hnRNP) family bind to silencer sequences and inhibit splicing. This might however be a simplification as some proteins were shown to promote or inhibit sites depending on their relative position to the splice site. Mechanisms for regulation include attraction of the spliceosome, blocking splice sites, and altering distances between sites. (Dvinge, 2018; Kelemen et al., 2013)

Protein can further interact with each other. Such interaction can be antagonistic, for example when the two SR proteins RBM4 and PTB compete with another to bind to a 11-mer motif in *TPM1* to either excite or inhibit exon expression respectively (Lin

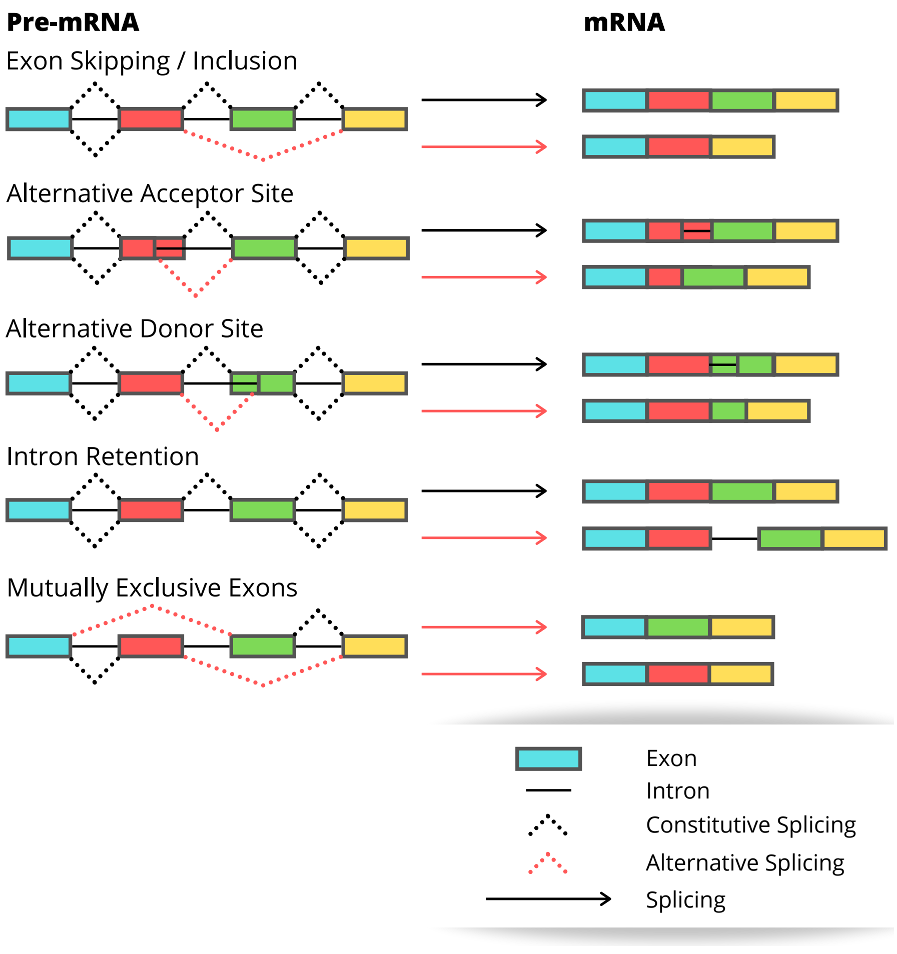


FIGURE 1.2: **Alternative Splicing.** The same pre-mRNA can produce different mature mRNA. Exons can be skipped, there can be alternative splice sites on both ends of the exon, introns can be included, and exons can mutually exclude each other. Adapted from Sen (2018).

and Tarn, 2005). Regulatory protein can also co-activate, for example PSF recruits Fox-3 which binds to a a motif, upregulating exon expression in neural cells (Kim et al., 2011).

Splicing can also regulated by *microRNA* (miRNA), short non-coding sequences of typically 19–22 nucleotides of RNA, which typically bind to the 3' end and often suppress translation or cause mRNA cleavage or degradation. miRNA expression originates from introns between coding and non-coding genes, exons, or intergenic regions and can in turn be regulated per tissue; some mRNA regulatory elements were shown to also regulate in miRNA expression, forming complex and intrinsic regulatory relationships. (Ratnadiwakara et al., 2018)

Although often modelled as that, splicing is not a distinct process that happens independently after transcription. Instead both processes appear to be co-dependent and part of a shared macronuclear complex. During transcription, part of the RNA polymerase II recruits splicing factors to the complex, directly changing alternative splicing factors through spatial coupling in the *Deoxyribonucleic Acid* (DNA) transcription process. Pre-mRNA splicing and folding appears to be further affected by the elongation rate, the speed of transcription and pauses thereof. It was suggested that sequences transcribed earlier have longer time to recruit splice modifiers than regions closer to the transcription end. These pauses in transcription might even be part of a two-way coupled system as studies have observed slower transcription at splice sites. Whether this pause is required for co-transcriptional splicing has not been shown yet and it is unclear if this represents cause or effect of splicing. (Saldi et al., 2016)

Alternative splicing is often observed to be specific to tissue (Wang et al., 2008) and observed isoforms can change depending on the developmental stage (Kalsotra and Cooper, 2011). For example, the protein family PTB consists of three protein coding genes: *PTBP1* which is expressed in the majority of cells except the neurons, *PTBP2* which is limited to the nervous system, and *PTBP3* that is mostly associated with the immune system. Expression is found to be regulated through splicing (hnRNP 1). During neural development, *PTBP1* is inhibited and *PTBP2* is excited, and after neuron maturity, both are regulated down. This regulatory dependency plays an essential role during cell differentiation of neurons and disruptions of expression can cause failure to mature (Hu et al., 2018; Keppetipola et al., 2012).

This is an example of a common PTB mechanism responsible for both tissue and developmental stage specific splicing (Black, 2003). PTB is commonly expressed throughout the body and downregulates highly specific exons in many genes. In some tissue, such as neurons, PTB concentration is generally lower which allow these exons to be expressed. Other tissue might have higher concentration of competing protein. For example, ETR-3 is a direct antagonist to PTB that upregulates *cTNT* exon 5 expression in embryonic muscle cells. When ETR-3 binds to intronic binding sequences in these cells, it binds in lieu of PTB and exon 5 switches from skipping to inclusion (Charlet-B et al., 2002). PTB expression itself might also be changed depending on tissue type and developmental stage, for example by changing its isoform to an inactive transcript or to homologues that bind differently (Black, 2003).

1.1.2 Diagnosing Splicing Disease

Splicing occurs in many eukaryotic organisms and is highly conserved from evolutionary drift (Blakes et al., 2022). Disrupted splicing can have devastating impacts on the health of patients. It is estimated that a majority of pathogenic *Single Nucleotide Variants* (SNVs) disrupt splicing (Truty et al., 2021); discussion of *Multi Nucleotide Variants* (MNVs) is rarer and no such statistics exist to my knowledge. How reliably these estimates that sometimes reach over 60% quantify the actual contribution of splicing towards genetic disease however is hard to asses and disputed across literature (Gonorazky et al., 2019; López-Bigas et al., 2005; Truty et al., 2021); one major cause for this is that many bioinformatic pipelines filter down to protein-coding regions and remove synonymous variants that don't affect amino acid coding directly (Richards et al., 2015), but still might cause aberrant splicing.

The effect of splice disruption can be as versatile and complex as healthy alternative splicing. Mutations can disrupt, alter, or create splice sites and regulatory binding sites, which in turn can cause abnormal partial or whole exon skipping or inclusion, intron retention or skipping, or otherwise disrupt the splicing processes. Disruption of regulatory sequences, including in non-coding and intergenic regions, may hinder binding and cascade to abnormal expression. All of these effects can affect the 3D protein structure and may cause phenotypic dysfunction. For example, the severity of *Spinal muscular atrophy* (SMA), a genetic disease causing severe muscle degeneration through a mutation in the gene *SMN1*, correlates with how well the gene *SMN2* is regulated and alternatively spliced. Depending on the expression of *SMN2*, the patients' life expectancy can range from mere weeks to months or well into adulthood (Lunn and Wang, 2008). Even intronic and silent variants can change mRNA expression and cause genetic disease like dementia and parkinsonism (D'Souza et al., 1999).

Due to the inherent complexity of the splicing process, it is not understood completely yet. When finding a variant of uncertain significance through genomic DNA sequencing, the effect on the splicing mechanism is often not obvious and diagnosis requires functional analysis *in vitro*. Experts need to either transfect the wild type and variant DNA sequence into a minigene and compare both RNA expressions (*minigene assay*) (Gaildrat et al., 2010), or sequence a patient's RNA and compare it to a healthy reference (Wai et al., 2020). Both approaches, while accurate, are expensive and time consuming expert tasks that seem unfit for broad, personalised genetic screening.

In clinical practice, evidence of pathogenicity of genetic variants are commonly classified using the *American College of Medical Genetics* (ACMG) guidelines (Richards et al., 2015). Variants that are shown to cause a frame-shift or nonsense mutation, exon deletion, or disrupt a canonical splice site sequence or start codon in a gene where loss of function is known to cause disease are rated to *very strongly* indicate pathogenicity (PVS1). Variants shown to have a damaging effect on the gene or protein *in vitro* or *in vivo*, such as in a minigene essay, are classified as PS3, corresponding to *strong* evidence of pathogenicity.

If we understood and modelled the splicing process *in silico*, we could improve genetic diagnoses significantly. While under ACMG guidelines computational results are only accepted as *supporting* evidence (PP3), they are often used as a pre-filtering tool to select for variants of interest that can then be further analysed through more reliable and expensive means.

There is a vast number of computational tools that predict splice sites (Jian et al., 2014; Rowlands et al., 2019), and a discussion of all of them is not within the scope of this work. This review rather contains the most predominantly used and best performing tools.

Early research focused on *Position-Weight Matrix* (PWM) (Stormo et al., 1982) analysis to extract common motifs of splice sites. PWM analysis extracts individual nucleotide frequencies through information theory (detailed methodology in section 3.2.3), classically plotted into so-called *logos*: Visualisations where nucleotides are scaled according to importance and stacked upon each other. As an example, Figure 1.1 used logo visualisations to illustrate splicing motifs. The derived PWM can then be combined with expert knowledge and functional analysis into computational tools. *Human Splicing Finder* (Desmet et al., 2009) used a data-driven approach and derived site strengths from observed frequencies for *n-mer* sequences. *ESEFinder* (Cartegni et al., 2003) extracted PWM ESE motifs from functional analysis. *RESCUE-ESE* (Fairbrother et al., 2002) partitioned 6-mers at exon boundaries based on frequency analysis and clustering to derive 10 ESE 6-mer motifs that were experimentally validated to "rescue" weak splice sites.

A basic flaw in all tools based on PWM analysis is that they assume the frequency of observing a nucleotide at a certain position to be independent of the frequency of all other nucleotides. This assumption must be wrong due to the nature of protein binding of the spliceosome. *MaxEntScan* (MES) (Yeo and Burge, 2004) mitigates this issue by using maximum entropy models that calculate the likelihood of one nucleotide occurring in the context of its neighbourhood. The authors further used hierarchical clustering to distinguish acceptor from donor sites, which then can be visualised as a PWM themselves, providing more insight into context dependencies of nucleotide sequences.

Many of these tools often output conflicting results of poor quality (Jian et al., 2014; Rowlands et al., 2019). With the recent emergence of *Machine Learning* (ML), statistical tools to recognise patterns in data, and *deep learning*, very complex algorithms with up to millions of parameters, we hope to model splicing more reliably and to extract new knowledge of the underlying mechanisms.

Cheng et al. (2019) published *Modular Modeling of Splicing* (MMSplice) which uses multiple smaller neural networks ("modules") to predict splice sites given 18 or 53 nucleotides of context. The algorithm *Super-quick Information Content and Random Forest Learning for Splice Variants* (SQUIRLS) published in Danis et al. (2021) uses carefully engineered features from around the variant to classify splice disruptions using two decision trees and a logistic regression, and SpliceAI (Jaganathan et al., 2019) uses five deep *Convolutional Neural Networks* (CNNs) to predict splice sites based on 10,000 nucleotides of context. SpliceAI models the splicing process directly on a per-nucleotide basis, which may lead to insights into the splicing mechanism itself. Its authors attribute their model performing more accurately than its competitors predominantly to the bigger context size, allowing SpliceAI to classify even very deep variants (variants located hundreds of bases from a splice site) that cannot be detected by the other algorithms listed.

Modelling splicing per nucleotide allows granular predictions; if we could understand how the model reaches these conclusions, we could potentially gain insights into new splicing patterns and maybe even into the underlying biology itself. This might however prove infeasible due to the complexity of both deep learning and the raw input encoding of DNA sequences, which is a format that is hard for humans to understand. This complexity might explain why nobody was able to explain how SpliceAI reaches its conclusions yet. SQUIRLS on the other hand uses engineered features that have already incorporated many biological insights into the splicing process, making interpretation of feature contributions easier. There may be a fruitful middle ground between completely manually curated features and raw input sequences, such as encoding similar sequences close to each other in a semantic space such as *Word2Vec* (Mikolov et al., 2013) does, or through an overall incorporation of protein binding sites or conservation.

1.2 Overall Aims and Objectives

This work aims to improve diagnosis of splicing disease through computational analysis, and to extract insights into the underlying biological process. Datasets and ML tools are to be optimised towards their application to clinical diagnosis in order to assist, add to, or even replace parts of current genomic pipelines. Models are then analysed with the aim to extract insights into their workings, limitations, and to ultimately gain new biological insights into splicing related disease.

This is to be achieved through the following objectives:

1. Develop, test, and analyse algorithms to recognise splice sites (chapter 3)
   (a) Create ML datasets
   (b) Analyse datasets using statistical and unsupervised methods
   (c) Build simple baseline classification algorithms and compare them to SpliceAI
   (d) Incorporate weaker splice sites from the *Gene Encyclopedia of DNA Elements* (GENCODE)
   (e) Compare performance between models when training on primary splice sites versus inclusion of weaker GENCODE sites
   (f) Explain splice site classification results
2. Apply supervised algorithms to annotate aberrant splicing in variant data (chapter 4) :

   (a) Aggregate variant data from the literature
   (b) Investigate and resolve conflicts where sources disagree
   (c) Compare baseline classifiers to deep learning
   (d) Quantify how inclusion of weaker isoforms changes predictive accuracy

3. Improve splice site recognition through data engineering (chapter 5) :
   (a) Annotate protein regulation binding motifs
   (b) Incorporate conservation scores
   (c) Encode DNA in vector space through *DNA to vector* (DNA2Vec)

4. Make the newly developed *Collapsed Isoform SpliceAI* (CI-SpliceAI) algorithm accessible (chapter 6) :
   (a) Train CI-SpliceAI using a re-implementation
   (b) Predict if splice disruptions occur and what the exact effect on the pre-mRNA is
   (c) Compare it to other splice prediction tools
   (d) Package it into a command line tool
   (e) Publish it as a freely accessible website

1.3 Background

Machine Learning (ML) is a collection of techniques that apply statistical methods to model, recognise, and learn patterns in data. In contrast to rule-based approaches where engineers model a problem domain through expert knowledge, ML is a data-driven approach that *fits* parameters of a generalised and well-defined *model*. The fitted model can be used to make inferences (*predictions*) on both seen and unseen data. (Géron, 2019)

1.3.1 Supervised Machine Learning

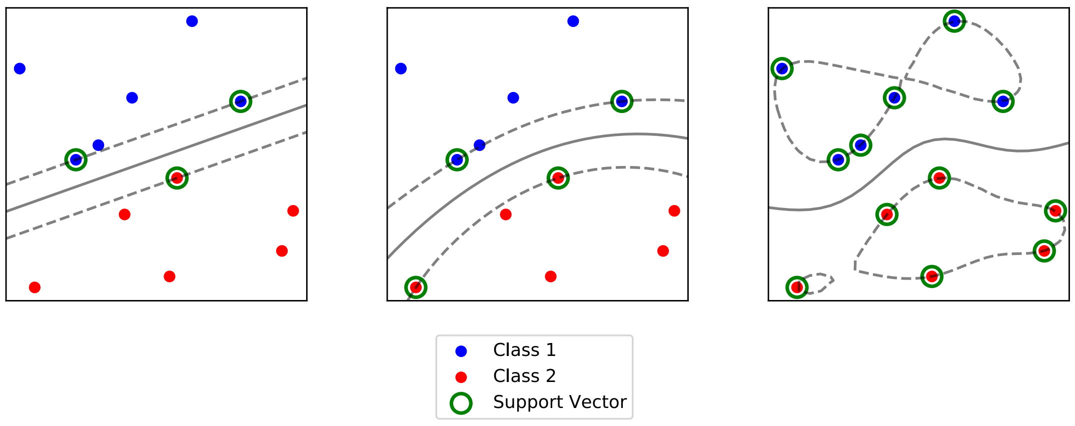
Supervised Machine Learning requires data to be annotated so that for every input, a desired output is defined. A model can range from simple linear regressions to complex neural networks that try to imitate basic brain functionality. The two sub types of supervised algorithms are *regression* (models that return continuous values, such as how often a site is spliced), and *classification* (models that output which class from a fixed set of classes is most likely, such as acceptor / donor / neither). (Géron, 2019)

Overfitting is the process where a supervised model learns specific solutions to the given training data instead of finding a generalised approach that can predict well on unseen data. To test if a model generalises well, the available data is split into *training* and *testing* partitions which allows evaluation of the trained model on unseen data. If training performance is significantly higher than testing performance, the model is overfit. (Géron, 2019)

Depending on the task, different scores for measuring model performance are common. For classification tasks, *accuracy* is the percentage of correctly classified instances while the *Area Under the Precision-Recall Curve* (AUC-PR) represents the integral over different thresholds balancing the ratio of true positive rate to false positive rate; for both measurements bigger values are better. (Géron, 2019)

The *average precision score* is very similar to the AUC-PR with the main difference being how the integral is calculated; while AUC-PR interpolates the curve to compute the integral with a trapezoidal rule and may return too optimistic results, average precision does not interpolate. (Davis and Goadrich, 2006; Flach and Kull, 2015)

1.3.1.1 Support Vector Classifiers



(A) (B) (C)

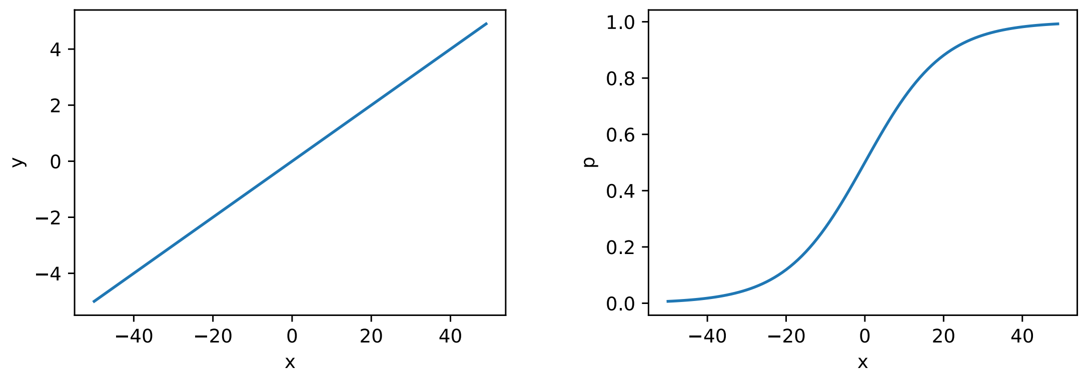
FIGURE 1.3: **Three SVCs on the same artificial data set.** (a) The linear SVC separates the data reasonably well. (b) An RBF SVC with a small gamma ($\gamma = 0.1$) allows the non-linear kernel to curve the street, capturing the data distribution even better. (c) The RBF overfits with extremely big gamma values ($\gamma = 1.25$) and use every data point as a support vector, resulting in a highly unstable boundary.

Support Vector Classifiers (SVCs), (also often called *Support Vector Machines*, SVMs), try to fit a hyper plane between two data distributions so that they are separated by as much space as possible. This space in between is often called *street* as illustrations often utilise three lines resembling a road (Figure 1.3a). The hyper plane and its margins are described by their supporting vectors, hence its name. After finding the separation between the two distributions, classifications of data points are calculated by determining which side of the street they fall on. (Cristianini et al., 2000)

Due to the linearity of a hyper plane, a linear SVC works best on data that has a linear boundary, which is often not the case. Kernel SVCs allow non-linear boundaries by transforming the data using a non-linear kernel function, such as a *Radial-Basis Function* (RBF), allowing a linear hyper plane to separate non-linear data (Figure 1.3b). (Cristianini et al., 2000)

In the kernel case, the hyper parameter γ regularises the importance of data points depending on their distance to the decision boundary; higher values weigh close points higher and might result in an overfitted boundary, lower values flatten the boundary and might underfit. How to optimise regularisation parameters is described in section 1.3.1.10 on page 19.

1.3.1.2 Logistic Regression



(A) (B)

FIGURE 1.4: **Comparison between a linear regression and a logistic transformation.** Both are configured with $a = 0.1, b = 0$. (a) The output of a linear function leaves the interval $[0..1]$ for small and large values, which does not result in sensible probabilities. (b) The logistic transformation limits the output to the interval $[0..1]$ and can be interpreted as a probability

Despite its name containing 'regression', *Logistic Regression* (Logit) is used for classification tasks. The core idea is to use a linear model to predict the probability of a sample belonging to a class (i.e. the probability of a site being spliced). As illustrated in Figure 1.4a, a linear function $y = ax + b$ emits values outside of the interval $[0, 1]$ which are invalid probabilities. This is compensated by using a logistic transformation (also known as *sigmoid* function) $p = \frac{1}{1+e^{-y}}$ that saturates between zero and one and can therefore be interpreted as a probability, see Figure 1.4b. (Menard, 2002)

Training is achieved by maximizing the log-likelihood of the classifier. This is equivalent to maximum entropy (*MaxEnt*) modelling (Cristianini et al., 2000).

1.3.1.3 Multi-Class Classification

SVCs and Logit are binary classifiers, i.e. they can only distinguish two outcomes at a time. If more than two distributions are to be classified (*multi-class* classification), i.e. acceptor sites, donor sites, and neither, multiple binary classifiers are fitted.

For $N > 2$ classes, *one-versus-rest* classification (also called *one-versus-all*) trains N classifiers, each distinguishing one class versus the remainder. *One-versus-one* (also called *all-versus-all*) fits one classifier for every possible pair of classes, resulting in $\frac{K(K-1)}{2}$ classifiers (Aly, 2005). For $N = 3$ classes, such as acceptor / donor / neither, both approaches result in 3 classifiers.

1.3.1.4 Decision Trees, Random Forest

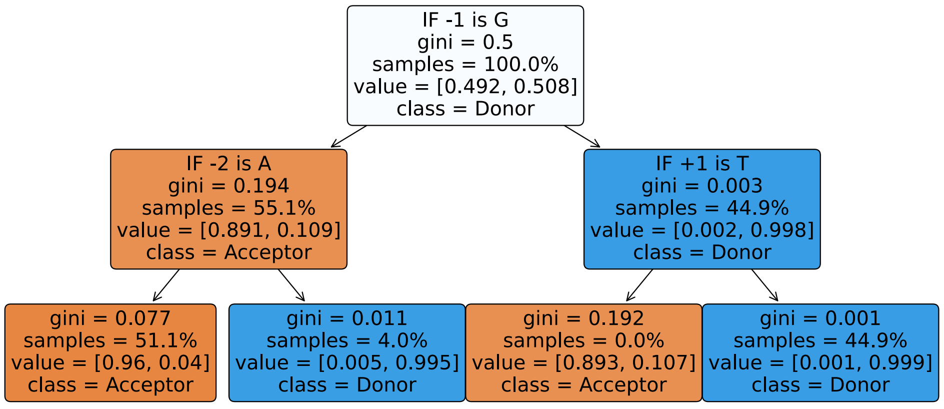


FIGURE 1.5: **A simple decision tree of depth 2 to distinguish acceptors from donors.** To determine the classification output for a location on the genome, follow each decision, starting at the root node. If the datapoint fulfils the decision criterion, follow the left arrow, else follow the right arrow. This specific model first checks the nucleotide upstream is a G; if yes and the nucleotide one further up is an A, the canonical AG motif is detected and the point is classified as an acceptor. The ”value” property of the corresponding leaf node (bottom left) also shows that 4% of training data that this path represents is misclassified.

A *Decision Tree* (Rokach, 2016), as illustrated in Figure 1.5, represents a number of successive data splits. Every parent node represents one decision to make, every leaf node represents a classification result. Each decision splits data into two disjoint partitions, one variable at a time. All nodes connected build a binary tree with one root node. Incoming data is classified by starting at the root node, following all applicable connections, until a leaf node determines the classification result.

A tree is trained using information theory: Given data, the best decision of a node is found by minimising impurity (i.e. gini score) of its children or by maximising the information gain of each split (Rokach, 2016). Formulae are found in Tangirala (2020).

One of the main issues with decision trees is overfitting. A decision tree where each leaf node represents a single class has most likely overfitted to the training data set and will evaluate poorly on unseen data. A number of regularisation parameters exist to mitigate overfitting, including (but not limited to) capping the number of layers, number of samples required to issue a split, or the number of leaf nodes. The effect of preventing overfitting through regularisation is semantically similar to the γ parameter of the SVC; how to optimise regularisation parameters is explained later in section 1.3.1.10 (page 19).

Decision trees are inherently unstable, meaning small changes in the data set will impact the topology drastically. Furthermore, most decision tree implementations are breaking ties randomly, introducing stochastic effects that further disturb stability.

Random Forests (RFs) have their roots in ensemble learning methods and mitigate instability by the law of big numbers. Instead of having one unstable decision tree, a forest of trees is grown, each based on a different subset of data. By combining a large number of very different classifiers and taking the majority vote, random side effects are reduced and the model becomes stable as a whole. The prediction performance grows with the number of trees but saturates eventually. (Rokach, 2016)

1.3.1.5 Simple Artificial Neural Networks

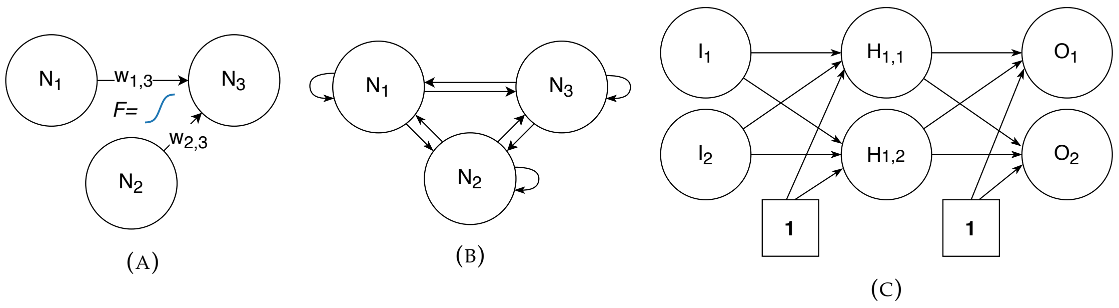


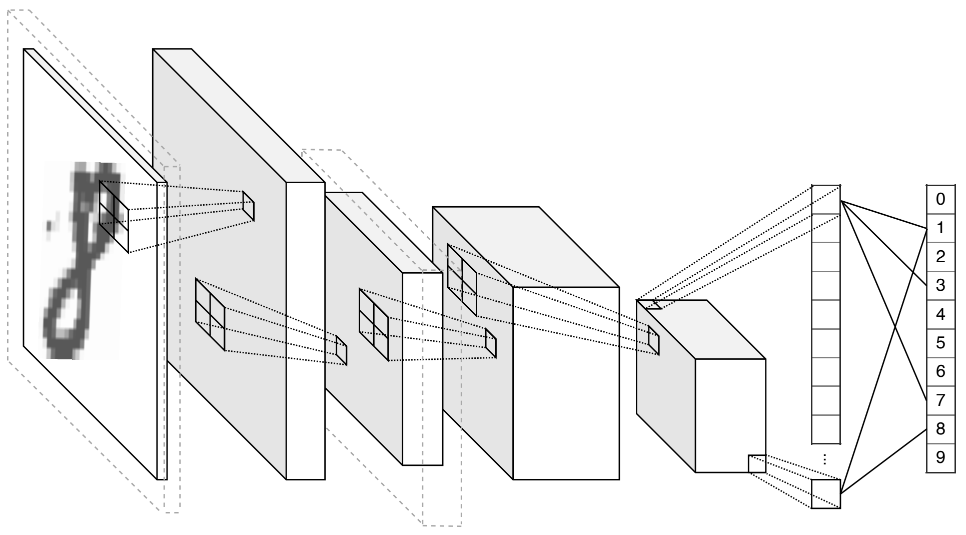
FIGURE 1.6: **Simple Artificial Neural Networks.** ANNs consist of neurons (circles), weights (arrows), and biases (squares). An activation function F is applied to all weighted inputs (not just where indicated). (a) Three neurons with two connections. The output of N_3 is calculated by applying the activation function to its weighted inputs: $N_3 = F(w_{1,3} * N_1 + w_{2,3} * N_2)$. (b) A fully connected RNN with three neurons. Every neuron is connected to every neuron, including itself. The number of weights therefore grows exponentially with the number of neurons. (c) A fully connected three-layer feed-forward network, also called Multi-Layer Perceptron Classifier. There are no loops feeding back, values are only passed forward. Biases emit a constant value of 1, allowing their corresponding weights to offset the input by a constant number.

Artificial Neural Networks (ANNs) are a drastic abstraction of biological neural networks used for both classification and regression. Neurons are connected through learnable weights (Figure 1.6a); each input neuron represents one feature, each output neuron one outcome variable. Each neuron sums up its weighted input signals and applies an activation function to determine its output signal. Most commonly used activation functions include the logistic function (*sigmoid*, see Figure 1.4b), a *Rectified Linear Unit* (ReLU), and variations thereof. If outcomes are mutually exclusive (i.e. classification task), a *softmax* function is applied on the output neurons. The softmax is related to the logistic transformation (section 1.3.1.2) in that it transforms raw outputs to a probability distribution of classes. The weights of an ANN are initialised randomly and found by propagating the training error between ground truth and current output back to earlier neurons, adjusting the weights iteratively by a certain *learning rate*. (Basheer and Hajmeer, 2000; Géron, 2019)

Recurrent Neural Networks (RNNs), as depicted in Figure 1.6b, describe the most flexible class of ANNs that allows loops, including neurons that are connected to themselves. Loops are *unrolled* over time, allowing to store and retrieve time-dependent information. However, since the number of parameters of a fully connected RNN explodes exponentially with the number of neurons, deep RNNs with thousands or even millions of neurons are infeasible with current technology.

Multi-Layer Perceptrons (MLPs) (Figure 1.6c), often also called multi-layer feed-forward networks, mitigate the number of connections by arranging neurons into a strict hierarchical, loop-free layer structure. The output of one layer is fully connected to the next layer. The layers in between input and output layers and their neurons are called *hidden layers* and *hidden neurons* respectively. MLPs are a common trade-off between complexity and accuracy. *Multi-Layer Perceptron Classifiers* (MLPCs) are MLPs with a softmax layer for classification.

1.3.1.6 Convolutional Neural Networks



Layer Type	Input	Convolution	Pooling	Convolution	Pooling	(Flatten)	MLP	Softmax, Output
Part	Feature Detection					Classification		

FIGURE 1.7: **A Convolutional Neural Network recognising a hand-written digit.** Early convolutional layers recognize small features; their input can be padded to prevent the data from shrinking. Pooling layer resize the data, allowing the following convolutional layers to create more complex features. Features are flattened and classified by a softmax-MLP. (Strauch, 2019)

CNNs were originally developed for image recognition, but today they are part of many other disciplines and represent a state-of-the-art ML algorithm, especially for deep learning. Their main distinction is that they re-use weights by moving *kernels* over their input, allowing them to analyse a sub section of the data at a time and utilise relationships of neighbouring features. (Géron, 2019; LeCun et al., 1989)

Early layers in the stack detect small basic motifs that are then re-combined by later layers to find macro patterns. Features are recognised in *Convolutional Layers* by multiplying their input data with learnable kernels. These kernels are moved over the input matrix like a torch, emitting one output value per position into an output matrix. *Pooling Layers* shrink their input data by applying a max or mean function, allowing the following convolutional layers to combine previous features and recognize bigger patterns. The same effect can be achieved by *dilation* which means that gaps between feature maps are introduced to allow a kernel to cover bigger areas. *Batch normalization* layers centre batches of data by subtracting the mean and dividing by standard deviation to provide better convergence.

1.3.1.7 SpliceAI

Jaganathan et al. (2019) presented ensemble models of five deep CNNs each to predict splice sites from raw DNA sequence. While their publication discusses different model architecture variations trained on different context sizes and layer depths, their best model presented, SpliceAI10k (Figure 1.8), is commonly referred to as *SpliceAI*. This architecture utilises five 35-layer deep CNNs with an input context length of 10,000 bases flanking the 5,000 nucleotides to predict (5,000 flanking nucleotides on either side).

The authors illustrate that bigger context sizes and deeper networks achieve higher accuracies. They also work with two datasets. First, they only train and test on primary GENCODE transcripts (i.e. those isoforms that are predominantly expressed throughout the body and developmental stages), and their SpliceAI10k model achieves an average precision[1] of 98% on test data. They then enrich their data with weaker and novel splice sites from the *Genotype Tissue-Expression* (GTEx) database, which they observed to perform better on clinical variant data. Their final published SpliceAI model is trained on 19 chromosomes (excluding the test chromosomes 1,3,5,7,9) on the GENCODE+GTEx data. The authors did not publish how well this model performs on the test split of the combined dataset, but a third party measured an average precision of around 84% which was confirmed by the authors (Riepe and Jaganathan, 2022).

Using a CNN has great advantages in the context of splicing. As explained in the previous section, CNN kernels are applied to neighbouring nucleotides and can model their relationship. They detect small detailed features and combine them into new macro

[1]The authors report this score as AUC-PR

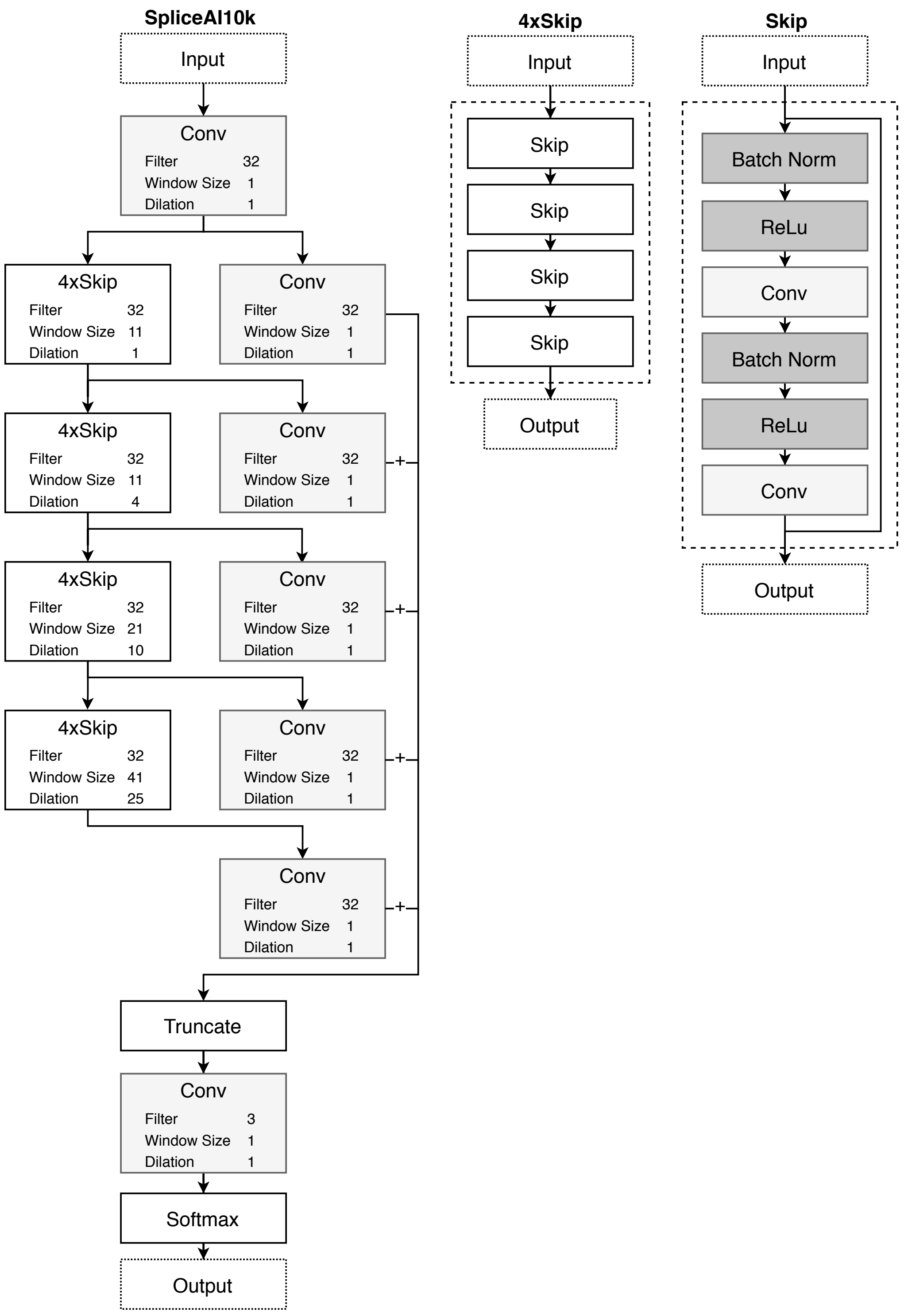


FIGURE 1.8: **The SpliceAI10k architecture.** The network is very deep. The main layer stack splits into two strands. The left strand consists of four skip blocks of each four skip layers that in turn have two convolutional layers, each with a batch normalization and ReLU layer. The right strand functions as an additional skip connection consisting of four convolutional layers. The first and last convolutional layers in the stack are there to provide data shapes. All other convolutional layers have 32 filters, and their window size and dilation increase in later layers to compile smaller features into bigger ones. In the end, the output is truncated to the 5,000 nucleotides in the middle and a softmax determines classification.

features on later layers. This is important because as shown by MES, neighbouring nucleotides have relations to one another and compile into macro motifs (section 1.1.2). Another advantage of a CNN is that it can output a matrix, allowing SpliceAI to predict not only a single splice site at a time, but thousands of neighbouring nucleotides at once.

The big disadvantage of deep neural networks however is that the reasoning behind the algorithm can be hard to understand. Through investigation of the kernels found, one can extract some information, but the deeper a CNN, the less interpretable they are. Analysis is further impacted by SpliceAI using dilation, which means that kernels are spread out and have gaps. Their algorithm therefore is great as a tool, but extracting new insights into the splicing process is non-trivial.

Furthermore, the choice of training the final SpliceAI model on the combined dataset of primary GENCODE transcripts enriched with ”novel splice junctions commonly observed in the GTEx cohort” (Jaganathan et al., 2019) might introduce problems. Firstly, the inclusion of novel junctions from GTEx data, filtered to those observed in at least five patients, was not validated by humans. This could mean batch effects or noise being sampled from the data. Secondly, if there are related genetic conditions in at least five patients, those conditions will be part of an otherwise assumed healthy dataset. Thirdly, the combination with GTEx data means that the input and output of the machine learning data is de-correlated. Their training code generates input sequences sampled from the reference genome and not from GTEx donors. Output annotations on the other hand contain splice sites subject to all of the donors’ genetic variation. This may teach the ML algorithm to detect splice sites in DNA where there are none, potentially creating confusing and even conflicting annotations. To prevent this, one should either use GTEx inputs for GTEx annotations (and the reference genome for GENCODE annotations), or stick to GENCODE by filtering it to also contain weaker splice sites and not combining it with GTEx. Lastly, the choice of excluding five chromosomes from the train data of the final model might bias the algorithm and hinder its clinical application.

1.3.1.8 Cross-Validation Partitioning

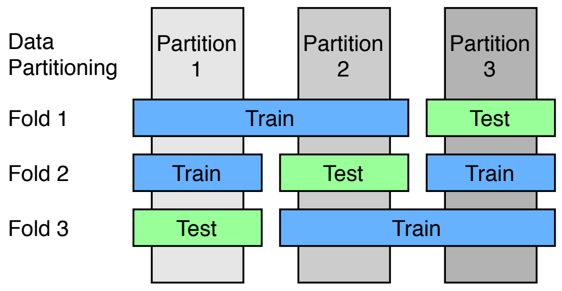


FIGURE 1.9: **How K-Fold operates on 3 folds.** K-Fold splits the data into K (3 in this case) partitions and evaluates the model in K folds, selecting one partition as test data and the remainder for training.

In order to measure how well algorithms adapt to unseen data, the general procedure is to split the data into train and test partitions. Despite various techniques and best practises in how to find a sensible split, the choice of split will impact model performance on any finite data set. To prove that the performance of a model does not rely significantly on the split, *Cross Validation* (CV) operates on multiple splits, each consisting of a train and test run. (Wong, 2015)

K-Fold is one of the most popular CV techniques. As illustrated in Figure 1.9, it splits the data into K partitions of equal size. The model is trained and tested K times; for every run, one partition is selected as a test set and the remainder is used for training. (Wong, 2015)

The model performance between runs will fluctuate; the less fluctuation, the higher the confidence in the stability of the model. A standard practice is to report the mean $\overline{m}$ and the standard deviation σ of the performance measure on the test partition across all runs in the format $\overline{m} \pm \sigma$.

There is an obvious trade-off with computational cost. SpliceAI trains on ten *Graphics Processing Units* (GPUs) for each 12 hours, i.e. 120 hours of total GPU time per fold. CV would therefore be very expensive. The authors instead decided to evaluate their model on a single train/test split.

1.3.1.9 Regularisation

Regularisation can help in simplifying and stabilising models. Models with fewer weights, represented by their β coefficients, can be easier to interpret and reduce complexity, fewer extreme weights can help the model stabilise.

Lasso regularisation adds the l_1 norm of all parameters, defined as $\sum |\beta|$, to the optimisation problem (subject to minimisation), which punishes non-zero weights and drives the model coefficients towards sparsity. Similarly, *ridge* regularisation utilises the l_2

norm defined as $\sum \beta^2$, punishing big weights and driving the model towards a more even weight distribution (Hastie et al., 2009). The combination of both is called *Elastic Net*. In scikit-learn, the lasso parameter in a linear regression is called α, and for algorithms that support both l_1 and l_2, the weight is called C independent of which exact norm was used. Either way, this parameter determines how much the regularisation affects training, bigger values increase the weight of the regularisation factor.

1.3.1.10 Hyper Parameter Optimisation

As opposed to the model parameters found during training, *hyper parameters* are set-up *a priori*. For the algorithms discussed so far, hyper parameters are i.e. the depth of a decision tree, the learning rate for ANNs, or the regularisation coefficient C. Apart from setting parameters by hand, based on experience and trial and error, they can be found systematically.

Grid Search is the systematic exploration of all hyper parameter combinations within a search space. That means that for every hyper parameter, a search domain is defined by hand. Every possible combination of all parameters in the search space form a grid; each combination is trained and tested on, and the best test result defines which parameter configuration is assumed to be optimal. Since grid searches scale exponentially with the search domains defined, an engineer still needs to define sensible search spaces, extending the search space manually if necessary. A modern approach is *Bayesian Optimisation* (Kandasamy et al., 2020) that models the loss function subject to hyper parameters themselves using bayesian inference and gaussian processes.

Grid Search CV is the process that for every parameter configuration on the grid, a CV determines the performance of the parameter configuration. This strengthens the trust in the hyper parameters found.

1.3.1.11 Explaining Supervised ML Decisions

Understanding the reasoning why and how a supervised ML algorithm comes to a conclusion is important for understanding classification results, errors, and could even reveal insights about the underlying problem. Especially with tasks where humans are not able to recognise the patterns reliably and have to use expensive analysis, such as in the splicing domain, understanding the reasoning could help us not only refine the data pipelines and models, but we might even generate new insights into patterns and the underlying biology.

One decision tree is inherently comprehensible since its decisions can be plotted out, illustrating the decisions made transparently (see Figure 1.5). This comes at some cost: A single decision tree is unlikely to model complex problems. Picking models is often a trade-off between accuracy and interpretability. RFs utilise many decision trees in an ensemble which detriments transparency. On the other end of the spectrum, there are deep learning methods which can predict on complex problem domains but are often treated as a black box; One of the big challenges to date is to understand their decision making.

Probing is the process of estimating feature importance by examining model parameters or structures. When training decision trees, each split decreases the impurity of the data set in respect to a specific feature, allowing to estimate feature importance by measuring total impurity reduction. For RFs, one can mean over this score for all trees. Probing SVCs, Logit and linear regressions is achieved by looking at the absolute weights of each feature ($|\beta|$) - assuming the data was normalised, bigger weights mean bigger feature influence. (Azodi et al., 2020)

Feature importance can also be determined through *Sensitivity Analysis*: Manipulating one feature at a time by either leaving it out or meaning over it, and measuring the change in predictions. The bigger the change, the more important a feature is. This technique can be applied to any ML algorithm. (Azodi et al., 2020)

Both techniques are unreliable if features are heavily correlated. If two features have a high degree of shared information, probing them might either split feature importance between both, or estimate one to be vastly more important than the other. Leaving either out during sensitivity analysis might not affect performance much, but leaving both out might drop results significantly. (Azodi et al., 2020)

Recursive Feature Elimination (RFE) (Guyon et al., 2002) mitigates this problem by combining both methods. Feature importances are estimated through probing, and the weakest N features are removed. The model is re-trained on the remainder of features, and this process is repeated until a desired number of strongest features remain. The elimination path is equal to the reverse ranking of features. Small N values return more accurate rankings. This method is applicable for a moderate amount of features and if the underlying ML algorithm can be trained repeatedly on a smaller feature set, which normally does not apply to deep learning methods.

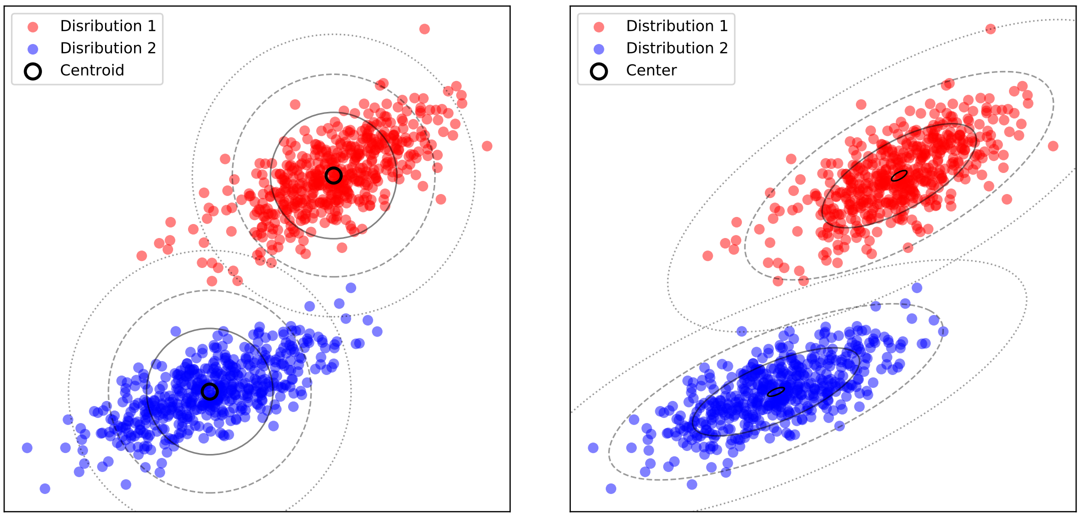
Deep Neural Networks can be analysed by investigating their model weights. CNN kernels can reveal some insights into their working. Alternatively, one can also track their *gradients*. This however is much more complicated due to the hierarchical structure and co-dependency of weights. Early approaches tracked gradients between a neutral input (i.e. a black image or DNA made up of the unknown nucleotide 'N') through the network using backpropagation to derive which inputs affect the gradients most (Baehrens et al., 2010; Simonyan et al., 2013). Tracking gradients was shown to be non-optimal since they might flatten at an input despite it being important, which can be overcome by integrating them (Sundararajan et al., 2017). These *Integrated Gradients* can then be used to visualise input feature contributions for a specific data point.

Integrated Gradients are *local explanations*, i.e. they explain feature contributions for a specific data point; RFE is a *global explanation* over the whole corpus of data. Both techniques are used in this work to shed light into the workings of ML algorithms in the context of splicing.

1.3.2 Unsupervised ML

While Supervised ML is about training an algorithm on an annotated ground truth, *unsupervised Machine Learning* has no desired output annotation. Instead it detects patterns in the data itself in order to provide new insights. There are two main areas of unsupervised ML - clustering and dimensionality reduction.

1.3.2.1 Clustering



(A) (B)

FIGURE 1.10: **Comparison between K-Means and Gaussian Mixtures on synthetic data.** Both algorithms, K-Means and Gaussian Mixtures, need the number of clusters *a priori*, and both fit centroids to the data so that they converge to the centre of their surrounding data points. The main difference is in their assumption about the data distribution. (a) The assumption of circular distribution does not hold for this artificial dataset, meaning it will cluster some data points incorrectly. (b) The centres of a Gaussian Mixture converged to about the same location as with K-Means. But the clustering fits the data distribution better due to respecting covariances and (in this example) will not mis-cluster data points.

K-Means is a simple clustering algorithm that finds *K centroids* (reference points in N-dimensional space) to the data, one centroid per cluster (see Figure 1.10a). The number of centroids *K* needs to be defined in advance.

Centroids are initialised (pseudo-) randomly. Each data point is assigned to the closest centroid, and the centroid is moved to the mean position of all assigned data points. This process is repeated until centroid positions converge. (Jain et al., 1999)

The algorithm has some known drawbacks such as stochasticity due to random initialisation and the assumption of spherical data distributions. Stochasticity can be compensated through repeated application.

The assumption of spherical distributions can be relaxed: *Gaussian Mixtures* (Figure 1.10b) allow modelling of feature co-dependency through a hyper parameter defining the shape of the covariance matrix between features. A further advantage of Gaussian Mixtures over K-Means is that they model the probability for each data point belonging to each cluster (i.e. a *mixture* of mappings), which allows soft partitioning as opposed to a hard assignment of each data point to exactly one cluster. (Jain et al., 1999)

1.3.2.2 Dimensionality Reduction

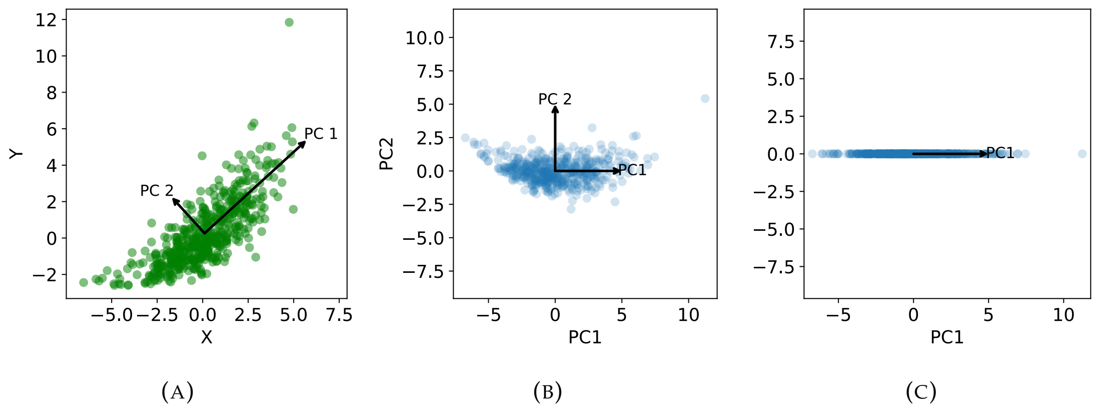


FIGURE 1.11: **PCA finds principal components and projects data onto them.** (a) The first principal component is the eigenvector of the data where it has the highest variance. All following components lie orthogonal, if multiple components are possible (higher dimensional data), the one with highest variance is chosen. (b) This PCA projects the data onto two components. Dimensionality stays the same, but the data is rotated and rescaled. (c) Projecting the data onto the first component reduces the dimensions to one but preserves the axis with highest variance. If one would just drop the Y axis from the original data, the outlying point at the top-right corner would not be distinct any more.

Principal Component Analysis (PCA) is a popular algorithm that projects data from high to low dimensional space using *Singular Value Decomposition* (SVD). Projecting to two or three dimensions allows plotting a representation of high dimensional data. For N target dimensions, PCA projects data on N principal components. The first principal component is extracted by finding the direction that retains the highest variance, which is the *eigenvector* of the data. All following principal components are orthogonal hyper planes, again retaining highest variance in the data. The data is transformed by projecting it on each hyper plane. (Wold et al., 1987)

Conventional PCA needs to hold all data in memory and perform batch operations on it, which is infeasible for big datasets. *Incremental PCA* resolves this by incremental SVD calculations over batches of data. Due to the immense number of splice sites in the human genome, incremental PCA is used when plotting the data in chapter 3.

The PCA projection is linear and works best on multi-variate normal distributions. A drawback of PCA however is that close points in high dimensional space do not necessarily end up close to each other in low dimensional space.

If one desires neighbours to be preserved, a better approach is to apply *T-distributed Stochastic Neighbour Embedding* (t-SNE). This non-linear, neighbour-preserving algorithm transforms the similarity (or, originally the euclidean distance) between any two points into a probability of them staying next to each other in low dimensional space. The low dimensional space is constructed to retain this probability distribution as closely as possible by reducing entropy (*Kullback–Leibler divergence*). (Van der Maaten and Hinton, 2008)

While the projections of PCA can be repeated for unseen data, t-SNE cannot be applied to new data points without some new predictive measures, which is why t-SNE is rarely found in supervised ML pipelines.

1.3.3 Types of Data

ML needs *numerical data* such as age, height, or white blood cell count. *Ordinal* data such as non-smoker / rare smoker / regular smoker have a hierarchy and can be translated to numerical data, as can *binary data* (immunosuppressed /not immunosuppressed). (Géron, 2019)

This leaves *nominal* data such as acceptor / donor / neither, natural language and DNA nucleotides. These values are often *one-hot encoded*, which is equal to transforming each distinct value into a vector of zeros with exactly one one (Géron, 2019). The position of the one is unique per *term* in the *vocabulary*. For example, a one-hot encoding of the vocabulary acceptor / donor / neither could be $[1,0,0]$ / $[0,1,0]$ / $[0,0,1]$.

It's easy to see that the encoding vector grows with the number of distinct values. If the English language with a vocabulary of around 150,000 words serves as the vocabulary corpus, encoding a single word results in about 150,000 zeros and a single one. Some arising technical challenges such as memory and disk usage have been resolved by software engineering. However many ML algorithms work suboptimally on such sparse data. For example ANNs need to multiply and add many zeros unnecessarily. One-Hot encoding also worsens the *Curse of Dimensionality*, unfavourable properties of geometry and distance measures in high dimensional space (Verleysen and François, 2005).

Another notable shortcoming of one-hot encoded data is that since every term has its own dimension, all terms have the same distance to each other. Ideally, similar terms should be closer to another and opposing terms further apart. Such a *semantic space* can be generated through *Word2Vec* (Mikolov et al., 2013), which leans how to map one-hot encoded data to vector space given only natural language. Application of Word2Vec to DNA and splicing is described in chapter 5.

1.3.4 Handling Imbalanced Datasets

A balanced dataset means that all outcomes are (roughly) equally frequent in the data. If that's not the case, there are various implications and pitfalls.

For example, on a dataset where only 1 in 1,000 patients has a disease, a classifier predicting everybody to be healthy would achieve an accuracy of 99.9%. Training a classifier on such an imbalanced dataset might also skew its abilities to distinguish the two classes. However, simply balancing the data to a 1/1 ratio through down-sampling will remove a big partition of data and might make the problem seem easier, also affecting training score. Sampling up to a 1/1 ratio would need a method to generate new or resample existing data, which introduces new, domain specific challenges. (Ganganwar, 2012)

In any case, imbalanced datasets require special care during evaluation, and using AUC-PR or average precision score should be preferred in order to account for imbalances.

Chapter 2

Methodology

The research presented in this work is highly computational and great care needs to be taken to produce a sound technical development strategy both locally and on the university's *High-Performance Computer* (HPC) called IRIDIS. While every chapter focuses on different research questions with distinct objectives, the overall ML approach is often similar.

This chapter is dedicated to the development practices and software engineering required in the following experiment chapters. Each chapter may then extend this general framework with its own methods as required.

2.1 General Technical Framework

ML code is developed locally using a small development data set containing less than 50 splice sites across 8 genes. This data set can be processed almost instantly and allows code to be tested for syntactic errors. Code is debugged locally using the python debugger provided by *Visual Studio Code*.

Bash scripts coordinate the queue scheduler *slurm* (Yoo et al., 2003) used by IRIDIS5 and utilise a custom implemented helper library that abstracts slurm functions, allowing scripts to be tested locally first. This prevents slurm scripts with syntactic errors being stuck in a queue just to fail immediately once they are scheduled to start.

Python dependencies and environments are maintained using *conda*. This allows (almost) the same dependencies to be used on IRIDIS and locally, minimising integration error.

The published SpliceAI code (McRae et al., 2019b) utilises *TensorFlow-GPU* (Abadi et al., 2016) version 1.4.1 and *keras* version 2.0.5 in *python2* (Van Rossum and Drake Jr, 1995). Python2 has been deprecated since the beginning of 2020, however migrating to newer versions caused many problems due to dependencies. The newly developed CI-SpliceAI is contained in its own *python3* (Van Rossum and Drake, 2009) project.

DNA2Vec (Ng, 2017a) uses *gensim* (Řehůřek et al., 2011) in a python3 implementation, forcing the main project to be split into two branches with their own dependencies.

BEDtools (Quinlan and Hall, 2010), *BEDOPS* (Neph et al., 2012) and *pyfaidx* (Shirley, 2014) provide genetic tooling. Big data sets are stored using *hdfpy* (Colette, 2014). Genomic variant annotations are created using *Ensemble Variant Recoder* (Ensembl, b) and parsed using *Ensemble Variant Effect Predictor* (Ensembl, a); coordinates are lifted over using *hgLiftOver* from *University of California, Santa Cruz* (UCSC) (University of California, a).

Most ML algorithms, except for CNNs and DNA2Vec, are implemented with *scikit-learn* (Pedregosa et al., 2011). Python standard toolings (*numpy* for maths, *pandas* for csv files, *requests* for web communication, *xlrd* for excel support) were used where appropriate.

Data visualisations were created using *matplotlib* (Hunter, 2007). Visualisations of intron/exons were implemented from scratch using *matplotlib* as well. Results were obtained locally and through the university's IRIDIS HPC where needed.

Code is versioned through *git* which is also used to deploy to IRIDIS.

Due to extensive local testing, integration and deployment on IRIDIS was mostly without problems. Due to differences in the operating system and hardware configuration between IRIDIS and the local machine, updating dependencies and tracking bugs due to version differences were time consuming and needed manual research. Especially migration towards newer Keras/Tensorflow slowed down development substantially due to their untransparent constraints to each other as well as CUDA (NVIDIA et al., 2020a) and CuDNN (NVIDIA et al., 2020b) versions.

2.2 Supervised Training Methodology for Splice Site Recognition

2.2.1 Performance Measure

Due to the imbalanced data set, accuracy cannot be used as a performance measure. To produce comparable results, the performance measure of SpliceAI, the mean average precision score[1] of the binary classification tasks acceptor / rest and donor / rest, was adopted.

[1]The SpliceAI authors refer to the average precision score as AUC-PR in their publication

2.2.2 Hyper Parameters for Baseline Classifier

Baseline algorithms (SVC, Logit, RF, and MLPC) first need their hyper parameters optimised for a given data set. This is done using GridSearchCV with three partitions (random partitions ignoring paralogs). For each model, the best hyper parameters are found by selecting the one that maximises mean AUC-PR on the test splits.

2.2.2.1 Support Vector Classifiers

All SVCs are configured to use one-versus-rest classifications and 10,000 max iterations.

Three separate grid searches, one for each SVC kernel (linear, RBF and polynomial) are conducted. Each spans over the grid of b that compensates class imbalances (true/-false), l_2 regularisation coefficient $C \in \{0.0001, 0.001, 0.01, 0.1, 1, 10, 100\}$ and regularisation parameter $\gamma \in \{0.0001, 0.001, 0.01, 0.1, 1, 10\}$. The grid of the polynomial SVC includes polynomials of degree $d \in \{2, 3, 4\}$.

2.2.2.2 Logistic Regression

The base configuration was set to use one-versus-rest classification, 2,000 max iterations and utilise the *saga* solver, which scales to bigger data sets and allows l_1 and l_2 penalties. The dual parameter was further set to false, which is recommended when $n_{\text{samples}} > n_{\text{features}}$. The grid was initialised over regularisation $l \in \{l_1, l_2\}$ penalties using $C \in \{0.001, 0.01, 0.1, 1, 10\}$.

2.2.2.3 Random Forest

During search, the RF was configured to utilise 20 trees, each with a max_depth $d_m \in [10..25]$. When a good hyper parameter configuration was found for 20 trees (*RF20*), the forest is also extended to 500 trees (*RF500*). This assumes that parameters found for a small ensemble are transferable to bigger forests.

2.2.2.4 Multi-Layer Perceptron Classifier

The hidden layer configuration of a MLPC is documented as tuples, where each number represents the number of hidden neurons in that layer. The configuration $(50, 100)$ therefore represents a four layer MLPC with two hidden layers that have 50 and 100 neurons respectively.

The grid was configured over the hidden layer configurations
$L \in \{(50,), (100,), (300,), (500,), (100, 20), (20, 100)\}$.

2.2.3 Hyper Parameters for SpliceAI

The SpliceAI10k architecture was not changed and is depicted in Figure 1.8, page 16.

2.2.4 Training and Testing Methodology

2.2.4.1 Baseline Classifiers

After hyper parameters are found as described in section 2.2.2, all baseline algorithms are evaluated using CV on partitions shown in table 3.1 (page 42). This is done on the whole corpus of data; if algorithms are not able to return CV results after 60 hours, the process is repeated on a subset of data (as described in experiment design). Mean and standard deviation of the performance measures of test partitions are reported.

2.2.4.2 SpliceAI

The code to train and test SpliceAI (McRae et al., 2019b) was ported to the IRIDIS system.

Five independent CNNs are trained in total and used as an ensemble predictor. Each CNN is split on two GPUs and trained using stochastic gradient descend, adam optimizer, categorical cross-entropy loss and a batch size of 12. Each training epoch iterates over a random chunk representing 100 successive genes, each gene represented by slices of 15,000 nucleotides (10,000 nucleotides of context and 5,000 nucleotides to be predicted). Training is repeated for 10 times the number of chunks. The learning rate starts at 0.001 and halves at 60%, 70%, 80%, and 90% of training.

SpliceAI is not evaluated using CV because of 1) its computational complexity and 2) to produce comparable scores to the ones published by the original paper (Jaganathan et al., 2019), instead it is trained on chromosomes 2, 4, 6, 8, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, X, and Y.

The ensemble of 5 CNNs is then tested on the test partition 1 of table 3.1 (chromosomes 1, 3, 5, 7, 9 excluding paralogs) using a batch size of 6. The ensemble averages over all predictions, and the performance on the test set is reported.

2.2.5 Explaining Supervised ML Decisions

To explain how the trained classifiers reach their conclusions, RFE will be applied to baseline algorithms and Integrated Gradients to SpliceAI models (see section 1.3.1.11).

2.2.5.1 Explaining Baseline Classifiers

Feature contributions towards decisions of baseline classifiers are determined using the RFE implementation provided by *scikit-learn* (Pedregosa et al., 2011), unless stated otherwise. The weakest four features are removed at a time and the classifier is retrained on the remainder. This process is repeated until four features or fewer are left. The ranking of features is then plotted out with a cut-off so the top 20% of features are visible.

This implementation determines feature importance via probing (section 1.3.1.11), which is not available for neural networks. While a custom probing algorithm for neural networks could be implemented in theory, it was out of scope of this project; therefore MLPCs cannot be interpreted.

2.2.5.2 Explaining Deep Learning / SpliceAI

Feature contributions towards predictive classes were calculated using Integrated Gradients (Sundararajan et al., 2017): Gradients between a neutral input and a concrete example are propagated from the output back through the network to the input. By approximating gradient integration though linear interpolation, feature contributions are be derived.

To calculate this, an Integrated Gradients implementation for keras (Hiranuma, 2018) was adopted. This implementation uses the keras API, requiring all calculations to be done within the keras framework.

This algorithm is applied to the published SpliceAI models. The published SpliceAI code (see section 2.2.4.2) calculates the ensemble average using numpy, which would not allow the calculation of gradients within keras. Ensemble averaging was therefore migrated into the keras framework. This would however cause all five CNNs and the averaging to be calculated on the GPU at the same time, which consumed too much memory. Keras was therefore configured to run on the *Central Processing Unit* (CPU); this was possible as the networks were already trained and prediction on a CPU is performant enough. While migration to keras was relatively straight forward for splice site recognition, adaptation to variant annotation was significantly harder as described in chapter 4.3.3 (page 74).

To provide explanations for the classification result at a specific basepair, a function that returns DNA slices of one nucleotide at an arbitrary offset plus 10k flanking context was created. A neutral output of same length is generated, this is the all-zero vector which represents the unknown nucleotide N. Gradients between the neutral input and a specific input are calculated, traced back through the network, and integration is approximated with 50 steps.

The Integrated Gradients return one measure per input feature representing feature contributions. For a single decision, i.e. splicing or not, this returns 10,001 feature contributions (10,000 of flanking context plus the nucleotide in question). Contributions can be both positive and negative and are plotted in frequency diagrams, similar to logo visualisations used to analyse PWMs.

Feature contributions, both negative and positive, are used to derive the height of each letter / nucleotide. Negative weights cause their letter representations to be drawn downwards. Loci don't need to sum up to 1 as they would do in a frequency diagram. To aid visualisation of small contributions, feature contributions are also depicted using a symmetrical $\log_{10}$ transform with a linear threshold component. The linear component prevents the log transform to output extreme numbers near zero and is estimated by the biggest absolute feature contribution divided by 25. This value resulted in a trade-off between the visualisation of small weights and retained some indication of their relative differences.

2.3 Data and Ethics

Patient variant data was sourced from publications (Ellingford et al., 2019; Houdayer et al., 2012; Ito et al., 2017; Jian et al., 2014; Leman et al., 2018; Maddirevula et al., 2020). One source was not published yet during early experimentation; the study was cleared through the university ethics committee (ERGO II 23056.A1). Since then, the data was published in Wai et al. (2020).

All remaining data used has been previously published. This work incorporates the Human Reference Genome (Church et al., 2011; Schneider et al., 2017), GENCODE (Frankish et al., 2019), gene annotations from Ensembl (Yates et al., 2020) and the *National Center for Biotechnology Information* (NCBI) (NCBI; Pruitt et al., 2002), REST APIs from Ensembl (Ensembl, a) and UCSC (University of California, a), splice sites from McRae et al. (2019b), protein binding sites from Cáceres and Hurst (2013); Venables (2007); Wang et al. (2013, 2004), and conservation scores from University of California (b), namely *phyloP* (Pollard et al., 2010) and *PhastCons* (Siepel et al., 2005).

Chapter 3

Application of Machine Learning to Splice Sites

3.1 Introduction

One of the main goals of this thesis is to improve diagnosis of splicing related disease using ML. Before doing so, we first need to find and evaluate algorithms to model and predict the splicing process itself on the Human Reference Genome, i.e. only using healthy sequence data. The more accurately splice sites can be recognised, the better these algorithms are expected to perform when applied to clinical variants later on.

This chapter describes the application of ML algorithms to classify any position in the Human Reference Genome into acceptor, donor, or neither. To do so, a training dataset is created and analysed using statistical tools and unsupervised ML.

As this is an application of data science, great care needs to be taken during data set creation and the data needs to be validated. The data should be up to date to include any fixes in data sources and minimise noise. Different algorithms need different data set formats: Deep learning (i.e. CNNs) can utilise massive parallel processing to output a matrix of predictions and therefore train on very big genomic slices; simpler models (*baseline classifiers*) can only emit one class at a time and need to be trained and tested on a smaller dataset. Algorithms that need a long time to train need a further subset of data. Validation of classifiers needs to ensure that the split in train and test data is not lucky or unlucky. To mitigate this issue, a CV partitioning table needs to be introduced and utilised when feasible.

Two main variations of the data need to be introduced: One to only contain strong splice sites and one containing all splice sites. These two variations will be compared to each other throughout.

Different supervised ML algorithms are trained and evaluated on this data, and the decision making of the best algorithms are investigated.

3.1.1 Aims and Objectives

This chapter relates to the general objective 1: Develop, test, and analyse algorithms to recognise splice sites.

First, two main datasets are to be created: 1) One that includes all *primary* transcripts, i.e. those predominantly expressed in the human body, by re-implementing the SpliceAI GENCODE data pipeline; and 2) a new dataset that additionally includes weaker GENCODE splice sites, *collapsed* into one pseudo-transcript. To ensure that these two datasets are representative of the problem, and to quantify and compare the problem complexity, the two datasets are visualised: Both datasets are plotted and clustered; motif frequencies are compared to literature; and the *primary* dataset is compared to the data published with SpliceAI.

To explore if deep learning is actually needed for this problem domain, *baseline classifiers* (i.e. SVCs, Logits, RFs, and MLPCs) are to be optimised and trained. To do so, the two datasets need to be adjusted and reduced in size. Cross-validation is introduced where appropriate to help quantifying performance of algorithms. Algorithmic performance is then compared to SpliceAI. The training process of the SpliceAI architecture is ported to the high performance system IRIDIS to allow training CNNs on the newly created data.

Lastly, splice site classification results are to be explained by investigating feature contributions using RFE and Integrated Gradients on baseline classifiers and CNNs respectively.

3.2 Background

3.2.1 Public Sources of Genomic Sequences and Annotations

The *Human Reference Genome* is curated from anonymous donors of mostly Western heritage. The latest assembly at the time of writing is published as *Genome Reference Consortium human* (GRCh) on build 38 version p13 (*GRCh38.p13*). The build indicates the version of *genomic coordinates* that specify which offset each gene lies on, and the version relates to how new the data is respective to this build. GRCh37 is deprecated, and conversion between GRCh37 and GRCh38 is called *lifting*. (Church et al., 2011; Schneider et al., 2017)

The one big advantage of the Human Reference Genome is also its biggest disadvantage: It contains exactly one fictitious genome. This has two drawbacks: 1) It cannot possibly cover all benign variants and 2) it is inherently biased towards the heritage of the individuals that donated their sequences (mostly American/African, Schneider et al., 2017). On the other hand, the Human Reference Genome is broadly used and believed to be healthy.

There are many approaches to resolve the first challenge by including more benign sequences. *GenBank* (Benson et al., 2017), maintained by the American NCBI, is a database containing sequences submitted by many laboratories from around the world and updated monthly. GTEx (Carithers et al., 2015) is a major collection of reference RNA data with an emphasis on tissue specific expression. Samples were collected post-mortem from just under a thousand human donors in various tissue across the body. Most patients died from traumatic injuries and heart disease. Patients suffering from immunodeficiency or metastatic cancer were excluded, however some samples collected may be related to hereditary disease. The *100,000 Genomes Project* (England, 2016) is an ongoing effort to sequence many genomes from British patients.

Controlling for heritage and ethnicity however is much more challenging both logistically and scientifically. The *1,000 Genomes Project* (Siva, 2008), a predecessor of the 100,000 Genomes Project, included more ethnically diverse genomes from healthy patients. The biggest effort to date to diversify sequencing is *gnomAD* (Karczewski et al., 2020), which analysed and stratified over the genetic heritage of their sample donors.

Based on these reference sequences, there are databases to annotate genes, transcripts, and exons. *GENCODE* (Frankish et al., 2019) is a project annotating genes of the Human Reference Genome with a primary focus on protein coding regions. This includes *Ensembl* (Yates et al., 2020) annotations, derived from clinical experimentation pipelines, and *HAVANA*, a corpus of manually annotated data. Genes, transcripts and exons are identified by their Ensembl ID. GENCODE includes annotation levels to distinguish HAVANA from Ensembl transcripts: Level 1 represents verified annotations, level 2 manual annotations, and level 3 are automatically annotated instances.

The main alternative to GENCODE, curated by the European EMBL-EBI, is *RefSeq* (Pruitt et al., 2002) which is mainly developed by the NCBI and based on their GenBank data. It annotates genes, transcripts, exons, and even some common variants.

Whereas many bioinformaticians might choose between GENCODE and RefSeq depending on whether they reside in the Americas or Europe, Frankish et al. (2015) shows that GENCODE is ”richer in alternative splicing, novel in CDS [protein coding sequences], novel exons and has higher genomic coverage than RefSeq”.

The two maintainers joined efforts to form the *Matched Annotation from NCBI and EMBL-EBI* (MANE) collaboration and published an approach to finding transcripts of high quality, which curated two main collections of transcripts for 97% of protein coding genes: The MANE Select set which consists of one representative transcript per gene and the MANE Plus Clinical set that curates more than one transcript where clinically relevant (Morales et al., 2022). MANE also directly maps Ensembl and *RefSeq* identifiers to standardise the field and bring together the two systems.

The SpliceAI authors train their final model on the Human Reference Genome as input to predict annotations derived from *primary* GENCODE transcripts combined with novel splice junctions observed in at least 5 GTEx samples (Jaganathan et al., 2019). Their publication also discusses an intermediate model that only recognises primary GENCODE transcripts.

3.2.2 Selection of Primary Isoform for Training Data

Figure 3.1 shows all isoforms for the gene *SH3YL1* contained in GENCODE v37GRCh38. Not all isoforms contain all splice sites. When parsing this data set for Machine Learning, using each isoform as a separate data point would annotate ambivalent ground truth and hinder training.

Instead, Jaganathan et al. select one isoform to generate an unambiguous ground truth with exactly one input mapping to one output. What exactly they determine to be the primary transcript however is not accurately described in their publication, and attempts to clarify this with the authors were unsuccessful (Erdem and Jaganathan, 2021). At the time their paper and most of this dissertation was written, the MANE sets were unpublished, meaning MANE Select transcript could bot be not be used instead. The selected GENCODE isoform is then enriched with novel splice sites observed in at least five GTEx samples, creating a pseudo-isoform.

This approach however might introduce significant problems. First, the enrichment of novel splice sites from five samples might introduce sequencing noise that has to be validated by humans, and it is unclear how the threshold of five samples affects the number of false positives. Secondly, and more importantly, it decouples the ML in- and output data. The authors included novel RNA splice sites of patients, but still train on the DNA of the reference genome. If a patient suffered from hereditary and splice-related disease, which was not an exclusion criterium in the GTEx study, their genes might be subject to different or aberrant splicing compared to the reference. This potential discrepancy might train the CNNs to try to associate healthy DNA with abnormal expression.

From the visualisation it is apparent that a number of exons are left out and some potential pseudo-exons where included, which might skew both train and testing of SpliceAI. Not only does the training data have a significant number of novel splice sites, some of which could be false positives, there are also some false negatives, i.e. sites that

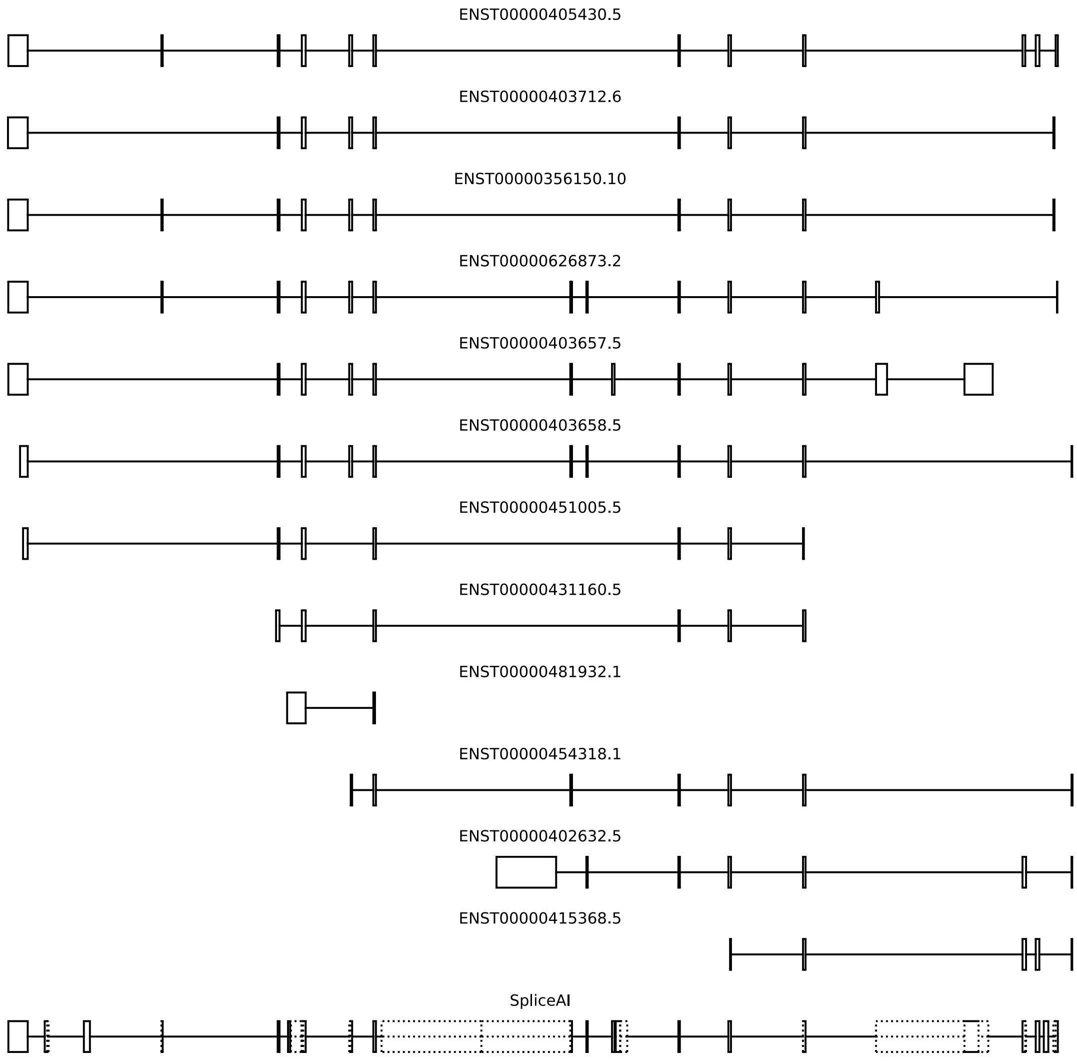


FIGURE 3.1: **All *SH3YL1* isoforms on GENCODE v37GRCh38, and the training data used for SpliceAI.** *SH3YL1* is the first gene on the SpliceAI training data with more than one validated transcript. All transcripts are level 1-2 (verified and/or annotated by humans). The first one is the one Jaganathan et al. selected as a *primary isoform*, which was then enriched with novel junctions from GTEx which results in the pseudo transcript of the last row. Ambivalent regions that can code for both exons and introns are depicted by dotted lines. On this gene, 26 (46%) of splice sites in the SpliceAI training data are novel GTEx sites and have not been validated by humans; 7 (18%) of verified splice sites are not present in SpliceAI training data.

human researchers annotated or verified in GENCODE. Because of the selection of only the primary GENCODE isoform, validated sites were excluded from the SpliceAI data. A possible consequence is that if a mutation disrupts a site that is not spliced in this training transcript, the algorithm might falsely classify it as benign, and vice versa. To improve clinical application, a more straight forward approach would be to combine all transcripts that were annotated and/or checked by researchers and have been observed relative to the reference genome.

The choice of GENCODE over other data sources is probably a computational one to facilitate ML model development. GENCODE annotates a single input sequence from the Human Reference Genome that SpliceAI can be trained on. The training process was optimised for hardware acceleration, which entails training on one long RNA input sequence (15,000 nucleotides) with 5,000 splice annotations, which can directly be extracted from the Human Reference Genome and GENCODE. Other data sources discussed earlier, such as GenBank, are not based on a single input sequence. While these alternative data sources could widen the scope and remove bias towards the reference genome, it would also require more storage, memory, and training time. It was therefore decided to retain the data pipeline and base annotations on GENCODE and the reference genome.

3.2.3 Logo Visualisation of Position-Weight-Matrix Analysis

Splicing motifs are often depicted as *logos*, diagrams of nucleotide sequences that were scaled according to frequency. This technique can also help debugging the data pipeline as we can compare its output to literature. One particularly helpful way of doing so are *logo visualisations* of PWMs; this method uses Information Theory to quantify how *surprising* the observation of a certain nucleotide frequency is.

We can measure how much information observations bear using the logarithmic function (Van der Lubbe and Hoeve, 1997). When the log_2 function is used, this unit is called *bits* (binary digits). Bits can also be used as a measure of how surprising the outcome is. In DNA, we can observe four different nucleotides. Under the assumption that nucleotides occur with the same background probability of $b = \frac{1}{4}$, always observing a specific nucleotide at a specific location would be $-log_2(b) = 2$bit surprising, which is the maximum measure of surprise for DNA.

A PWM (Stormo et al., 1982) finds motifs and measures their information gain compared to b using information theory. By aligning[1] splice sites and deriving nucleotide frequencies, we can create a *Position-Probability Matrix* (PPM) which specifies the probability of observing each nucleotide at an offset. The matrix P describes a PPM where

[1]*Alignment* does not refer to sequence alignment of smaller fragments, in this context DNA slices are simply stacked upon so that splice sites are at the same position

$P_{n,p}$ is the probability of observing nucleotide n at position p. Under the assumption that all four nucleotides have the same base probability b, we can then can derive a PWM M by using the log transform $M_{n,p} = log_2(P_{n,p}/b)$.

The PWM motifs are often drawn as graphs referred to as *logo*, where the information of each nucleotide is represented by a scaled letter; letters at each position are stacked on top and often sorted by size and coloured. Figure 3.2 shows an example PWM logo.

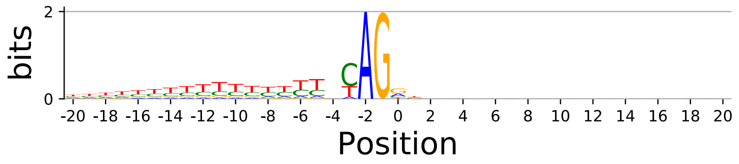


FIGURE 3.2: **PWM Logo visualisation of acceptor sites.** 2 bits mean that this base is observed at every single nucleotide. If there are no letters visible, there is no significant difference to the expected background frequencies b.

3.3 Methods

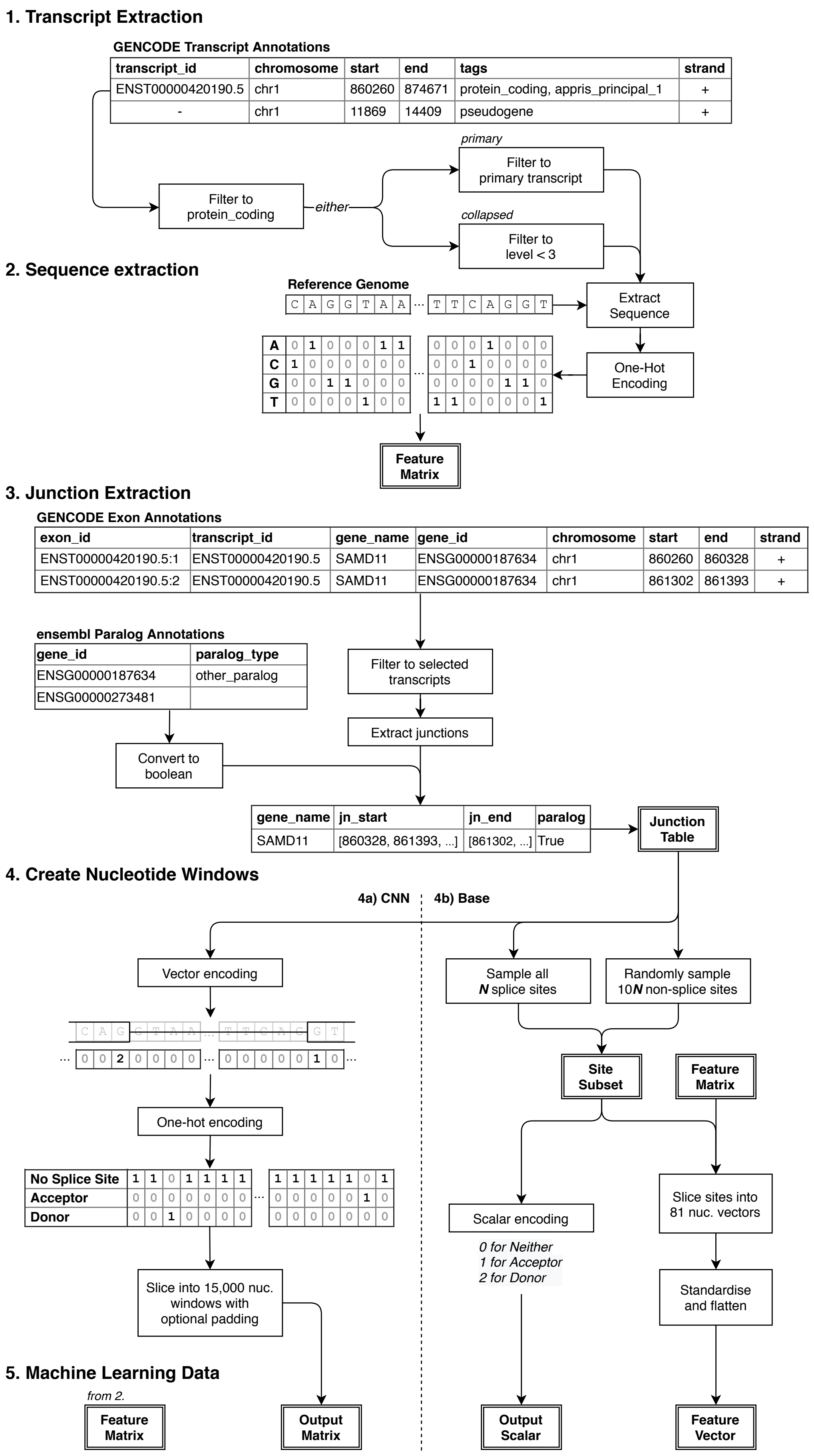
3.3.1 Adapting the Data Pipeline of SpliceAI

It was decided to re-implement the SpliceAI data pipeline (Jaganathan et al., 2019) as, at the time of writing, it is the most popular and promising tool. Figure 3.3 1-4(a) illustrates this process with example data for CNNs.

GENCODE protein coding transcripts are extracted and filtered. Transcripts are filtered for either the principal transcript (creating the *primary* dataset, see section 3.3.1.3), or for the GENCODE annotation level<3 (creating the *collapsed* dataset, section 3.3.2). Sequences are extracted from the Human Reference Genome and one-hot encoded; transcripts on the negative strand are reverse-complemented. This will be referred to as *feature matrix*; its further processing will differ between CNNs and baseline classifiers (sections 3.3.1.1 and 3.3.1.2 respectively).

Exon annotations are matched to their transcripts using their transcript ID[2]. Exon boundaries that align with a transcript start or end annotation are removed since they represent non-splice sites, implicitly removing transcripts/genes consisting of only one exon. Paralogs are annotated by joining GENCODE with *Ensembl* (Yates et al., 2020) data, using gene ids to match the two[2]. Paralogs are later excluded from testing, removing potential overlap between training and test data due to gene repetition.

[2]This detail was not described by Jaganathan et al.



GENCODE Transcript Annotations

transcript_id	chromosome	start	end	tags	strand
ENST00000420190.5	chr1	860260	874671	protein_coding, appris_principal_1	+
-	chr1	11869	14409	pseudogene	+

GENCODE Exon Annotations

exon_id	transcript_id	gene_name	gene_id	chromosome	start	end	strand
ENST00000420190.5:1	ENST00000420190.5	SAMD11	ENSG00000187634	chr1	860260	860328	+
ENST00000420190.5:2	ENST00000420190.5	SAMD11	ENSG00000187634	chr1	861302	861393	+

ensembl Paralog Annotations

gene_id	paralog_type
ENSG00000187634	other_paralog
ENSG00000273481	

gene_name	jn_start	jn_end	paralog
SAMD11	[860328, 861393, ...]	[861302, ...]	True

1. Transcript Extraction
Filter to protein_coding
either
primary
Filter to primary transcript
collapsed
Filter to level < 3
2. Sequence extraction
Reference Genome
C A G G T A A ... T T C A G G T
Extract Sequence
One-Hot Encoding
A 0 1 0 0 0 1 1 ... 0 0 0 1 0 0 0
C 1 0 0 0 0 0 0 ... 0 0 1 0 0 0 0
G 0 0 1 1 0 0 0 ... 0 0 0 0 1 1 0
T 0 0 0 0 1 0 0 ... 1 1 0 0 0 0 1
Feature Matrix
3. Junction Extraction
Filter to selected transcripts
Extract junctions
Convert to boolean
Junction Table
4. Create Nucleotide Windows
4a) CNN
4b) Base
Vector encoding
C A G G T A A ... T T C A G G T
... 0 0 2 0 0 0 0 ... 0 0 0 0 0 1 0 ...
One-hot encoding
No Splice Site 1 1 0 1 1 1 1 ... 1 1 1 1 1 0 1
Acceptor 0 0 0 0 0 0 0 ... 0 0 0 0 0 1 0
Donor 0 0 1 0 0 0 0 ... 0 0 0 0 0 0 0
Slice into 15,000 nuc. windows with optional padding
Sample all N splice sites
Randomly sample 10N non-splice sites
Site Subset
Feature Matrix
Scalar encoding
0 for Neither
1 for Acceptor
2 for Donor
Slice sites into 81 nuc. vectors
Standardise and flatten
5. Machine Learning Data
from 2.
Feature Matrix
Output Matrix
Output Scalar
Feature Vector

FIGURE 3.3 (cont.): **Data Pipeline.**
1) GENCODE transcripts are filtered for protein coding, and optionally for either the primary transcript or transcript level. Ensembl provides paralog annotations.
2) Transcripts are extracted from the reference genome, building the reverse complement for negative stranded genes. A numerical transcript matrix representation is created using one-hot encoding.
3) Junctions are calculated by parsing GENCODE exon annotations. The first and last exon boundaries are removed so that only splice sites are retained, implicitly dropping all transcripts that consist of only one exon. Junctions are then encoded into a vector that indicates where exons start and end (acceptor/donors).
4) From both, the features and exon data (*ground truth*), nucleotide windows are extracted depending on the algorithm. 4a) For the CNN, the exon vector is one-hot encoded and each gene is sliced into windows of 15,000 bases, short genes are padded with *unknown* (dedicated one-hot representation). 4b) For baseline classifiers, all splice sites and ten times more non-splice sites are sampled from the exon vector. A nucleotide window of 40 bases left and right around each site is sliced to provide input, forming a window of 81 nucleotides. The ground truth for each window is a scalar determining if the nucleotide in the middle is an acceptor, donor, or neither. Data is then standardised by subtracting the mean and dividing by standard deviation
5) The features and ground truth are split into training and test sets chromosome-wise. Paralogs are excluded from test split. The algorithms are trained and tested, and a score is calculated for comparison.

3.3.1.1 CNN specific encoding

Following SpliceAI, each transcript is enlarged by 5,000 nucleotides on both sides to provide 10,000 nucleotides of context. For this enlarged window, sequences are extracted from the *feature matrix* and sliced into windows of 15,000 bases, where each slice predicts 5,000 nucleotides at a time. Short genes are padded with 'N' (*unknown nucleotide*) if necessary so that no nucleotides outside of gene annotations are included. This nucleotide vector is then one-hot encoded, resulting in an input matrix of 15,000x4 (A,C,G,T). The N nucleotide is encoded as a vector of all zeros.

For the 5,000 positions in the middle, an exon vector is created for the ground truth, where 0 indicates non-splice sites, 1 indicates the first exon nucleotide at an acceptor, and 2 indicates the last exon nucleotide at a donor position. This vector again is one-hot encoded, resulting in a 5,000x3 matrix.

3.3.1.2 Extension for Baseline Classifiers

In contrast to CNNs, most formulations of splicing predictions with ML algorithms cannot predict an output matrix, but only a single splice site along the sequence. That means, instead of using windows of 15,000 bases to predict annotations for 5,000 nucleotides at once, simpler algorithms can only predict one site at a time. The data therefore had to be further transformed, so it could be used with baseline classifiers.

With more than one billion nucleotides, an inherent challenge is the mass of data. Thanks to hardware acceleration and clever algorithmic design, this immense data volume can be fed to CNNs, but for baseline classifiers this is infeasible with current technology.

A second challenge is the imbalance of the data. Preliminary experimentation has shown that rebalancing to a 1/1/1 (acceptor/donor/neither) ratio by random subsampling simplified the problem too much and skewed classification metrics. It was therefore decided to include ten times more negative splice sites than positives. Sampling negative splice sites also resolves the first challenge for the most part, except for SVCs where the data was further subsampled (see section 3.3.3).

Negative splice sites were sampled from uniformly random[3] positions between transcript start and transcript end that are not marked as splice sites.

Each site is then assigned a numerical output value and the *feature matrix* is cropped to windows of 40 bases to the left and right of each site. This nucleotide window therefore represents 81 nucleotides as an input of a ML algorithm, and the output is 0 for non-splice sites, 1 for acceptor, and 2 for donor sites. The nucleotide sequence was then one-hot encoded and standardised by subtracting mean and dividing by standard deviation (using a partial fitting of 500 samples at a time). This results in a 81x4 input matrix; a sample data point is visualised in Figure 3.4. This matrix is then flattened to a vector (324x1) because baseline classifiers cannot work with multi-dimensional data.

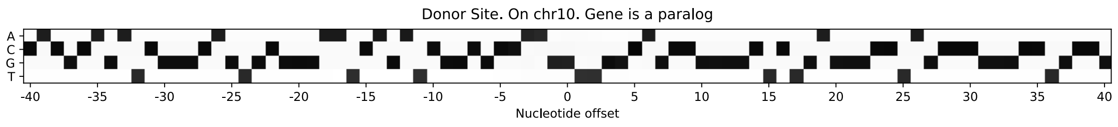


FIGURE 3.4: **One training instance in the base dataset.** The sequence can be read by following the black squares representing ones (resulting in CACG..). The splice site in question is at offset 0 and is a donor site where we can see consensus GT sequence. Matrix is flattened for machine learning.

3.3.1.3 Extracting Primary Transcripts

Filtering for principal transcripts was not straight forward despite GENCODE annotations. Annotations contain *appris_principal* and *appris_principal_X* where $X \in [1..5]$. The transcript(s) with the lowest X were selected[2] under the assumption that *appris_principal* is equal to *appris_principal_1*[2]. If a gene has less or more than one annotated primary transcript, the transcript with the most associated exons is selected[2]. Transcripts on the Y chromosome that also exist on the X chromosome are dropped[2].

[3]Uniform random: All outcomes are equally likely

3.3.2 Collapsing Isoforms into Single Datapoint

If more than one isoform exists because of alternative splicing, Jaganathan et al. select the *principal transcript*. Their main training data is later enriched with junctions observed in GTEx data. As described in section 1.3.1.7, this might introduce novel splice sites and disrupt the mapping between input and output data.

Instead, the *collapsed* dataset from this work uses all exon annotations of different isoforms and compiles them together. To not introduce bias of annotation software, the pipeline filters to only those transcripts that were annotated by a human (HAVANA annotations): GENCODE exons are labelled with confidence levels (GENCODE, 2020); level 3 means that the exon was annotated automatically and was therefore filtered out. The remaining splice sites were compiled into one pseudo isoform that does not exist biologically due to duplicate acceptor or donor sites next to each other, but rather indicates all potential sites. This selection is expected to teach the algorithm to recognise splice sites independent from their isoform. The compiled gene for *SAMD11* is shown in Figure 3.5. Note that there are ambivalent regions that code for both exons and introns, as depicted by dotted lines.

The dataset described will be referred to as *gencode.v33grch38.collapsed*.

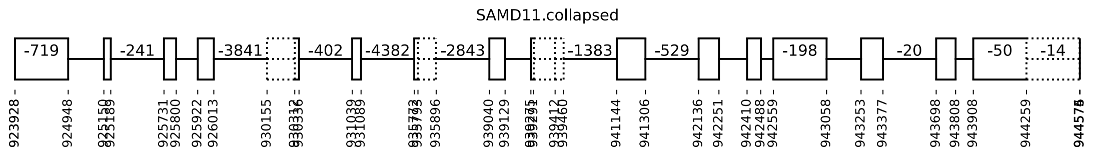


FIGURE 3.5: **Zoomed in version of the collapsed isoform of *SAMD11*.** It includes all splice sites from validated isoforms. Dotted lines indicate ambivalent regions where multiple donors or acceptors follow another. For a better visualisation, intron/exon regions longer than 300bps were shortened as indicated ($-N$, where N is the number of nucleotides removed).

3.3.3 Data Subsets for Convergence

The *subsetN.D* dataset takes the first (i.e. smallest genetic coordinates) N non-paralog genes from each chromosome from the dataset D. For example, *subset2.gencode.v33-grch38.collapsed* will take the first 2 non-paralog genes within each genome from *gencode.v33grch38.collapsed*.

Subsets will be used when base classifiers do not converge within 60 hours.

3.3.4 Limiting Data Leakage through Cross-Validation Partitioning

Jaganathan et al. partitioned their data into train and test chromosome-wise. The chromosome-specific split was presumably introduced to make testing harder by excluding paralogous genes or gene families with similar function and splice motifs and therefore prevent data leakage from the training to testing partition. Their code uses chromosomes 1, 3, 5, 7 and 9 for testing, and the remainder for training. For algorithms that converge faster than deep learning, a better approach would be to use a CV partitioning table.

Because human chromosomes differ significantly in size, this table was curated manually so that the proportion of train to test data was maintained. The partitioning is shown in table 3.1.

Partition	Chromosomes in Test Set	Test Set Size		
		Jaganathan et al.	GENCODE v33GRCh38 primary	GENCODE v33GRCh38 collapsed
1	1, 3, 5, 7, 9 *(original test partition)*	29.23%	29.54%	29.13%
2	2, 4, 6, 8, 13, 14, 20, 22	29.44%	29.27%	29.18%
3	10, 11, 15, 17, 19, 21, X, Y	29.82%	29.73%	30.00%
4	1, 2, 12, 16, 18	29.53%	29.49%	29.35%

TABLE 3.1: **Cross-Validation Partitioning Table.** The testing set within all four folds and all three datasets is between 29% and 30%.

With this 4-fold partitioning table, the models developed can be tested using a K-Fold method. Note however that the CNN is still only evaluated on the first partition due to the long time required to train it. This could be a future extension.

3.3.5 Position-Weight Matrix Analysis of Splice Motifs

PWM is used to visualise frequencies in the dataset and compare them to literature and each other.

From the datasets *gencode.v33grch38.primary* and *gencode.v33grch38.collapsed*, acceptor and donor sites are extracted with a window size of 20 nucleotides around the annotated site. The extracted and aligned DNA slices are then converted to a PWM and their logo representation is drawn.

PWM and logo implementations are based on a github code base (Azofeifa, 2017) which was adapted. The original implementation draws nucleotide letters manually using *matplotlib*'s plot tools, which doesn't scale to very long sequences of DNA.

Instead, the logo drawing was implemented by using a system font. Unfortunately, standard system fonts have letters of varying height and offset. Since both letter height and offset are crucial to PWM visualisations, each nucleotide letter had to be calibrated by converting them to their vector graphic representation, measuring their actual size and offset, and transforming them to the desired values.

Furthermore, the code was adapted to also depict the unknown nucleotide N in grey, and to sort all letters by weight so that the most frequent letter sits on top of the stack.

3.3.6 Plot Analysis of Acceptors, Donors, and Neither

Nucleotide windows around each splice site were extracted from *gencode.v33grch38.primary* and *gencode.v33grch38.collapsed*, following the pipeline for baseline classifiers (Figure 3.3).

Different nucleotide windows of size $N \in \{1, 2, 5, 20, 40\}$ around each annotated splice site were extracted, forming sequences of length $2N + 1$. These sequences were plotted using incremental PCA of batch size 200 in two dimensional space.

3.3.7 Clustering Analysis of Acceptor and Donor Sites

To help explore the complexity of distinguishing acceptors from donors, a clustering algorithm to distinguish between them is evaluated. Since these two data distributions seem to overlap to a certain degree, some sites are expected to be mis-clustered. These mis-clustered sites can be interesting since they would represent splice sites that do not follow most obvious patterns and might give new insights into the underlying biology. During development, this also helped find and correct for mistakes in the data processing pipeline.

The same data set introduced in the plot analysis (section 3.3.6) was used, with one exception: The plot analysis reveals that the *neither* class overlaps heavily with the other two, and is therefore expected to not cluster at all, and was removed.

Again, all window sizes, $N \in \{1, 2, 5, 20, 40\}$, were used and the N with fewest mis-clustered data points was selected.

All clustering algorithms described in section 1.3.2.1, namely K-Means and Gaussian mixtures with all *Covariance Types* (CVTs), tied, diagonal, and spherical, were evaluated using *scikit-learn*. The results for the clustering algorithm with the fewest mis-clustered sites per dataset and window size N are reported; the best configuration found is then further analysed by PCA visualisation, manual exploration, and per-class PWM analysis.

3.3.8 Comparison to Original SpliceAI Dataset

5 CNNs (ensemble classification) per dataset are trained and tested as described in section 2.2.4.2, one ensemble is trained and tested on the original primary GENCODE data set from Jaganathan et al., the other is trained and tested on *gencode.v33grch38.primary*. If both test scores are comparable to each other, trust in the new data pipeline can be further strengthened.

3.3.9 Training of Baseline Classifiers and Comparison to Deep Learning

Baseline classifiers, namely SVCs, Logits, RFs, and MLPCs are optimised and trained. This allows a comparison between different algorithms and allows future investigation of these algorithms and their workings.

Hyper parameters for baseline classifiers are found on *gencode.v33grch38.primary* using a 3-fold validation (for implementation details, see section 2.2.2). Hyper parameters found and their intermediary performance measures are reported. SVCs and Logit are optimised on *subset2* (see section 3.3.3) due to them scaling exponentially with the number of training points. The remainder of models is optimised on the complete corpus.

The best model configurations are then evaluated using CV on both *gencode.v33grch38.primary* and *gencode.v33grch38.collapsed* as described in section 2.2.4.1, and their final performance scores are reported.

3.3.10 Explaining Splice Site Classifications

The best baseline algorithm found and the SpliceAI models trained on *gencode.v33grch38.primary* were analysed using RFE and Integrated Gradients respectively (see section 2.2.5 on page 28).

The best baseline algorithm found was the MLPC followed by the RF500. As MLPCs cannot be probed using *scikit-learn*, the RF was investigated using RFE. Conducting an RFE analysis on RF500 did not complete within 24 hours, the number of trees was therefore reduced to 100. The RFE search was configured to remove the four least important features at a time on *gencode.v33grch38.primary* until four or less features are found. The reverse path of feature elimination represents the ranking of features and the top 25% of features are plotted.

SpliceAI feature contributions were calculated using Integrated Gradients and plotted as described in section 2.2.5. Two sites, an acceptor at chr1:930155 and a donor at chr1:9935492, were chosen to be representative for this report.

3.4 Results

3.4.1 Splice Site Dataset Comparisons

The reimplementation of the primary transcript dataset on GENCODE version 33 using GRCh38 comes close to the original, however it is not exactly the same. The overlap of selected genes is 97%. Out of the intersection of selected genes, paralog annotations are equal in 98% and transcript start and stop annotations match in 90%. CNN classifiers use all non-splice sites on the genes, baseline classifiers use ten times more non-splice sites than splice sites.

Identifier	No. Genes	No. Splice Sites	Proportion Acceptor/Donor
GENCODE primary (Jaganathan et al.)	18,677	369,660	Equal
gencode.v33grch38.primary	18,559	373,420	Equal
subset2.gencode.v33grch38.primary	46	766	Equal
subset15.gencode.v33grch38.primary	345	6,328	Equal
GENCODE+GTEx (Jaganathan et al.)	18,678	556,174	2% more acceptors
gencode.v33grch38.collapsed	18,563	423,182	3% more donors

TABLE 3.2: **Numeric data description of data sets, measured on all chromosomes.**

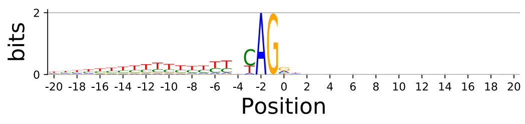
Table 3.2 shows the numeric description of the original SpliceAI data sets and all new data sets created. The number of genes and the number of primary splice sites are similar between the original SpliceAI datasets and the ones produced. The number of collapsed splice sites created in this work is smaller than the combined GENCODE+GTEx dataset, which is due to the addition of novel splice sites in Jaganathan et al.. While the newly produced collapsed dataset has slightly more donor than acceptor sites, the SpliceAI dataset is slightly skewed towards acceptors. The primary splice sites are all split equally between acceptors and donors, as would be expected for a collection of single transcripts.

3.4.2 Consensus Site Analysis between Primary and Collapsed Dataset

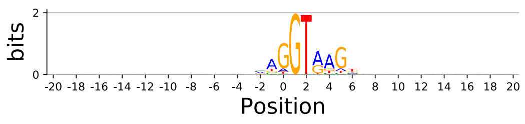
Consensus site frequencies of the PPM are listed in table 3.3. The vast majority of both datasets consists of the canonical splicing motifs AG and GT, with almost no difference between the primary and collapsed collection. Compared to the primary dataset, collapsed data points have slightly more canonical motifs in acceptor sites and slightly fewer canonical donor sites, indicating more alternative splicing affecting the donor end of a splice.

Dataset	Acceptor		Donor	
	A	G	G	T
gencode.v33grch38.primary	99.94%	99.85%	99.82%	99.14%
gencode.v33grch38.collapsed	99.95%	99.86%	99.86%	98.70%

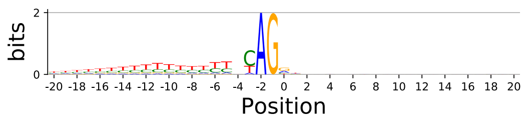
TABLE 3.3: **Nucleotide frequencies in consensus locations.** Not all splice sites contain consensus sites, but almost all do. The two datasets do not differ substantially.



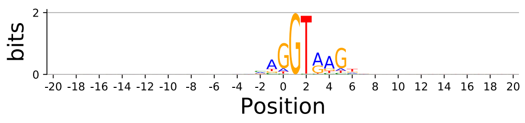
(A) Acceptor sites on *gencode.v33grch38.primary*



(B) Donor sites on *gencode.v33grch38.primary*



(C) Acceptor sites on *gencode.v33grch38.collapsed*



(D) Donor sites on *gencode.v33grch38.collapsed*

FIGURE 3.6: **Acceptor and donor logo comparison between primary and collapsed transcripts.** Consensus motifs AG and GT stand out, and intronic regions right before the acceptor have a higher density of C/T nucleotides (polypyrimidine tract). The PWM visualisations do not change much when including weaker splice sites, there are barely any differences visible.

Figure 3.6 shows acceptor and donor logos for both datasets, one containing only primary sites, and one containing all validated isoforms. There is no obvious difference in the splicing motifs between the two datasets. Both datasets show the polypyrimidine tract preceding acceptor sites and the canonical splicing motifs. There is a gap at -4 in acceptor sites - the frequency of this nucleotide appears random from an Information Theory perspective.

3.4.3 Splice Site PCA Plot

Naturally, very small nucleotide windows produced little variety, and the bigger the window, the bigger the distribution spread. For $N = 1$, no obvious clusters were visible. For $N \in \{2, 5, 20\}$, the distributions grow and acceptor and donor clusters become more distinguishable while the *neither* cluster overlaps with both heavily. For $N > 20$, the distributions overlap more and more; bigger values than 40 make the two distributions anneal and overlap more due to the increased diversity in sequences.

It is hard to decide which N is "best", because there clearly is not one correct answer. For consistency with other experiments, Figure 3.7 shows $N = 40$ for *gencode.v33grch-38.primary* and Figure 3.8 shows $N = 40$ for *gencode.v33grch38.collapsed*. We can see that the acceptor and donor distributions overlap a bit, but they do look like two clusters; a few data points overlap substantially between the two. The sampled non-splice sites are distributed circularly and overlap heavily with the acceptor and donor clusters. This might be 1) because the window size of 81 nucleotides is quite small, or 2) because the ground truth has missing or faulty annotations, or 3) because of other cis-factors, or 4) because of shortcomings of PCA, or 5) because binding patterns can be anywhere around a splice site and are not bound to a specific offset, which can't be modelled in this experiment. Most likely all reasons apply. Again, there is no obvious difference between the distributions for primary and collapsed isoforms.

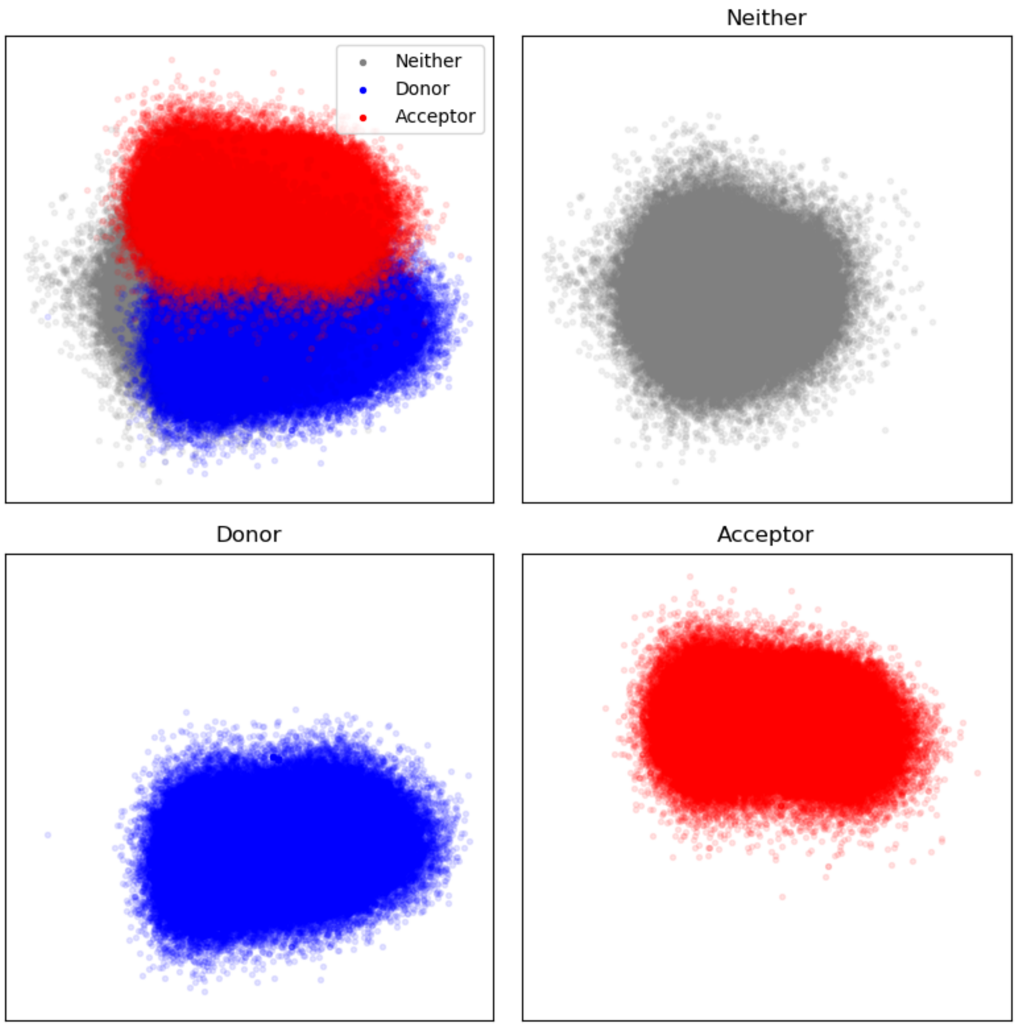


FIGURE 3.7: **PCA visualisation of the *gencode.v33grch38.primary* dataset.** 81 nucleotide windows (40 left and 40 right from the last nucleotide of an exon). The clusters overlap, but there are obvious acceptor and donor clusters. The *neither* partition overlaps with the other two quite drastically.

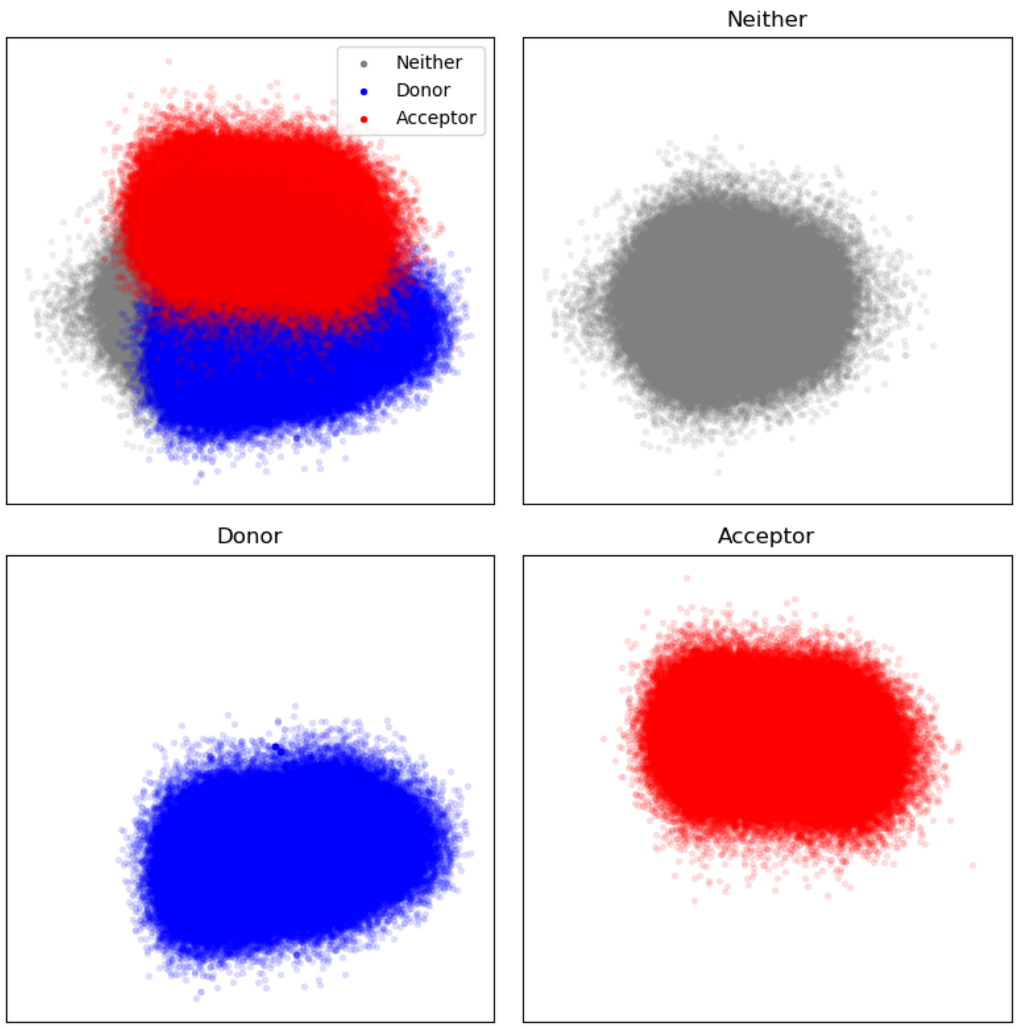


FIGURE 3.8: **PCA visualisation of the *gencode.v33grch38.collapsed* dataset.** 81 nucleotide windows (40 left and 40 right from the last nucleotide of an exon). It looks almost identical to the primary transcripts depicted in Figure 3.7.

3.4.4 Clustering of Acceptors and Donors

Table 3.4 shows the best clustering algorithm per window size N and how many sites were mis-clustered each. None of the clustering algorithms investigated performed consistently better than the rest. As expected, windows of $N < 5$ are too small and return a high number of mis-clustered points. For both datasets, the best result was achieved using $N = 20$ with only a small fraction of sites mis-clustered. Application to windows smaller than 40 nucleotides increases the number of mis-cluster donor sites significantly; $N = 40$ has the best trade-off between mis-clustered acceptor and donor sites of 50-55%. The percentages of mis-clustered sites on *gencode.v33grch38.primary* are lower than on *gencode.v33grch38.collapsed*, indicating that weaker isoforms are harder to classify. In all mis-clustered partitions, the consensus motif of the correct class is the strongest pattern except for in one case: Incorrectly clustered donor sites on *gencode.v33grch38.collapsed* have a stronger AG than the consensus GT motif. This partition is therefore a good candidate for future research.

N	Best algorithm	Mis-clustered sites			
		Total number	% of all splice sites	No. acceptors	% of mis-clustered acceptors/donors
		gencode.v33grch38.primary			
1	K-Means	21,073	5.64%	65	0.31%
2	Gaussian[CVT=diag]	7,454	2.00%	119	1.60%
5	K-Means	1,587	0.42%	650	40.96%
20	Gaussian[CVT=spherical]	834	0.22%	341	40.89%
40	K-Means	929	0.25%	472	50.81%
		gencode.v33grch38.collapsed			
1	K-Means	25,519	6.03%	67	0.26%
2	Gaussian[CVT=full]	8,710	2.06%	72	0.83%
5	Gaussian[CVT=diag]	2,007	0.47%	710	35.38%
20	Gaussian[CVT=full]	1,029	0.24%	212	20.60%
40	K-Means	1,453	0.34%	798	54.92%

TABLE 3.4: **Clustering results per data set and window size N.**

Figures 3.9 and 3.10 show the clustering result on a PCA plot for the two datasets and the window size of $N = 40$. The vast majority of data points were clustered correctly, and all mis-clustered points are on the border of the two distributions, which establishes that clustering and PCA returns sensible results, and that again the wider context is important to distinguish the two types of splice sites.

On *gencode.v33grch38.collapsed* (Figure 3.10), mis-clustered points spread more on the Y axis and overlap more with the acceptor and donor cluster, than for the primary dataset. This indicates that some non-primary sites are harder to classify.

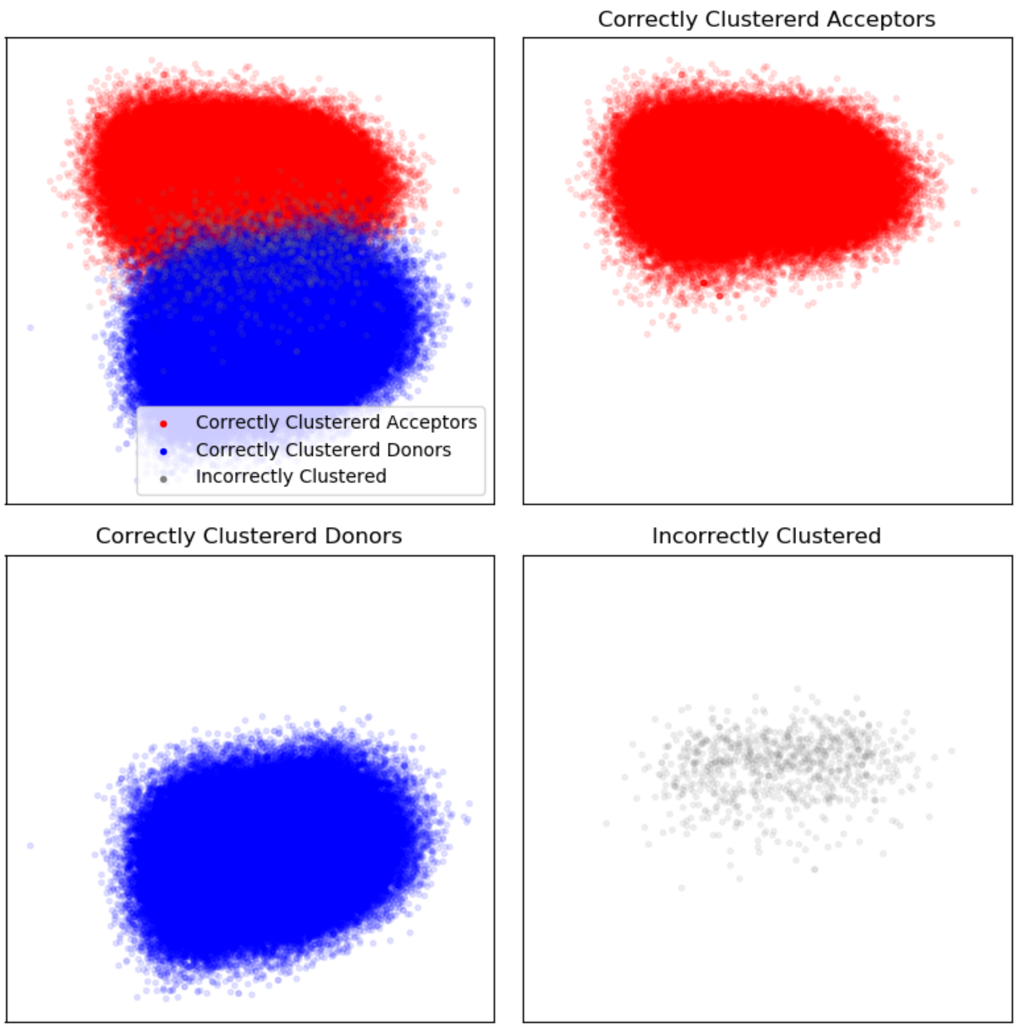


FIGURE 3.9: **PCA visualisation of the cluster analysis results on *gencode.v33grch-38.primary* dataset.** 41 nucleotide windows (20 left and 20 right from the last nucleotide of an exon). Mis-clustered data points are in between the two distributions.

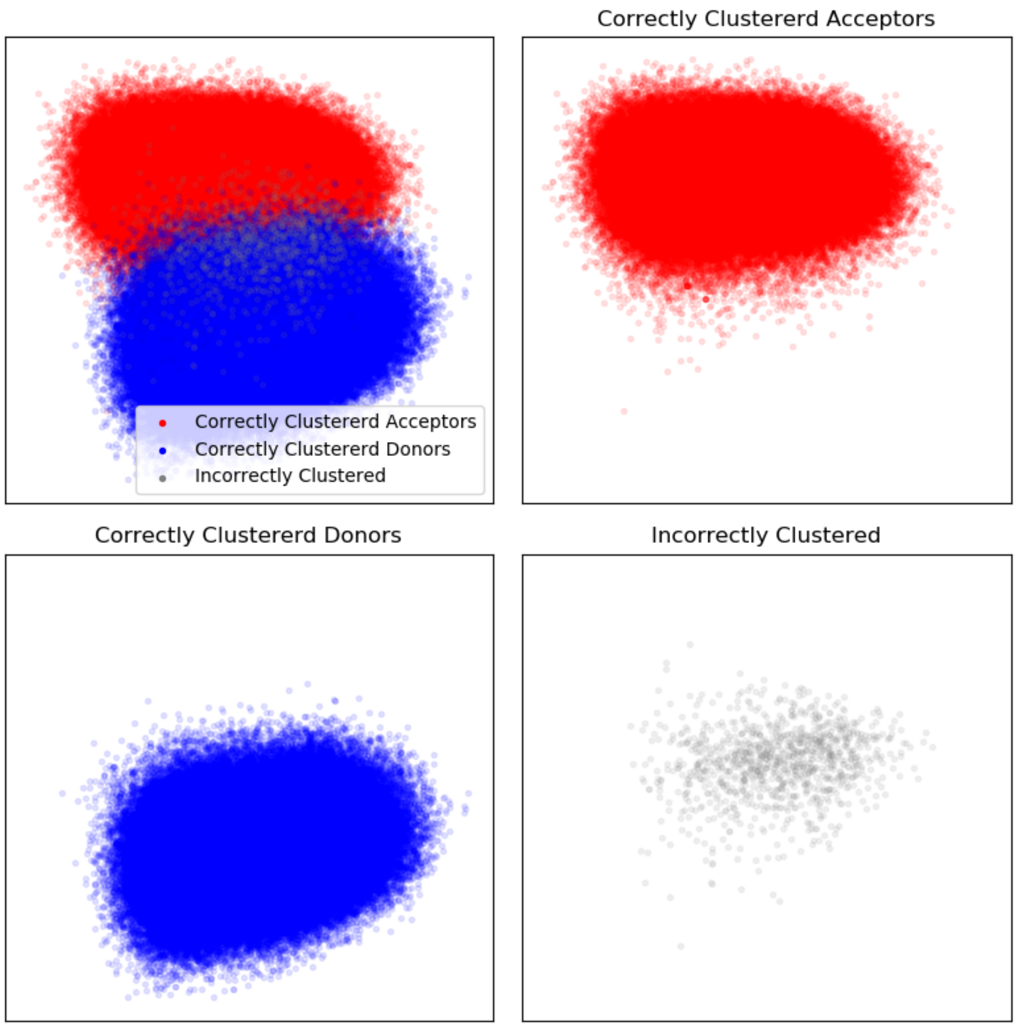
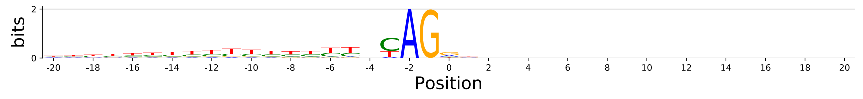
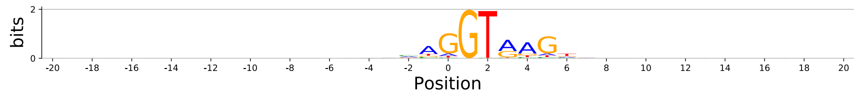


FIGURE 3.10: **PCA visualisation of the cluster analysis results on *gencode.v33grch-38.collapsed* dataset.** 41 nucleotide windows (20 left and 20 right from the last nucleotide of an exon). Compared to the primary transcripts, Figure 3.9, mis-clustered points spread more on the Y axis.

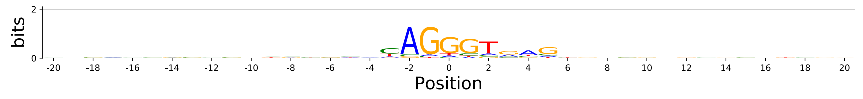
Correctly and incorrectly clustered data points were visualised as PWM logos in Figure 3.11. The PWM motifs of successfully clustered sites include clear consensus motifs (AG or GT for acceptors and donors respectively). Wrongly clustered acceptor sites always have a stronger AG than GT motif which shows that the algorithm does not only use consensus sites for its classification. The same upholds for mis-clustered donor sites on the primary set where the canonical GT motif is observed more frequently than AG. The exception to this are incorrect clustered donors belonging to the collapsed representation, where the AG is stronger than the GT pattern. Comparing primary to validated isoforms, the motifs for correctly clustered splice sites are the same. Incorrectly clustered PWM signatures are different, indicating that non-primary isoforms do differ in their signatures.



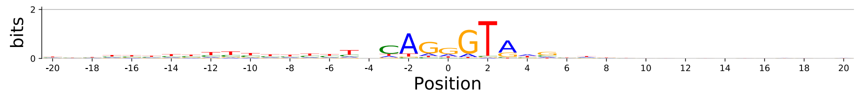
(A) Correctly clustered acceptor sites on *gencode.v33grch38.primary*



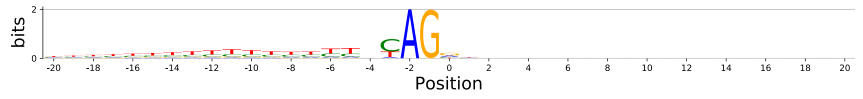
(B) Correctly clustered donor sites on *gencode.v33grch38.primary*



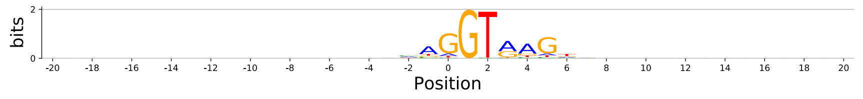
(C) Incorrectly clustered acceptor sites on *gencode.v33grch38.primary*



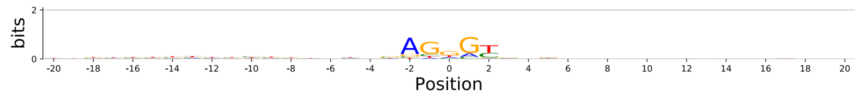
(D) Incorrectly clustered donor sites on *gencode.v33grch38.primary*



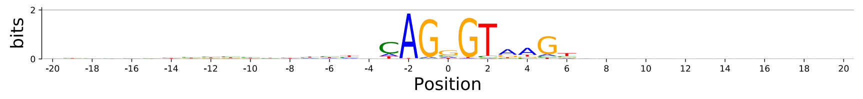
(E) Correctly clustered acceptor sites on *gencode.v33grch38.collapsed*



(F) Correctly clustered donor sites on *gencode.v33grch38.collapsed*



(G) Incorrectly clustered acceptor sites on *gencode.v33grch38.collapsed*



(H) Incorrectly clustered donor sites on *gencode.v33grch38.collapsed*

FIGURE 3.11: **PWM visualisations for clustered splice sites using N = 20 nucleotide windows left and right of a splice site.** Correctly clustered sites have very dominant consensus sites, incorrectly clustered sites show both acceptor and donor patterns. (h) Incorrectly clustered donor sites on *gencode.v33grch38.collapsed* have a stronger AG pattern than GT.

3.4.5 Comparison of Supervised Splice Site Recognition

Table 3.5 compares CNN test scores on the original and new dataset. The measures on the three primary datasets are very similar to each other, and my measures match the original published SpliceAI scores. This indicates that both the technical re-implementation of the SpliceAI training and the primary data set has worked. The two datasets that contain weaker splice sites at the bottom show both a drop in performance score, with the GENCODE+GTEx data being considerably harder to predict than the novel collapsed dataset.

Dataset	Measured in	Test avg.prec.
GENCODE (Jaganathan et al., 2019)	This work	97.58%
GENCODE (Jaganathan et al., 2019)	Jaganathan et al. (2019)	98%
gencode.v33grch38.primary	This work	97.88%
GENCODE+GTEx (Jaganathan et al., 2019)	Riepe and Jaganathan (2022)	87.70%
gencode.v33grch38.collapsed	This work	94.18%

TABLE 3.5: **Comparison of trained CNNs on different datasets.**

The best hyper parameters and their respective average precision on the 3-fold validation of *gencode.v33grch38.primary* are shown in table 3.6. The hyper parameters are documented for reproduction purposes, their corresponding AUC-PR is an intermediary result due to the 3-fold validation not respecting chromosomes and paralogs, therefore enabling data leakage between the train and test partitions. The best algorithms are MLPC and RF20.

Algorithm	Best Parameters				AUC-PR
SVC-linear	$b = true$	$C = 0.0001$			93.53%
SVC-RBF	$b = true$	$C = 1$	$\gamma = 0.001$		93.81%
SVC-poly	$b = true$	$C = 1$	$\gamma = 1$	$d = 3$	89.43%
Logit	$l = l_2$	$C = 0.001$			92.82%
RF20	$d_m = 20$				95.86%
MLPC	$L = (50,)$				98.05%

TABLE 3.6: **Hyper parameters found for baseline algorithms on *gencode.v33grch-38.primary*.** AUC-PR is an intermediate result.

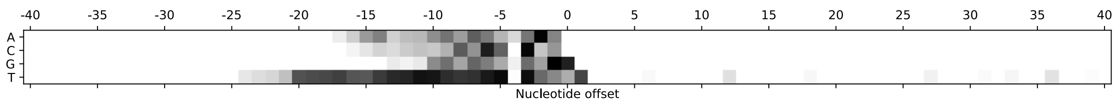
The best hyper parameters found were used to evaluate all algorithms on the actual CV table, see table 3.7. None of the SVC algorithms completed even the first CV partition within 24 hours; they were therefore tested on *subset2.gencode.v33grch38.primary* and *subset2.gencode.v33grch38.collapsed* respectively. All baseline models are reasonably stable (i.e. low standard deviation), the most stable models are the MLPC and RF500, which are also the best base models overall (95-98% AUC-PR). Across all algorithms, the collapsed representation is consistently harder to classify than the primary dataset and donors are easier to recognise across both datasets.

	AUC-PR (%)		
Model	Mean	Acceptor	Donor
	gencode.v33grch38.primary		
SVC-RBF	94.076 ± 1.539	92.553 ± 0.269	95.600 ± 0.165
Logit	93.180 ± 1.731	91.459 ± 0.136	94.901 ± 0.223
RF20	95.774 ± 1.394	94.391 ± 0.251	97.156 ± 0.044
RF500	96.738 ± 0.959	95.787 ± 0.160	97.689 ± 0.053
MLPC	98.032 ± 0.480	97.557 ± 0.097	98.506 ± 0.011
	gencode.v33grch38.collapsed		
SVC-RBF	93.358 ± 1.447	92.383 ± 1.190	94.333 ± 0.934
Logit	92.272 ± 1.971	90.322 ± 0.373	94.223 ± 0.147
RF20	94.983 ± 1.635	93.354 ± 0.107	96.613 ± 0.144
RF500	96.018 ± 1.225	94.794 ± 0.025	97.242 ± 0.081
MLPC	97.373 ± 0.690	96.685 ± 0.070	98.060 ± 0.048

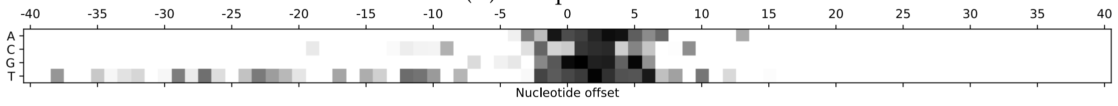
TABLE 3.7: **Cross-Validated AUC-PRs.** SVCs were evaluated on the respective *subset2*.

3.4.6 RF Feature Contributions

Figure 3.12 visualises the top 25% most important features for acceptor and donor sites. Many motifs found in the PWM analysis can be rediscovered in the RFE features: The T trail prior to acceptors, the gap at -4 preceding acceptors, the canonical AG and GT motifs, and their surrounding patterns. The most important motifs are at the immediate exon-intron boundary and its surroundings. Nucleotides in the exon at the 3′ site are less relevant than on 5′. However not all important features match what we know exist biologically: There are some relatively important T nucleotides on the exon preceding donor sites that we know do not exist in sequencing data. A possible explanation is that the absence of higher concentrations of Ts is important for classifying donor sites. This is a shortcoming of the feature estimation process; it reminds us that no direct transfer from RFE importance to biological features is possible. This discrepancy makes it hard to reach new conclusions about the underlying biology based on RFE analysis.



(A) Acceptor sites

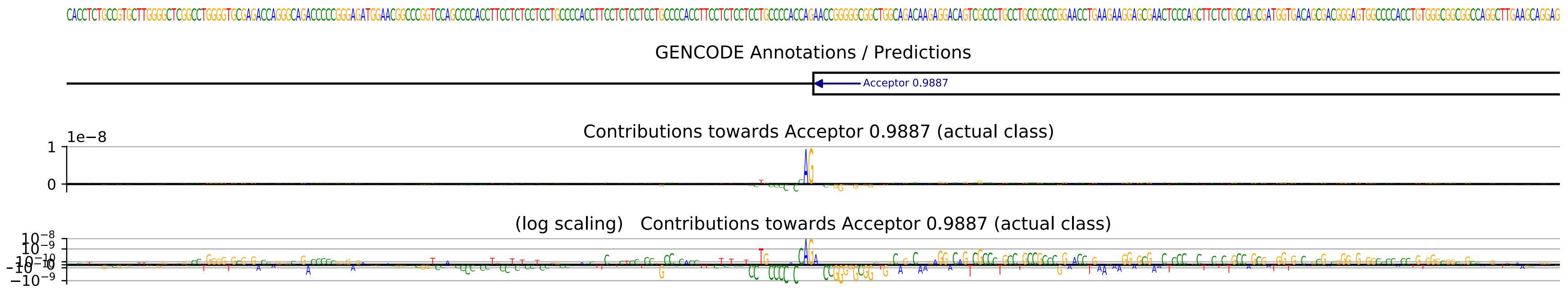


(B) Donor sites

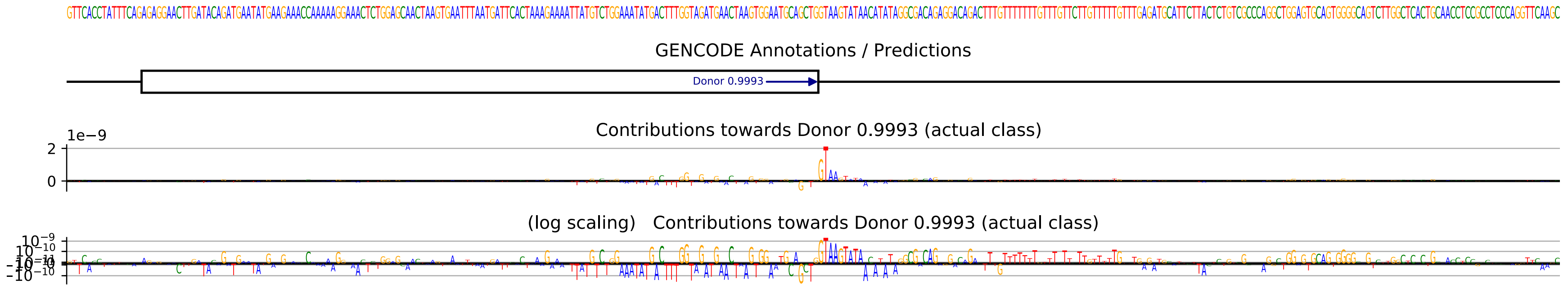
FIGURE 3.12: **Top feature contributions found by RF100 classifying splice sites.** The darker, the higher the importance of a feature. All ranks below 25% were cut off (white). (a) Prior to acceptor sites, there is a T/C trail prior to the sites, and a gap on offset -4. Most nucleotides on the exon are unimportant. A clear (C/T)-AG-GT pattern is found at the consensus site. (b) Donor sites have a strong AG-GT motif at the consensus site. Features are found in both exon and intron. There is a spaced out trail of important Ts preceding donor sites.

3.4.7 SpliceAI Feature Contributions

Figure 3.13 shows feature contributions towards an acceptor and a donor site. Due the immense input size (5,000 flanking nucleotides on either side), only the 600 nucleotides in the middle are shown. The results are very granular and subjective to the actual data point investigated. As expected, canonical motifs are the most significant positive contributions towards the classification results, followed by the immediately surrounding nucleotides. Nucleotides further away have less of an impact on the classification, but feature contributions span over the whole input sequence. C and G nucleotides are generally preferred in the end of an exon and opposing acceptor sides in the beginning of an exon. As and Ts at the end of an exon oppose the donor classification result.



(A) **Acceptor site at chr1:930155.**



(B) **Donor site at chr1:9935492.**

FIGURE 3.13: **Integrated Gradients visualising deep learning feature contributions.** Only 600 out of 10,001 input positions shown. Positive contributions are in favour of the classification class, negative contributions are opposing this result. (a) The CAGA motif at the consensus site is the biggest contributing factor, and the immediate surrounding Cs in the intron and Gs in the exon are not in favour of an acceptor class. There is a TG motif at -9, and Cs and Gs deeper in the exon generally strengthen the acceptor classification. (b) The intronic GTAAGTATA sequence is a clear contributor towards this splice site. Exonic A and T nucleotides are considered opposing this splice, where Gs and Cs in the exon are contributing factors.

3.5 Discussion

3.5.1 Differences to SpliceAI GENCODE Dataset

As described in section 3.4.1, the primary dataset from McRae et al. (2019a) could not be reproduced completely. The following shortcomings of the data sources were found to relate to this mismatch:

3.5.1.1 GENCODE Updates, Lifting, and Naming Issues

Between the two genome builds called GRCh37 and GRCh38, gene coordinates changed completely. This means that when GENCODE releases an update (new gene annotations), they create their database on the recent 38 build and *lift* (re-map) the data back to GRCh37, allowing them to release a lifted legacy build as well. This however only lifts coordinates, if gene symbols change, they are not mapped to their legacy names. Updates of GENCODE annotations are released in *versions*, the latest version being v33 at the project start[4]. Jaganathan et al. used the GRCh37 build for v24.

Duplicate gene symbols with unique Ensembl IDs per chromosome were found, i.e. on GENCODE v24lift37, the gene *PIK3R3* has two unique IDs (ENSG00000278139 and ENSG00000117461), so does *TMEM236* (ENSG00000148483 and ENSG00000184040) and 54 others. The data set of Jaganathan et al. uses gene symbols as unique identifiers, therefore duplicates have to be filtered out, which is questionable. The number of naming collisions reduce with both newer GENCODE versions and by migrating to GRCh38, see table 3.8 on page 59.

A similar problem was observed on transcript names and IDs. There are some duplicate transcript names with unique gene IDs, i.e. the gene *XAGE2*: GENCODEv24 has two transcripts for the gene *XAGE2*, one on each strand, and they share the same transcript name.

This implementation uses Ensembl IDs rather than names in order to circumvent these issues.

Another problem found on GRCh37 are *tiny introns*. For example, after filtering to protein coding transcripts and level < 3, the gene *ZNF280A* has one transcript called ENST00000302097.3, which exists on both builds. On GRCh38, this transcript consists of two exons, on GRCh37 it has ten. On GRCh37 they are annotated with *remap_status=partial*; they are separated by very small introns and make little sense. 1,455 protein-coding exons are marked with this flag.

[4] At the time of writing, the latest version is v37. Future experimentation will use an updated version, however so far all experimentation is based on v33.

3.5.1.2 Mismatches between Ensembl and GENCODE

GENCODE has archived all legacy versions to download, allowing recreation of older data pipelines, however the Ensembl website providing paralogs has only the newest version for build 37 and 38, and it is unclear how their annotations relate to GENCODE versions. This is an issue as it was already shown that gene symbols may change between GENCODE versions. More confusingly, this is inconsistent between GENCODE and Ensembl: Using the Ensembl database, it was found that the gene symbol of ENSG00000142920 changed somewhere in the past from *ADC* to *AZIN2*, and both names are used in Ensembl build 37 and 38 respectively. GENCODE on the other hand refers to it as *AZIN2* in both builds, because their lifting algorithm only translates coordinates and not names. Jaganathan et al. annotated the gene *AZIN2* as 'no paralog', but both Ensembl on build 37 and 38 claim ENSG00000142920 is. *AZIN2* does not exist on Ensembl 37, which would indicate that they fill mismatches between GENCODE and Ensembl with *'no paralog'*. To not fall into the same trap, this implementation again uses the Ensembl ID rather than its name to join the two sources. Another sensible explanation is that Jaganathan et al. are not working with the most recent Ensembl data set, but instead with an outdated list of paralogs. The exact issue is hard to reconstruct without the code of their data pipeline or a history of Ensembl annotations.

Mismatched paralog annotations between Ensembl and GENCODE were counted. Table 3.8 shows that there is still a considerable amount of these mismatches, and that the number goes down when using more recent GENCODE versions or build 38. The 18 mismatches on v33 build 38 are all on chromosome Y.

GENCODE (Genome version)	Name Duplicates within GENCODE, on protein coding transcripts	Ensembl (Genome version)	Mismatches between GENCODE and Ensembl, on primary protein coding transcripts
v24lift37(p5)	50	GRCh37(p13)	328
v24(p5)	86	GRCh38(p13)	269
v33lift37(p13)	34	GRCh37(p13)	581
v33(p13)	30	GRCh38(p13)	18

TABLE 3.8: **Gene and name duplications respective to GENCODE versions and builds.** The number of gene symbol duplicates and missing Ensembl annotations for protein coding primary transcripts goes down when using GRCh38 and/or GENCODE version 33.

It was therefore decided that the last row is most fit as a dataset.

3.5.1.3 Selection of Transcripts

In their work, Jaganathan et al. document to have selected protein coding transcripts, but not their exact process. Filtering transcripts for associated protein IDs returns the exact same set of genes in the resulting splice table. However, it would also mean including annotations tagged as *transcript_type=nonsense_mediated_decay*, which are of poor quality and unlikely actual splice sites. The same problem would apply filtering to *gene_type=protein_coding*. A third variant is to filter transcripts for *transcript_type=protein_coding*. Some of these annotations are faulty, i.e. *TRBV6-4* has an associated protein but no transcripts tagged as *transcript_type=protein_coding*, but it was included in SpliceAI. I decided to commit to the latter method in order to exclude nonsense mediate decay data, trading off a 3% mismatch in the selection of genes.

In the same manner, Jaganathan et al. did not document how they select primary transcripts, and clarification requests remain unsolved to date (Erdem and Jaganathan, 2021). Different filtering was evaluated, and no perfect reimplementation was found. As described in section 3.3.1, the process selects the transcript with the lowest *appris_principal* tag, under the assumption that *appris_principal_1=appris_principal*, and breaks ties by selecting the transcripts with more exons. This comes close: Out of the 97% matching genes, transcript annotations are equal in 90%. This score obviously depends on the pre-selection of protein coding transcripts.

Instead of selecting for the *appris_principal* tag, selecting the MANE Select transcript would be a more modern and promising strategy. Pozo et al. (2022) evaluated these two against other popular selection strategies, such as taking the longest transcript, and have shown that in the context of splicing, *appris_principal* and MANE Select are superior. At the time of experimentation, MANE was still under development and could not be used; for future experimentation the MANE selection could improve the selection and reproduction issues observed in this work, and might improve data quality even further.

The data pipeline is based on the Human Reference Genome and two subsets of GENCODE annotations. As motivated in section 3.2.1, the Human Reference Genome is biased towards Western heritage and cannot include benign variants. This might hinder application, both regarding benign variants, and application to the general public of varying ethnicity. While the substitution of data to more diverse sequences will raise technical challenges due to the inflation of data, it might produce a more comprehensive model. It would also require much more care when evaluating performance as results would need stratification over different backgrounds, which, as demonstrated by gnomAD (Karczewski et al., 2020), is by no means an easily achieved task.

3.5.2 Splice Site Quality

The motifs found for primary splice sites closely resemble literature (i.e. Lord et al., 2019, Figure 1-2), indicating that the data pipeline extracted sensible splice sites. The consensus motifs AG and GT are found at almost 100% frequency (2bits), which is what is to be expected from literature (Burset et al., 2000).

Based on both the plot and clustering analysis, it seems that algorithms should be able to distinguish acceptors from donors quite well, but being less effective in separating splice sites from non-splice sites. This explains why supervised classification is preferred to unsupervised clustering in the wider are of splice site classification.

A subset of mis-clustered sites was inspected manually to assure data quality. Some splice sites are not annotated on UCSC (Karolchik et al., 2003) such as ENSG00000003756: 50092043, indicating the potential need for additional filtering to improve data quality. Some sites do not follow consensus patterns for the major spliceosome, which is expected to be mis-clustered due to the small number of sites. The vast majority of instances extracted seem to be real splice sites.

It must be emphasised that the results from the clustering process have to be taken with some reservations. The pipeline already filtered down to validated splice sites, which simplifies the problem of splice site detection drastically. Furthermore, the clustering algorithms, especially K-Means, are very simple and known to often result in unexpected and stochastic results.

3.5.3 ML Model Comparison

When rounding to the same precision as Jaganathan et al., the CNNs score on the original data set are equivalent to their publication. This shows that the train and test code was adapted to IRIDIS successfully. The performance on *gencode.v33grch38.primary* is similar; variances in performance are to be expected due to the data being slightly different and random stochasticity in the training process.

The baseline classifiers return unexpectedly high test scores compared to the CNNs. However, all of these scores need to be taken with a pinch of salt: One cannot compare base classification metrics with those measured on CNNs due to the difference in their train and test data. Baseline algorithms were only trained and evaluated on a subset of non-splice sites. The SVC results cannot even be compared to the rest of baseline classifiers due to it being trained on an even smaller dataset.

The collapsed representation seems to make the problem considerably harder as the performance for all algorithms dropped, which is somewhat surprising since the plot and cluster experimentations have found only minor differences between the two datasets.

The drop in predictive quality between primary and collapsed isoforms is unlikely a mistake as it was also observed for the GENCODE+GTEx data published with SpliceAI (Riepe and Jaganathan, 2022), where the average precision score dropped from 98% to 88%. The gap between their 88% and the 94% measured on the collapsed data is likely due to 1) the collapsed data only including splice sites previously annotated by a human, 2) their inclusion of *novel* splice sites, and 3) through the mismatching input and output data due to the Human Reference Genome being different to the genome of GTEx patients. At this stage it is however hardly possible to state which dataset is better for clinical use; this has to be analysed in a different experiment.

The comparison of feature contributions between shallow and deep learning reflects their difference in complexity. While feature contributions for the RF were global and didn't reveal much detail, the weights on the CNN input were local to one datapoint. Both levels of granularity made learning about the splicing process much harder. The log scaling of SpliceAI feature contributions however revealed that the whole input sequence affects the classification result, strengthening the thesis that bigger context sizes are important made by both Jaganathan et al. and mentioned in the preceding cluster analysis. Unfortunately, while positive and negative feature contributions are certainly insightful, there seems to be no obvious learning from this as results are too detailed and no apparent rules could be derived.

3.6 Conclusion

Despite many challenges during the reimplementation of the SpliceAI data due to missing documentation in the literature, it was shown that the new dataset was of at least comparable quality. The resulting *gencode.v33grch38.primary* dataset was updated to a more recent GENCODE version and build, and it was fit for usage for different ML algorithms. A *subset* mechanism allows fast creation of smaller data sets, and a CV table was introduced to facilitate testing of baseline algorithms.

In the context of rare genetic disease, classification of weaker splice sites is important. The novel *gencode.v33grch38.collapsed* dataset was created differently to the main training data of SpliceAI: It was filtered to include only transcripts that were annotated by a human, instead of joining them with novel splice sites from GTEx. The resulting dataset is easier to classify than SpliceAI's GENCODE+GTEx data, however that does not necessarily mean that it is better or worse in a clinical setting; this has to be explored in an independent experiment.

Motif analysis was used throughout to visualise the datasets as a whole as well as investigate mis-predictions, and it was shown that logo patterns are sensible and comparable to literature. It can therefore be concluded that both data sets curated are of good quality representing the problem domain. However some challenges remain:

Non-canonical patterns are naturally under-represented, the subset of negative sites will affect classification results, and some annotations are disputed between Ensembl and UCSC.

Motif and plot visualisations have shown that distinguishing donors from acceptors is generally easy, so easy in fact that simple unsupervised clustering methods can do so with very high accuracies. The challenge however is clustering negative samples versus rest: There is a big overlap between spliced sequences and those that aren't. Inclusion of weaker splice sites did not change this.

All ”top-down” analysis (i.e. excluding supervised learning) revealed no immediately obvious differences between the primary and collapsed dataset. This is because these experiments look at the data very superficially and do not reveal subtle patterns in data; neither do they look at the wider context of splicing. During supervised training, it was shown that weaker splice sites are actually harder to recognise, and the Integrated Gradients revealed feature contributions are indeed distributed across thousands of nucleotides; the context size therefore is very important and a ”top-down” look of the data is not an appropriate representation of the problem complexity.

The RFE analysis conducted revealed global feature contributions. Many known splicing motifs could be traced back to these features, however the inverse did not hold and no new insights into the workings could be generated. In contrast, Integrated Gradient analysis offered a very granular view into the feature contributions of a deep neural network, however that view was way too detailed for intuitive interpretation. If explaining simple cases with clear splicing motifs and significant predictions is hard to do, using this system to explain mis-classifications or even learn about the splicing process itself doesn't seem feasible.

Both CNNs and baseline classifiers returned impressive scores classifying splice sites. Baseline models were shown to be stable, indicating sufficient data for Machine Learning. However, these scores could not always be compared due to different ways of how their train and test datasets are created. This means no conclusion of what the best algorithm is can be reached; to do so, a more fair comparison is needed.

As all goals set in this chapter were made, the general objective 1 is considered resolved.

Two main questions are left unanswered: Whether the collapsed representation is, despite being harder to classify, more useful in a clinical setting, and if simple baseline algorithms can substitute the deep learning algorithms. This will be answered in the next chapter, where all algorithms trained on both datasets are evaluated on an independent dataset of genomic variants.

Chapter 4

Application of Machine Learning to Variant Data

4.1 Introduction

The previous chapter described how to train a set of supervised classifiers that are able to recognise splice sites in the Human Reference Genome with high test scores. Recognising splice sites is however of limited clinical use because most splice sites in the Human Reference Genome are already known (and required to train the algorithms to begin with). Two major questions remained unanswered: 1) Whether the collapsed representation that also includes weaker splice sites is better than the one containing primary isoforms, maybe even better than the GENCODE+GTEx data used to train SpliceAI, and 2) if simpler baseline classifiers can be a substitution to the deep CNN architecture of SpliceAI.

This chapter investigates the application of all trained supervised ML tools onto patient variant data to replicate functional analysis results, i.e. if variants affect splicing. This also allows a fair comparison between deep learning and simpler baseline classification, which was not possible before due to data constraints.

Finding an algorithm, no matter the training dataset or model architecture, that can detect splice disruptions more reliably than previously published tools would be a major achievement as it can be transferred towards an application usable for both clinical and research purposes.

Great care needs to be taken to balance the number of variants and their quality. While a larger number of variants helps in determining which algorithm is best, the quality of annotation is just as important to strengthen confidence in the results.

This chapter partially overlaps with chapter 6, which expands the dataset, methods, and analysis described here further. There are however important methodological differences described in section 6.1.1.

4.1.1 Aims and Objectives

Objective 2, applying supervised algorithms to annotate aberrant splicing in variant data, is to be achieved. To do so, patient data from previously published functional studies is collected and aggregated. Collected data needs to be normalised into a common format, and shared variant annotations between different sources need to be compared. Any conflicts need to be resolved.

A method of using the supervised ML models from chapter 3 to determine if variants affect splicing, especially for insertions and deletions, needs to be developed. All ML algorithms, deep and shallow, trained on the two datasets of primary and collapsed isoforms, are then to be evaluated on the variant dataset to allow a fair and independent comparison.

4.2 Background

4.2.1 Splice Variant Classification using ML

There are generally two approaches on variant classification using ML: Teaching the algorithm to detect variants, or teaching the algorithm to model splicing and investigate the difference between reference and variant predictions.

SQUIRLS (Danis et al., 2021) is a Machine Leaning tool consisting of two RFs (one for acceptors, one for donors) followed by a Logit, and is trained directly on splicing variants. Training on variants directly optimises the algorithm on its primary application. On the other hand, tools like MMSplice (Cheng et al., 2019) and SpliceAI (Jaganathan et al., 2019) have never seen a variant during training and therefore cannot overfit on any variant data. If tools that were not trained on variant data can classify splice disruptions well, they demonstrate that they at least partially understand the underlying splicing process. This design seems preferable to training on variant data itself.

To use a model trained on splice site recognition alone, Jaganathan et al. (2019) build the difference between predictions for canonical and alternative sequences called the δ-score. As their model classifies thousands of neighbouring nucleotides at once, both outputs and δ-scores are inherent matrices. Figure 4.1 illustrates this process exemplified on a synthetic variant datapoint.

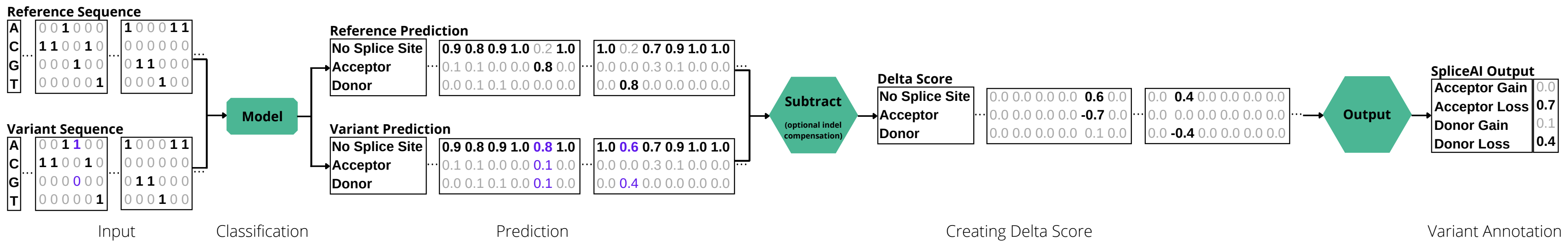


FIGURE 4.1: **Synthetic example of how to apply ML models recognising splice sites onto variant annotation.** The ML models output two predictive matrices, one for the reference and one for the variant sequence. Their difference builds the δ score. The classification output for SpliceAI extracts the highest and lowest values for acceptor and donor predictions. SpliceAI also returns the genomic location relative to the variant for each of these measures (not depicted). This example illustrates how a variant in an acceptor site disrupts the canonical AG motif and causes predictions to change for both the acceptor and donor site; the model variant annotation therefore represents an exon skip event.

In most cases the δ score is the direct subtraction of the two predictive matrices, with the exception of deletions and insertions where SpliceAI compensates by filling the variant sequence with zeros or truncating it with the max function respectively (McRae et al., 2019a). If the $|\delta|$-score is significant at any nucleotide, i.e. $\max(|\delta|) > \Delta$, the variant is determined to be splice affecting. For significance, the authors recommend a threshold of $\Delta = 0.2$, however other thresholds may be used to balance false positive and false negatives as desired. As the input sequence is of finite length, one has to choose the maximum distance to a variant. The SpliceAI paper advises to restrict analysis of the δ-score to a window around the variant; this threshold was fixed to 50 in their publication (Jaganathan et al., 2019) but since then was enlarged to 500 nucleotides (McRae et al., 2019a) to account for deeper variants.

4.2.2 Gene and Variant Annotation Formats

RefSeq, the system to describe genes and variants published by the NCBI (see section 3.2.1 on page 32) uses prefixed IDs to indicate the type of annotation, most notably (in the context of splicing) ”NC_” for chromosomes, ”NM_” for protein coding RNA, ”NP_” for protein and ”rs” for SNVs. Clinical variant IDs defined by RefSeq however are limited to the ones in their database and can therefore not be used for novel variants.

The predominant notation to describe generic variants is the one published by the *Human Genome Variation Society* (HGVS) (Cotton and Horaitis, 2001). These IDs are in the format ”Reference:Description”. ”Reference” can be any sequence ID (i.e. RefSeq or Ensembl ID). ”Description” contains the location within the sequence, the type of variant (substitutions, deletions, insertions, duplications, or deletion/insertions) and reference/variant sequences if appropriate. (Horaitis and Cotton, 2004)

Despite HGVS IDs allowing different sequence identifiers, for splicing the vast majority of literature report IDs that are based on RefSeq IDs describing protein coding RNA (prefixed with ”NM_”).

4.3 Methods

4.3.1 Aggregation of Splice Variant Data

The bigger the dataset, the more confidently we can identify which algorithms are appropriate, however the dataset quality is also very important. As a trade-off, all massive parallel splice essays were excluded; while they would provide a substantial corpus, they are often not representative of the clinical practice. It was decided to include both primary and secondary sources of splicing related variants, as long as their annotations of splice disruption have been functionally validated. I further limited my data sources to only include variants that are either already in the public domain, or were in the process of being published, so that the aggregated dataset can be published as well. To be usable, they should either annotate variant coordinates with a base-pair resolution, or use HGVS IDs.

The following sections describe how variant data was aggregated from six sources (Houdayer et al., 2012; Ito et al., 2017; Jian et al., 2014; Leman et al., 2018; Maddirevula et al., 2020; Wai et al., 2020). As described later, Jian et al. (2014) was excluded from the final dataset. The source data was parsed, genomic coordinates extracted or fetched, and data was aggregated into a common format.

During data analysis, multiple challenges were identified in the source material. Some sources reported HGVS IDs without genomic coordinates, some reported genomic coordinates without HGVS IDs. When genomic coordinates were provided, both GRCh37 and GRCh38 were used. For negatively stranded genes, some sources report reverse-complemented reference and variant annotations, some report them according to the forward strand. All these issues were rectified as described below.

To bring all data in a shared format, it was decided to follow convention and use HGVS IDs as primary identifiers. Since these are not unique, the same variant may be described with very distinct IDs; duplicates were therefore removed based on their genomic location on GRCh38.

The following general process was used:

1. Parse sources into structured data where needed and aggregate
2. Fill in missing HGVS IDs based on "NC_"-RefSeq descriptors, as described in section 4.3.1.3
3. Adjust HGVS IDs:
    (a) Replace non-ASCII characters with their ASCII equivalent
    (b) Remove line breaks
    (c) Remove leading and trailing whitespaces
4. Query GRCh38 locations for all variants
5. Reverse-complement incorrect annotations on negatively stranded genes where necessary
6. Investigate and manually resolve conflicts where sources disagree
7. Resolve duplicates based on genomic location

Furthermore, since many publications cite others, the citation chain was retained. This allows investigations into disagreeing sources (and their sources) in section 4.4.1.

4.3.1.1 Incorporating Wai et al.

This study (Wai et al., 2020) sequenced RNA from whole blood to assess splicing defects. Their table S1 consists of 258 (actually 259) variants across 65 genes.

There is a duplicate HGVS ID (variants 32/33), where apparently there was a copy-and-paste error. In the original publication, variant 32 was incorrectly called NM_007294.3:-c.5024C>T (duplicating the entry below) and with the authors help the ID was corrected to NM_007294.3:c.5074+7C>T. Variant 220 is really two; the authors could not determine which variant was causing the effect, so both variants were removed.

The splicing annotation from the source was changed to a binary form (”Normal” / everything else). After parsing the RefSeq IDs to genomic coordinates, 12 variant locations were found to be offset by 1bp, which was rectified.

4.3.1.2 Incorporating Maddirevula et al.

The publication Maddirevula et al. (2020) assesses splice disruptions by analysis of mostly blood RNA sequences. Some RNA expression was derived from cultured skin-derived fibroblasts and urine-derived renal epithelial cells. Their data, table S1, contains an aggregate of 272 (269 really since 3 were not disclosed) variants, 124 new ones, 50 previously published variants across 45 publications, and 98 without attribution.

The HGVS IDs needed to be manually healed extensively. Missing colons that are supposed to delimit transcripts from variants and mangled protein annotations of different kinds were healed manually. Furthermore, only data points where the RT-PCR outcome indicated a conclusive splicing disruption were included.

NM_001040656.1 was deprecated by NCBI and NM_001077416 is not supported by Ensembl *Variant Effect Predictor* (VEP), both variants were removed.

4.3.1.3 Incorporating Jian et al.

Table S2 from this purely computational study (Jian et al., 2014) aggregates 2,961 variants: 1,164 splice altering variants from DBASS (Buratti et al., 2010), *Human Gene Mutation Database* (HGMD) (Stenson et al., 2020), and SpliceDisease (Wang et al., 2012) and 1,795 healthy variants from the 1,000 Genomes Project phase 1 (Siva, 2008). The cited data sources are in turn aggregations of many different studies with vastly different experimental methods to assess splice predictions.

During validation of their reference annotations, 9 benign variants (originally from 1000G), where the reference genome was updated to reflect exactly that variant, were found. This illustrates how important it is to keep the reference genome up to date.

They report genomic coordinates, but no HGVS ID. To create a coherent data set, HGVS IDs were created: First their coordinates were lifted to GRCh38, then translated to HGVS IDs in the ”NC_” format (i.e. NC_000001:g.78429408G>C corresponds to chromosome 1, position 78429408, variant G>C). These ”NC_” IDs were parsed into ”NM_” IDs using Ensembl variant recoder (Ensembl, b). If this process fails, it was retried with GRCh37 IDs (using the original coordinate and the build 37 Ensembl service). This failed in 11 instances, in which the ”rs” NCBI ID was used.

4.3.1.4 Incorporating Leman et al.

Tables S1-S3 from Leman et al. (2018) were incorporated, containing a total of 254 variants across 67 publications (141 breast cancer variants of their own, the rest across 66 publications) across 11 genes. Splicing was assessed using minigene assays and RNA sequencing from lymphoblastoid cell lines, whole blood, and stimulated T lymphocytes.

NM_007294.3:c.133_136del is an invalid ID that could not be healed manually as it's unclear if this is a single nucleotide deletion or if it's removing a range of nucleotides. Transcript/Variant annotations were used to generate HGVS IDs, and the *Splicing_Effect* field was used as ground truth.

4.3.1.5 Incorporating Houdayer et al.

This study (Houdayer et al., 2012) assessed aberrant splicing using minigene assays and RNA sequencing of lymphoblastoid cell lines, blood, and stimulated T lymphocytes. Their publication includes 272 variants for *BRCA1* and *BRCA2* and partially overlaps with the Leman et al. dataset (in fact, Houdayer et al. worked on the Leman et al. paper too, and the Leman et al. paper accredits some variants to this paper).

65 of the variants are published as a HTML table and 207 variants on a PDF table across 17 pages. Annotations from HTML were extracted through copy-and-paste into Microsoft Excel (Microsoft Corporation), the PDF table was parsed using Tabula (2018), followed by manually healing of character recognition issues.

15 annotations where the outcome was not obvious were removed, only retaining entries tagged as acceptor/donor loss/gain / skipping / retention. One variant had no annotated observation (NM_000059.3:c.7056T>A), which was also removed. Some IDs

contained recurrence annotations in their ID, which were cleaned as they are syntactically invalid. Two variants (NM_000059.3:c.7397C>T, NM_007294.2:c.5074+68T>C) have mismatching reference annotations and were removed. NM_007294.3:c.5077_5080 del4ins10 has a missing insertion annotation and was removed.

When resolving duplicates, it was found that five variants that Leman et al. accredit this paper for, are not actually published by this paper.

4.3.1.6 Incorporating Ito et al.

Ito et al. (2017) published 57 *LMNA* variants in their table S5, 139 *MYBPC3* variants in their table S6, and another 43 and 31 (30 due one duplicate) *LMNA* and *MYBPC3* genes respectively in their table S7. Using splice assays in kidney cells, they compared normal and abnormal splicing reads and their statistical significance.

For *LMNA*, the RefSeq ID NM_170707.4 was used, *MYBPC3* was translated to NM_000256.3. Variants NM_170707.4:c.89C>A, NM_170707.4:c.95C>T, and NM_170707.4:n.890G>T have mismatching reference annotations, likely due to updates of the reference genome. These variants were not splice affecting and were removed.

The remaining 267 variants were extracted. Following their paper, variants with an annotated p-value < 0.01 were annotated as splice affecting; the remainder as non-affecting.

4.3.1.7 Extracting Genomic Coordinates

Except for the Jian et al. (2014) paper, GRCh38 coordinates for the HGVS IDs were fetched automatically using *Ensembl VEP* (Ensembl, a). This web service accepts batches of input, however if there is an invalid or unknown ID, the complete batch is rejected without indication which one caused the error.

IDs were therefore processed by Ensembl VEP one by one. Most of the IDs that returned an error had outdated RefSeq transcript versions. For every transcript, Ensembl can only serve a specific version, which most of the time is the latest for GRCh38. This was resolved by automatically querying the RefSeq ID in NCBI *Nucleotide* (NCBI) that will link to the latest RefSeq transcript version. For some IDs this still failed, in which the version was removed completely. Neither of these two strategies (version updating versus removing) is optimal and works in all cases. Finally, if both strategies failed, manual investigation was needed. Some faults (missing colons, mangled protein annotations, missing right-bounds) could be fixed, others had mismatching reference annotations, deprecated transcripts, missing variant annotations, or ambivalent range annotations.

For transparency, all changes to the IDs were documented in the dataset (published in Strauch et al., 2022).

4.3.2 Splice Variant Classification

All ML models from chapter 3 were trained on the whole corpus (train and test) of *gencode.v33grch38.primary* and *gencode.v33grch38.collapsed* using the corresponding hyper parameters reported. By teaching it splicing based on all chromosomes, it is expected that the overall understanding will grow due to the bigger size of data. The test split is unimportant for this experiment as the algorithm is evaluated on a completely independent validation set of variants. Additionally, the published SpliceAI models (McRae et al., 2019a) were included for comparison.

Base classifiers can only predict one site at once, while CNNs can predict thousands of neighbouring nucleotides. Base classifiers were therefore applied repeatedly for each position to generate a sequence output in the same format. Variant classification for baseline algorithms was implemented in the same way as described for deep learning (see Figure 4.1), so that all outputs have the same data format.

All models were run on the raw DNA input in a custom python script, which means that the annotation library from SpliceAI (McRae et al., 2019a) was not used, only the five CNN models.

Apart from reporting the accuracy between ground truth and significant changes, the AUC-PR between $\max(|\delta|)$ and annotations is reported to allow for better comparison of different thresholds. To account for deep variants, the maximum distance from a variant was set to 5,000 in this experimentation (i.e. baseline classifiers are run 5,000 times per variant, and the input sequence for CNN was set to 5,000 nucleotides plus 10,000 nucleotides providing context).

ML algorithms were run on all variants to measure how well they can predict if splicing is affected or not. The optimal threshold on this data is calculated by maximising accuracy respective to all possible thresholds.

4.3.2.1 Insertions and Deletions

For SNVs (and multiple successive SNVs), the δ-score can be built by direct subtraction. For insertions and deletions, the two matrices are not of equal length and cannot be subtracted directly. One needs to either pad the shorter or truncate the longer predictive vector. This problem was analysed thoroughly in table 4.1.

Nine scenarios were identified. As stated, the most trivial scenario is when the number of nucleotides between canonical and variant sequence do not change. Insertion and deletion was each broken down into four scenarios: If an annotated[1] nucleotide was deleted/inserted and/or if the variant affects splicing or not. Table 4.1 illustrates all nine scenarios, the underlying mathematical problem, and three mitigation strategies.

There are two basic strategies: Padding the shorter vector with zeros, and shortening the longer vector using a max function. The third strategy, used by SpliceAI (McRae et al., 2019a), is a hybrid that only changes the prediction vector of the variant and fails in one scenario.

Therefore, the max strategy seems to be the theoretically optimal solution as it has no scenario in which it fails. There is however a practical problem with it: For deletions, the strategy requires truncation of the reference sequence; if the delta score is highest at this position, the position of the variant on the reference genome is not immediately clear. The exact position however is of interest for clinical interpretation (more on that in chapter 6). Instead of further complicating the algorithm to rectify this through tracking of which original nucleotide returned the highest score, it was decided to adapt the strategy by Jaganathan et al. (2019). The scenario in which this approach fails, the deletion of an annotated nucleotide without affecting splicing, is a very rare (if not highly unlikely) edge case that was not observed to occur in the experimentation conducted.

4.3.3 Explaining SpliceAI Variant Annotations

To try to explain how the network reaches its conclusions, Integrated Gradients (see section 2.2.5.2 on page 29) were applied to the original SpliceAI models. To allow doing so, the whole variant prediction architecture was ported to keras and a *DifferenceModel* was implemented that allows integrating over the whole process. It takes two DNA input vectors, one for each the reference and variant genome, pipes them through the ensemble models, applies the insertion/deletion strategy if needed, and outputs the predictive change for all three classes per nucleotide, representing acceptor/donor loss and gain. Gradients between a specific variant input and a neutral all-zero input can then be traced back and integrated using the code described in section 2.2.5.2 (page 29).

The output of this new DifferenceModel is the change in prediction, the delta matrix. To exemplify how powerful and complex this analysis can be: Each of these 10,001 predictions (5,000 nucleotides max distance to the variant plus the variant itself, assuming an SNV) can be traced back to two input RNA sequences of each 20,001 nucleotides (output size plus 5,000 nucleotides of context either side), that in turn have four dimensions (A,C,G,T), totalling over 1.6 billion possible gradients to track.

This experimentation focuses on the clinically most relevant gradients: Those ending in the most significant acceptor/donor loss and gain ($|\delta| > 0.2$ per class).

[1]The machine learning algorithm annotates the first and last nucleotides of an exon as acceptor or donor

Scenario	Same Length			Deletion											
Annotated nucleotide involed?	*irrelevant*			**0**			**1**			**1**			**0**		
Splicing affected?	*irrelevant*			**0**			**0**			**1**			**1**		
Predictions															
Example Canonical Sequence	A	G	**T**	G	**T**	T	G	**G**	T	A	**G**	T	A	G	**T**
Canonical Donor Prediction	0	1	0	1	0	0	0	1	0	0	1	0	0	1	0
Example Variant Sequence	A	G	**G**	G		T	G		T	A		T	A	G	
Variant Donor Predictions	0	0	0	1		0	1		0	0		0	0	0	
Direct Subtraction															
|δ|-Score	0	1	0	0	n/a	0	1	n/a	0	0	n/a	0	0	1	n/a
Classification	**1**			**n/a**			**n/a**			**n/a**			**n/a**		
Fill Shorter With Zeros															
Processed Canonical Donor Prediction	0	1	0	1	0	0	0	1	0	0	1	0	0	1	0
Processed Variant Donor Predictions	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0
|δ|-Score	0	1	0	0	0	0	1	1	0	0	1	0	0	1	0
Classification	**1**			**0**			**1**			**1**			**1**		
Truncate Longer With Max															
Processed Canonical Donor Prediction	0	1	0	1		0	1		0	1		0	0	1	
Processed Variant Donor Predictions	0	0	0	1		0	1		0	0		0	0	0	
|δ|-Score	0	1	0	0		0	0		0	1		0	0	1	
Classification	**1**			**0**			**0**			**1**			**1**		
SpliceAI: Only change variant															
Untouched Canonical Donor Predictions	0	1	0	1	0	0	0	1	0	0	1	0	0	1	0
Processed Variant Donor Predictions	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0
|δ|-Score	0	1	0	0	0	0	1	1	0	0	1	0	0	1	0
Classification	**1**			**0**			**1**			**1**			**1**		
	Same Length			**Deletion**											

Scenario	Insertion											
Annotated nucleotide involed?	**0**			**1**			**1**			**0**		
Splicing affected?	**0**			**0**			**1**			**1**		
Predictions												
Example Canonical Sequence	G		T	G		T	A		T	A	G	
Canonical Donor Prediction	1		0	1		0	0		0	0	1	
Example Variant Sequence	G	**T**	T	G	**G**	T	A	**G**	T	A	G	**T**
Variant Donor Predictions	1	0	0	0	1	0	0	1	0	0	0	0
Direct Subtraction												
|δ|-Score	0	n/a	0	1	n/a	0	0	n/a	0	0	1	n/a
Classification	**n/a**			**n/a**			**n/a**			**n/a**		
Fill Shorter With Zeros												
Processed Canonical Donor Prediction	1	0	0	1	0	0	0	0	0	0	1	0
Processed Variant Donor Predictions	1	0	0	0	1	0	0	1	0	0	0	0
|δ|-Score	0	0	0	1	1	0	0	1	0	0	1	0
Classification	**0**			**1**			**1**			**1**		
Truncate Longer With Max												
Processed Canonical Donor Prediction	1		0	1		0	0		0	0	1	
Processed Variant Donor Predictions	1		0	1		0	1		0	0	0	
|δ|-Score	0		0	0		0	1		0	0	1	
Classification	**0**			**0**			**1**			**1**		
SpliceAI: Only change variant												
Untouched Canonical Donor Predictions	1		0	1		0	0		0	0	1	
Processed Variant Donor Predictions	1		0	1		0	1		0	0	0	
|δ|-Score	0		0	0		0	1		0	0	1	
Classification	**0**			**0**			**1**			**1**		
	Insertion											

Predictions are example data. The real algorithm emits three vectors, one for each class (acceptor/donor/neither).

TABLE 4.1: **Strategies to calculate the $|\delta|$-score between a canonical and variant sequence for the predictive donor vector.** When the canonical and variant sequence have the same length, this vector can be built directly by simple subtraction. For insertions and deletions, one can fill missing predictions with zeros or truncate the longer predictive vector using a max function. Filling missing predictions with zeros fails in two scenarios. Reducing the longer prediction vector with a max function works in all scenarios. Combining both approaches to only change the variant predictions, as SpliceAI (McRae et al., 2019a) does, fails in one scenario: When the annotated nucleotide is deleted without affecting splicing.

4.4 Results

4.4.1 Duplicate and Conflict Resolution of Variant Ground Truth

142 duplicates were resolved based on their genomic locations, reference and variant annotation. This resulted in 3,867 distinct variants, however 5 of them were found to have conflicting splicing annotations, see table 4.2. In four out of five instances, Jian et al. citing HGMD (Stenson et al., 2020) was involved. Out of these four, only two original sources within HGMD were found, and both times the source comes to no conclusive result.

Variant	Splice Affecting	Citation Chain(s)	Observation	Resolution
NM_000059.3:c.6935A>T	0	Houdayer et al.	No change	Keep and Merge
NM_000059.3:c.6935A>T	0	Leman et al. [Houdayer et al.]	No change	Keep and Merge
chr13:32918788A>T	1	Jian et al. [HGMD [Brandão et al.]]		Remove Source
NM_000059.3:c.9116C>T	0	Houdayer et al.	No change	Keep and Merge
NM_000059.3:c.9116C>T	0	Leman et al. [Houdayer et al.]	No change	Keep and Merge
chr13:32954049C>T	1	Jian et al. [HGMD [Llort et al.]]		Remove Source
NM_000138.4:c.3963A>G	0	Wai		Keep
chr15:48773853T>C	1	Jian et al. [HGMD [?]]		Remove Source
NM_000138.4:c.1588G>A	0	Wai		Keep
chr15:48805746C>T	1	Jian et al. [HGMD [?]]		Remove Source
NM_000059.3:c.9502-12T>G	0	Houdayer et al.	No change	Remove Variant
	1	Leman et al. [Houdayer et al.]	Exon 26 skipped	Remove Variant

TABLE 4.2: **Five variants where literature disagrees.** Brackets indicate citation chains. Four of five disputes contain Jian et al., one is a mis-citation of the Houdayer et al.-Leman et al. research group. Conflicts were resolved by removing the complete Jian et al. source and the last dispute was resolved by removing both the variant annotations.

For the first variant, chr13:32918788A>T (GRCh37), HGMD references Brandão et al. (2011). The original publication states:

> ”BRCA2 c.6935A>T, besides expressing the full-length transcript, increased expression of [..] BRCA2Δ12 [..]. As these are naturally occurring isoforms, also observed in controls, the clinical relevance is unclear.”
>
> (Brandão et al., 2011)

This is unlikely a splice site disruption.

The second variant, chr13:32954049C>T (GRCh37) comes from Llort et al. (2002), which calls it "Pro3039Leu". A footnote at this variant specifies:

> "These variants occur in conserved bases adjacent to intron/exon junctions and are predicted to interfere with acceptor site recognition (`http://www.fruitfly.org/seq_tools/splice.html`, Splice Site Prediction by Neural Network)."
>
> (Llort et al., 2002)

The link refers to a web interface for NNSPLICE (Reese et al., 1997). This reads as if this variant was not actually tested using functional analysis, but rather a result of automated annotation.

The other two instances where Jian et al. cites HGMD (chr15:48773853T>C and chr15:-48805746C>T, both GRCh37) could not be found in HGMD.

All these results indicate that the quality of the Jian et al. data set is not sufficient, and even the data quality without conflicts is hard to assess: One of the sources used by this paper, "SpliceDisease" (Wang et al., 2012), is not available any more. It was therefore decided to completely drop the Jian et al. source.

The last disputed entry was Leman et al. incorrectly citing Houdayer et al.; which found NM_000059.3:c.9502-12T>G to not affect splicing (1S), and Leman et al. cites Houdayer et al. and reports exon 26 to be skipped (2S). In the Leman et al. table, this is the last entry, but there is an almost empty row under it with an orphaned 1S annotation, making this line likely to be a copy-and-paste error in the Leman et al. table. To be sure, this variant was removed completely.

4.4.2 Overview of Aggregated Splice Variants

The curated dataset contains 1,209 functionally validated variants. Figure 4.2 visualises the data: There is almost a 50/50 split of strandedness and between affecting vs non-affecting variants. There is 1.4% overlap between sources (excluding all overlaps between direct reproductions of Houdayer et al. and Leman et al.). Most variants are SNVs, and most MNVs[2] are insertions. The data is considerably biased towards chromosomes 13, 17, 11, 1 and 15, mostly due to many sources investigating breast cancer. VEP predicts a third of all variants to be relating to splicing and a quarter to be intronic. The effect on the protein code is annotated to be vastly missense and a quarter are predicted to be synonymous. All variants affecting splicing are ACMG PS3 due to functional validation of the sources.

[2]This chapter will use the term MNV to refer to everything that is not an SNV, i.e. multi-nucleotide substitutions, insertions, and deletions

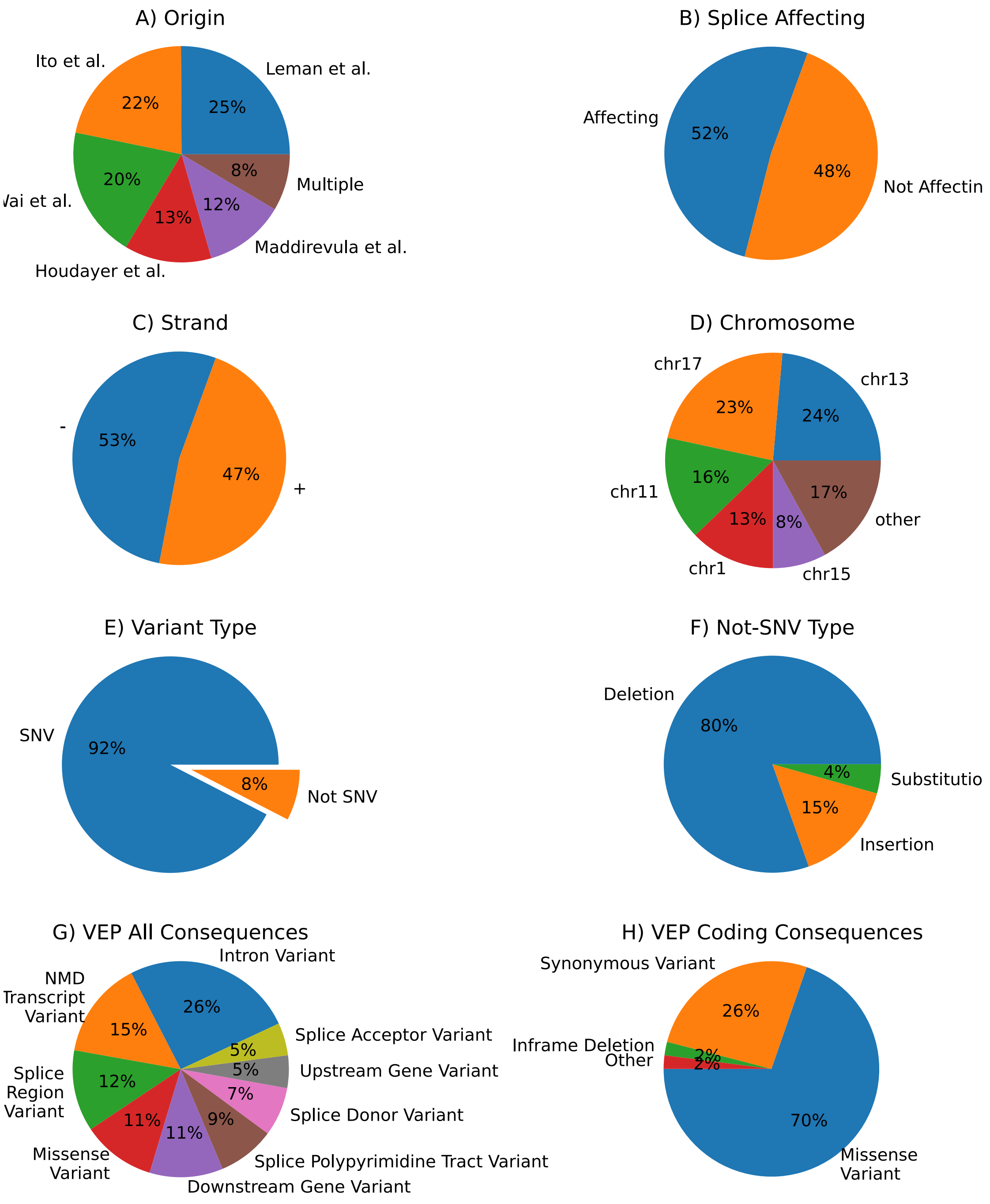


FIGURE 4.2: **The final variant data set consists of 1,209 unique variants.** (a) Apart from some direct citations between Houdayer et al. citing Leman et al., there is an overlap of 1.4% of variants, all with consensus. (b) The data set has an almost 50/50 split between variants affecting splicing and those without effect. (c) The data set is reasonably balanced between negatively and positively stranded genes. (d) Chromosomes 1, 11, 13, and 17 make up 75% of the variant data set. This could produce biased results. (e) The vast majority of variants are SNVs. The 4.7% MNVs are shown in (f): There is a significant bias towards deletions, followed by insertions. (g) VEP predicts a third of variants to be related to splicing and (h) the vast majority of consequences for the protein to be missense.

Three reverse-complemented data points in the source data were annotated on the wrong strand. After flipping their reference and variant annotation to the reverse strand, all three classified correctly by the CNN on *gencode.v33grch38.collapsed* (and three others), indicating that reversing the strand was an appropriate fix.

4.4.3 Binary Classification of Splice Disruption

Table 4.3 shows the resulting AUC-PR between the $\max(|\delta|)$-score and annotated labels and the accuracy of significant changes given an optimal threshold. The CNNs perform best, led by the one trained on collapsed isoforms with about one percentage point improvement in the AUC-PR. The CNN trained on the primary dataset produced in this work performs as good as SpliceAI. There is a significant drop of six percentage points and more between CNNs and baseline classifiers. While the CNN performed better on the collapsed representation, almost all baseline classifiers produce better results when trained on primary isoforms (except for the accuracy measure for Logit).

Algorithm	Training Dataset	AUC-PR	Opt. Threshold	Accuracy
CNN	*gencode.v33grch38.collapsed*	96.37%	0.204	90.74%
SpliceAI	Published by McRae et al.	95.50%	0.306	90.32%
CNN	*gencode.v33grch38.primary*	95.46%	0.248	89.99%
RF500	*gencode.v33grch38.primary*	89.20%	0.205	81.56%
RF500	*gencode.v33grch38.collapsed*	88.91%	0.237	81.14%
RF20	*gencode.v33grch38.primary*	87.78%	0.268	81.06%
RF20	*gencode.v33grch38.collapsed*	87.40%	0.275	79.98%
MLPC	*gencode.v33grch38.primary*	86.20%	0.312	79.16%
MLPC	*gencode.v33grch38.collapsed*	85.74%	0.333	78.58%
SVC-RBF	*gencode.v33grch38.primary*	78.48%	0.254	75.68%
SVC-RBF	*gencode.v33grch38.collapsed*	78.41%	0.219	75.35%
Logit	*gencode.v33grch38.primary*	76.29%	0.190	74.86%
Logit	*gencode.v33grch38.collapsed*	76.01%	0.167	75.02%

TABLE 4.3: **Performance of the different algorithms respective to their training data evaluated on the binary variant classification task.**

The published SpliceAI model produces very similar scores to the models that were trained on *gencode.v33grch38.primary* and *gencode.v33grch38.collapsed*, again strengthening that the migration to IRIDIS was successful.

4.4.4 Explaining ML SpliceAI Variant Annotations

Figure 4.3 (spanning across two pages) visualises Integrated Gradients for NM_016124.4:-c.1152A>C. Due the immense input size, only the 600 nucleotides in the middle are shown. This specific variant was chosen because it is a clear example of a skipped exon and should help understanding how to use Integrated Gradients to explain variant annotations.

SpliceAI correctly predicts a skipped exon. There are two inputs, one for the reference and one for the variant genome, and on both splice sites, these contributions cancel each other out except for the nucleotide at the variant position. This makes sense: Running the model on identical REF and ALT sequences should return no change in splicing, and this must not be attributed to any one input. However, the summed acceptor loss returns positive contributions while the summed donor loss returns negative ones. Why that is could not be determined.

The granularity and high number of features makes it hard to extract and understand the underlying classification process and no insights into the underlying biology could be generated.

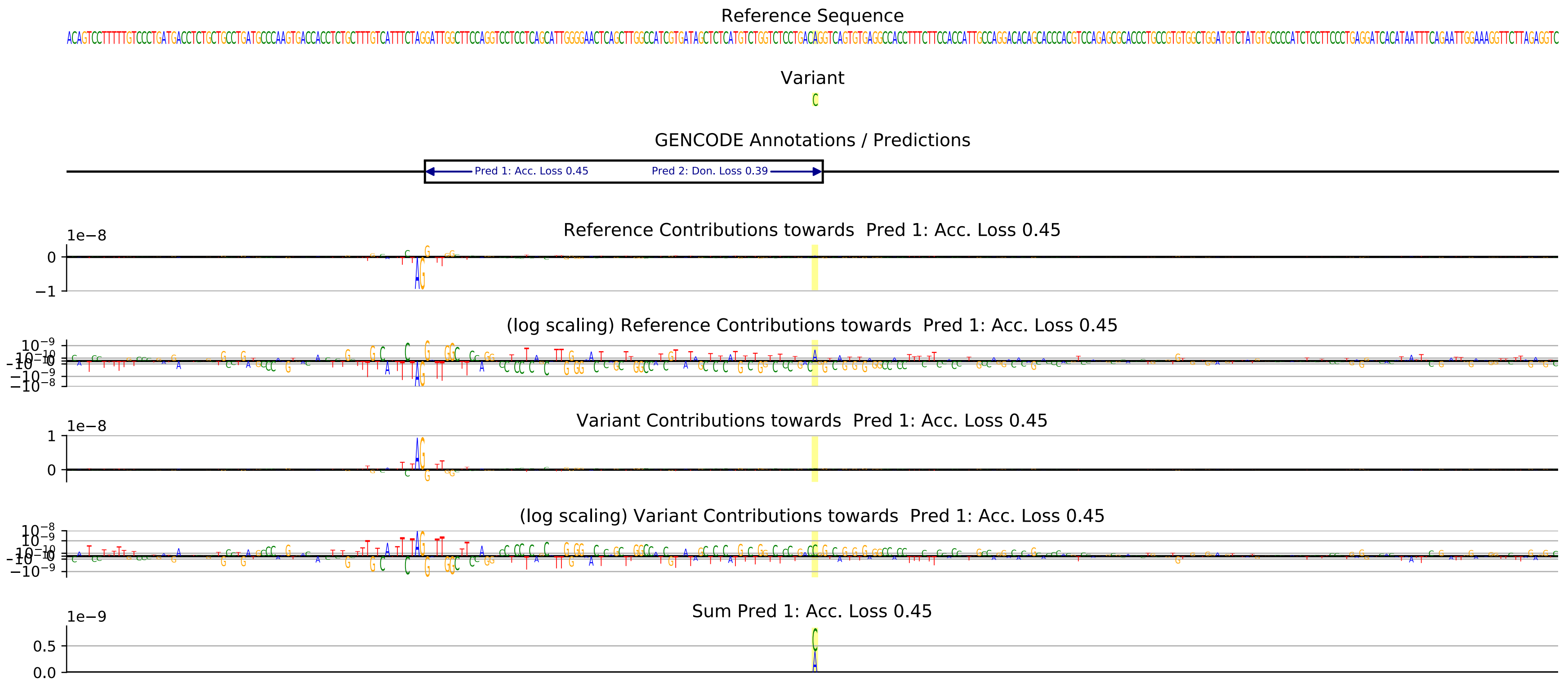


FIGURE 4.3: **Integrated Gradients for NM_016124.4:c.1152A>C**; continued on next page

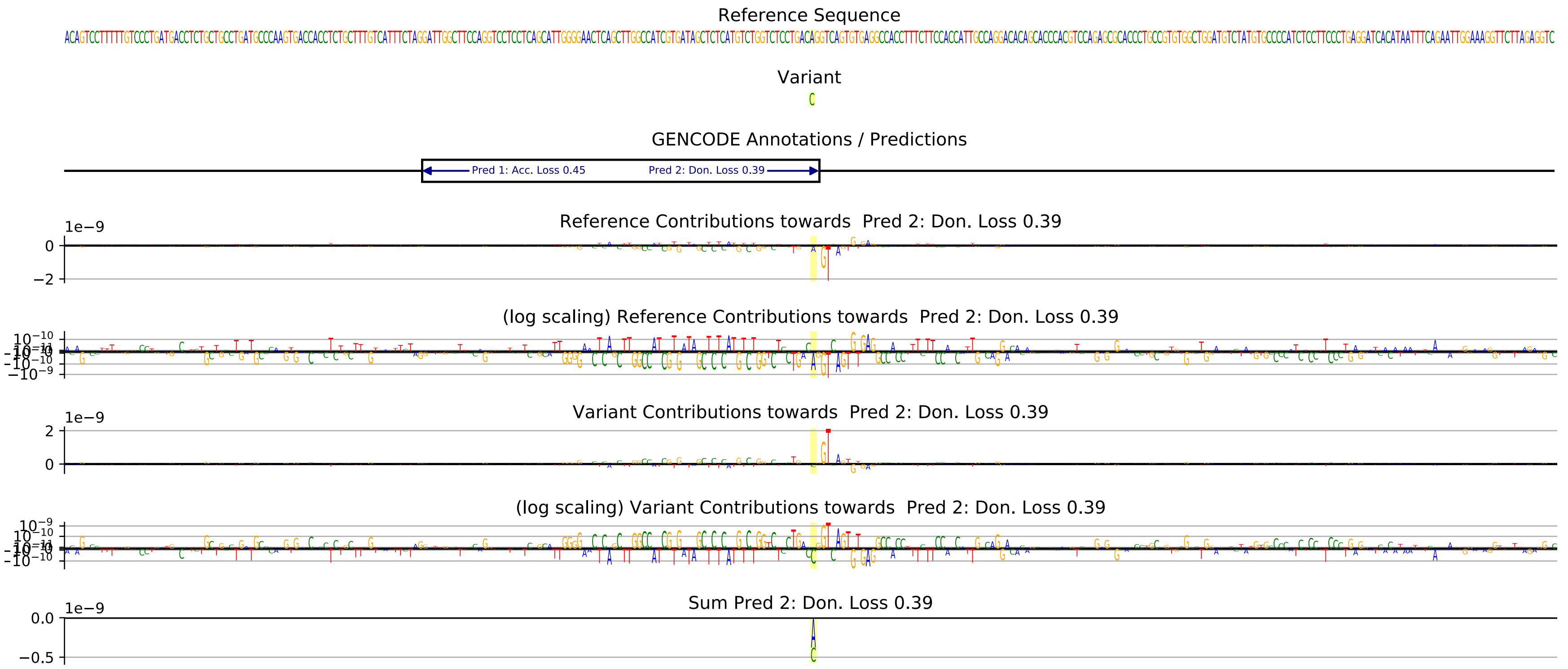


FIGURE 4.3 (cont.): **Integrated Gradients for NM_016124.4:c.1152A>C.** This variant (yellow backdrop) was shown to skip the whole exon using functional analysis. SpliceAI agrees; this is an example of it correctly predicting a pairwise loss. Only 600 input positions are shown around the variant.

4.5 Discussion

This dataset, consisting of 1,209 variants, is one of the biggest freely accessible datasets of splice sites and their functionally validated impact on splicing to date. Great care was taken to bring them into a shared data format and resolve disagreements. It is mostly balanced except for genes, chromosomes, and variant types. Achieving a balance however would be infeasible as it would greatly reduce the number of datapoints; it would also not reflect the distribution of variants observed in clinical practice.

The comparison between baseline and deep classifiers on variant data allowed a direct and fair comparison, in contrast to previous experiments that ran on different subsets of splice site data. Unfortunately, none of the baseline classifiers were found to be competitive; the deep learning methods with big context sizes are better at recognising splice sites from DNA sequences alone. Deep learning also performed better on the collapsed isoform dataset while almost all shallow learners could not cope with the added complexity. This might be due to hyper parameters of the baseline classifiers not being optimised on this data set, baseline classifiers being too simple, or the collapsed dataset being too hard. One of the strengths of deep learning is that it can be applied to raw data such as the sequencing input such as Jaganathan et al. were doing. For shallow machine learning, it is common to apply feature engineering to introduce domain knowledge and help the algorithm process information. For example, SQUIRLS uses decision trees and a Logit on engineered variant data and does not process any raw sequencing input.

The best algorithm, the CNN trained on collapsed isoforms, classifies variants a percentage point better for both AUC-PR and accuracy than the second best classifier, which is the published SpliceAI model. This is an exciting result and shows that our filtering to only include splice sites annotated by a human, rather than enriching GENCODE with novel GTEx junctions, is more appropriate on this set of variants.

One dataset (Jian et al., 2014) was removed due to concerns about its quality and direct conflicts with other sources. The remainder of sources disagreed in one other instance most likely representing a mis-citation. Conflicts could also be explained by the different ways to assess aberrant splicing: A variant could alter expression in a minigene assay but not in whole blood or vice versa. The data sources used a variety of different ways to assess splicing defects which could allow for such conflicts. Resolving conflicts by filtering the data to a single method of splice assessment however would not necessarily be appropriate for this analysis as the ML algorithm predicts all isoforms at once. Isoform- and especially tissue-specific predictions could solve this fundamental issue in the future.

4.6 Conclusion

A substantial dataset for the binary prediction of splice affecting vs not affecting was curated.

Based on the annotation of splice sites, a novel strategy to cope with insertions and deletions was introduced. While it was theoretically better than the one used in SpliceAI, in practice it would not return correct positions and therefore offset annotation of which sites are disrupted and was therefore abandoned.

The dataset with collapsed representations based on a new GENCODE filtering was shown to return the best results on the variant data when being fed to the SpliceAI CNN architecture. It was was able to reproduce functional analysis results with high scores from DNA sequences alone for both annotation tasks. We can therefore conclude objective 2, applying supervised algorithms to annotate aberrant splicing in variant data, to be resolved successfully.

Now that there is a novel algorithm, trained on the collapsed representation, that was shown to outperform the models of SpliceAI, it should be made accessible to the wider scientific community. It also needs to be compared to other tools from the field, and an annotation library to facilitate running it needs to be developed. Chapter 6 describes how the software package CI-SpliceAI was created to do so.

The Integrated Gradients approach did not return interpretable results; deep learning appears to remain a black box in this context. It might be preferable to try to optimise baseline algorithms instead as they are easier to interpret.

The baseline classifiers however did not return competitive results. It was suggested that their input needs to be engineered as their models may not be complex enough to process raw sequence input. The next chapter will explore strategies of annotating and encoding the sequencing data more appropriately.

Chapter 5

Improving Splice Site Recognition through Data Engineering

5.1 Introduction

In the previous chapter, it was shown that deep learning is superior on using raw sequence input to predict splice site disruption due to model complexity. While deep learning is known to work well on raw data, baseline classifiers seem to perform badly when given raw DNA sequences as input. Classical ML (i.e. not deep learning) needs more data engineering to produce competing results. For example, SQUIRLS (section 1.1.2 on page 4) is trained on engineered features like information content and delta scores at the closest splice sites to predict if a variant affects splicing or not. As motivated in section 4.2.1 (page 66), it would be preferable to not train on variant prediction and remain training on sequence input to allow analysis of the underlying models and try to gain insights into the mechanism itself.

There are some notable differences between baseline classifiers and deep learning methods in the context of splicing. Baseline classifiers take each input separately (i.e. "is there a T nucleotide at position 5?") whereas the filters of CNNs look at groups of neighbouring nucleotides at the same time, inherently looking at context. CNN filters are also moved over the input and therefore recognise features independent of their location, whereas baseline classifiers will treat a motif differently depending on its relative position. Lastly, the immense depth of the deep learning methods may allow the network to treat dissimilar sequences with similar functions in the same way, while baseline classifiers cannot model this complexity on their own and features need to be engineered to compensate.

We know that protein binding sites can occur anywhere around a splice site (see section 1.1), which will surely not be recognised by simpler ML algorithms. Binding sites will be annotated explicitly to help baseline algorithms recognise them independent of their position.

The one-hot encoding (see section 1.3.3 on page 23) might not be the best way to encode sequences for baseline classifiers as it forces the algorithm to focus on one nucleotide at a time. Instead, a new *semantic encoding* will be evaluated that allows variations of similar input sequences to be encoded similarly and hopefully require less complexity within the ML model itself. A potential way of engineering features better is DNA2Vec (Ng, 2017a), encoding DNA in a continuous vector representation where similar sequences lie close to each other, a so called *semantic space*. This could help ML algorithms to have more general rules since similar sequences are encoded similarly, therefore allowing them to focus on bigger patterns and requiring less complexity.

Lastly, we also know that splice sites are conserved more highly throughout evolution (section 5.2.1). Previous applications included conservation scores into their methodology (i.e. Cheng et al., 2019; Danis et al., 2021). Annotation of genomic conservation might help classification to recognise splice sites more easily and will be explored for baseline classification.

5.1.1 Aims and Objectives

The aim of this chapter is to improve baseline classification through data engineering, see objective 3. Three strategies for feature engineering of baseline classification are to be evaluated in independent experiments. For each strategy, the impact of baseline classification performance on splice site recognition is to be measured. If any strategy successfully improves splice site recognition, the trained algorithm is then applied to the variant data from section 4 to measure if they improve clinical utility.

Three strategies are to be evaluated:

1. Annotation of regulatory binding motifs from the literature
2. Annotation of evolutionary conservation
3. Transformation of one-hot encoded data into semantic vector space

5.2 Background

5.2.1 Measuring Evolutionary Conservation

By comparing genomes of different species, one can observe high conservation at canonical splice sites (Blakes et al., 2022). The exact method to derive this however can vary, and the two predominant methods are *PhastCons* that uses a hidden markov model to represent evolutionary mutations of regions (Siepel et al., 2005), and *phyloP* ("phylogeneticP-values") which models the probabilities of such mutations at each individual nucleotide (Pollard et al., 2010). PhastCons are smoother and better in modelling longer regions, whereas phyloP returns more granular results. This is illustrated in Figure 5.4 on page 92.

Both methods need a pool of species to align to the Human Genome. The set of species is the same for both methods, and the two predominant vertebrate supersets include 46 or 100 species across the subsets Primate, Euarchontoglires, Laurasiatheria, Afrotheria, Mammal, Aves, Sarcopterygii, and Fish. These data sets are referred to as *PhastCons46*, *PhastCons100*, *phyloP46*, and *phyloP100*.

SQUIRLS include the mean *phyloP100* score of the reference allele in their feature set (Danis et al., 2021). MMSplice also experimented with phyloP scores (Cheng et al., 2019). The best idea might be to empirically try out all variations of evolutionary conservation and pick the best.

5.2.2 Regulatory Splice Motifs

Alternative splicing is regulated through complex protein regulation and interaction (see section 1.1 on page 1). A general approach to categorise regulatory proteins is into splicing enhancers and silencers, which in turn can occur within the exon or intron. As described in section 1.1.1, while the general categorisation of protein into enhancers and silencers is a common one, it might be an oversimplification as some protein are known to both excite and exhibit splicing depending on where they are relative to the splice site (Dvinge, 2018). How this will affect predictive accuracy is unknown.

The freely accessible *INT3-400* dataset by Cáceres and Hurst (2013) contains 54 ESE 6-mers. 130 ESS 8-mers were published by Wang et al. (2004).

5.2.3 Semantic Encoding of DNA Sequences

As an alternative to the one-hot encoding of sequences, encoding DNA in a *semantic space* through DNA2Vec (Ng, 2017a) might help the algorithms to understand context. DNA2Vec is an adaptation of *Word2Vec* (Mikolov et al., 2013) which in turn is a variation of an *autoencoder*. The following sections describe the background in relation to Natural Language Processing, where these concepts originated in, before transferring them to the nucleotide encoding of DNA.

5.2.3.1 Autoencoders

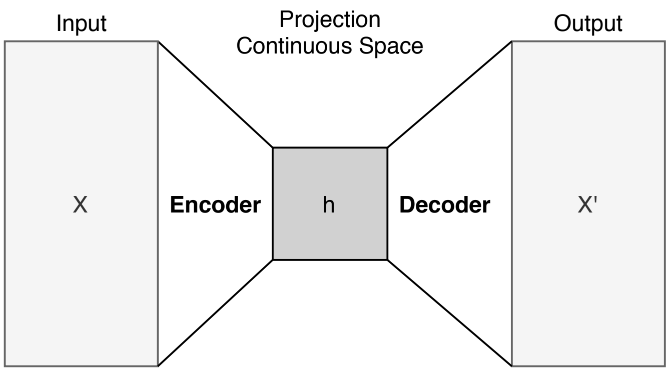


FIGURE 5.1: **An autoencoder is a simple neural network with one hidden layer.** Input and output layers have the same number of neurons. The encoder part of an autoencoder compresses the input X to a representation h in continuous space. This representation can then be decoded into X', where $X' \approx X$. Adapted from Pinaya et al. (2020)

Autoencoders (Figure 5.1) are two layer ANNs where the input and output layers have the same number of neurons. The training target is for the output to be as similar to the input as possible. This would be a non-trivial task if the hidden layer h in the middle had not significantly fewer neurons, forcing the model to learn a compression and decompression strategy called *encoder* and *decoder* respectively. (Baldi, 2012)

The projection h is a representation of the data in compressed, continuous space, which is equal to what PCA returns (Bourlard and Kamp, 1988). Apart from using autoencoders as a (de-)compression algorithm, they are also used for denoising, as a generative model emitting novel data, for pretraining of deep Neural Networks, and even machine translation of natural language (Lample et al., 2017; Pinaya et al., 2020).

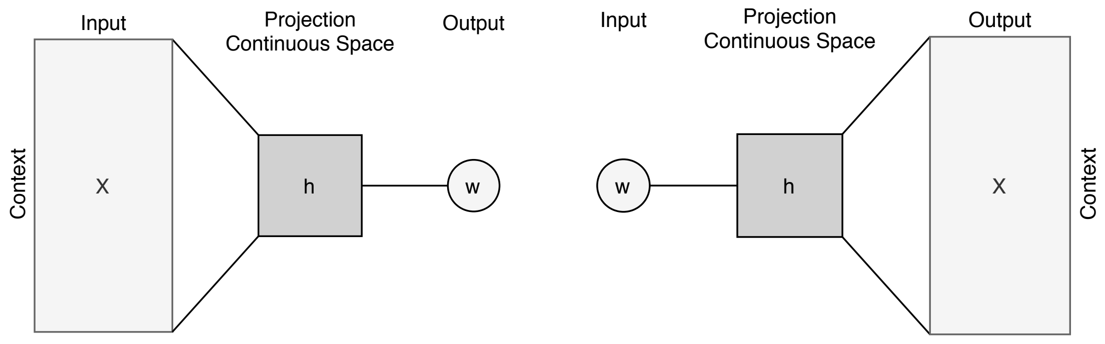
5.2.3.2 N-Gram and K-Mer Encoding

In Natural Language Processing, encoding every word independently from each other can lose the surrounding *context*, depending on further data processing and algorithmic choice. Retaining the context of neighbouring data might be desirable.

For a fixed context length, one could just stack all one-hot encoded neighbouring words to form a matrix. For sentences of varying length, this would produce matrices of varying shape which are unfit for many applications. Instead, *N-Grams* (ordered tuples of N successive words) are often one-hot encoded themselves, meaning each distinct N-Gram is encoded as its own vector (Shannon, 1948). In the genetic context, N-Grams would be equivalent to *K-Mers*, i.e. successive nucleotides of length K, and the same ideas apply.

By encoding each N-Gram or K-Mer into one dimension, the embedding vector length grows significantly. The bigger the tuple size (i.e. the more context is encoded), the rarer the N-Grams, and the bigger the vocabulary. This worsens the curse of dimensionality and its drawbacks drastically.

5.2.3.3 Continuous Vector Encoding of N-Grams



(A) Continuous Bag of Words Method (B) Skip-Gram Method

FIGURE 5.2: **Word2Vec projects one-hot encoded language data into h, a semantic continuous space.** There are two architectures of Word2Vec. (a) In the Continuous Bag of Words method, the context, that is the surrounding words of w, is used as an input, and the model predicts w. (b) The Skip-Gram architecture trains the model to predict the surrounding context of the word w. Adapted from Mikolov et al. (2013)

Word2Vec (Mikolov et al., 2013) is an unsupervised two layer feed-forward ANN used to project one-hot encoded N-Grams into continuous space, resolving the shortcomings of one-hot encoding.

There are two training architectures. The *Continuous Bag of Words* method (Figure 5.2a) predicts a word w from its context alone. The context is the set of immediately neighbouring words. The *Skip-Gram* architecture (Figure 5.2b) is reversed: The network predicts the surrounding context from w. The authors describe this architecture to train more slowly, but better fitting for infrequent words (Mikolov et al., 2015).

Looking at Figure 5.2, we can imagine joining the two networks on the vector space h and derive an autoencoder architecture (compare with Figure 5.1) that predicts the context from itself. We can therefore say that Word2Vec is closely related to autoencoders.

Both Word2Vec architectures have a continuous vector space in the hidden layer that allows vector arithmetic with surprisingly intelligent results. The vector function $V(w)$ returns the representation of word w in vector space. The authors found syntactical properties of that space: $V(\text{mouse}) - V(\text{mice}) \approx V(\text{dollar}) - V(\text{dollars})$, or $V(\text{biggest}) - V(\text{big}) + V(\text{small}) \approx V(\text{smallest})$. Even more exciting, the authors found semantic properties such as $V(\text{Athens}) - V(\text{Greece}) \approx V(\text{Oslo}) - V(\text{Norway})$ or $V(\text{king}) - V(\text{man}) + V(\text{woman}) \approx V(\text{queen})$. This shows that the model did learn about relationships of words, and the encoding is not statistical noise. Compared to the initial one-hot encoded representation where all terms had the same distance to each other, similar terms are now grouped together and some syntactical and semantical relationships are preserved in the vector space.

5.2.3.4 Continuous Vector Encoding of K-Mers

Word2Vec was adapted to DNA by DNA2Vec (Ng, 2017a). Since DNA has not a fixed vocabulary, the authors decided to iterate over DNA and take overlapping K-Mers of size [3..5] (k sampled randomly). They then trained a Skip-Gram architecture on this data set to optimise for rare sequences.

There are syntactic properties of the vector space for concatenating K-Mers. For example $V(\text{GAT}) + V(\textbf{ATC}) \approx V(\text{GAT}\textbf{ATC}) \approx V(\textbf{ATC}\text{GAT})$ or $V(\textbf{AC}\text{GAT}) - V(\text{GAT}) + V(\text{ATC}) \approx V(\textbf{AC}\text{ATC}) \approx V(\text{ATC}\textbf{AC})$. Since the addition in vector space is symmetric, the concatenation cannot distinguish if vectors are appended to the beginning or the end and are symmetric too.

These syntactic properties hold in most instances. For example, when concatenating two 3-mers in vector space, (i.e. add the vectors together), the five closest 6-mer vectors contain the desired 6-mer in 80.3% of cases (Ng, 2017a). Semantic properties were not found yet, which is remarkable since for natural language, the Skip-Gram architecture has been shown to produce better semantic qualities than the continuous bag of words model (Mikolov et al., 2013).

5.3 Methods

5.3.1 Incorporating Protein Regulation for Splice Prediction

The data pipeline for baseline classifiers is extended to include ESE and ESS motifs, provided by the *INT3-400* dataset by Cáceres and Hurst (2013) containing 54 ESE 6-mers, and a 130 ESS 8-mers dataset by Wang et al. (2004).

Each nucleotide in *gencode.v33grch38.primary* is annotated by counting how many binding motifs align with the reference genome per position. This creates two new feature vectors, each 81x1, that are appended to the one-hot encoded data before standardisation (end of step 2 in Figure 3.3 page 39). Figure 5.3 shows an example data point.

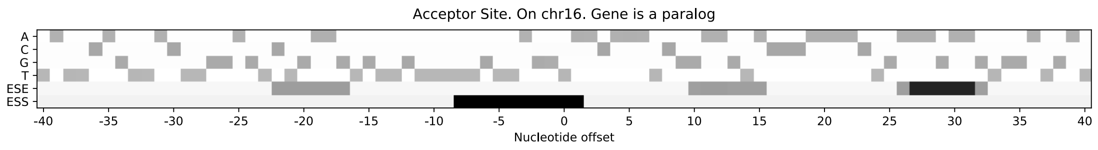


FIGURE 5.3: **Example data point with annotated ESE/ESS motifs.** There are two ESE annotations overlapping around +30, and one ESS motif starting at -8.

Even though the data sets contain *exonic* splicing enhancers and silencers, both exons and introns are annotated. Limiting annotations to exons would tell the algorithm where exons and introns are located, making splice site detection trivial.

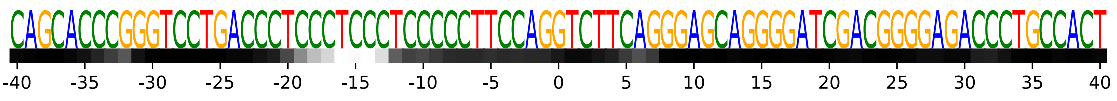
ESE and ESS annotations enlarged the number of features by 50%. Models are expected to need more complexity, it was therefore decided to repeat the hyper parameter optimisations presented in section 2.2.2.

Bigger MLPCs (300 and more neurons in a hidden layer) did not finish training due to their memory consumption. The grid search was reduced to the hidden layer configurations $L \in \{(50,),(100,),(100,20),(20,100)\}$.

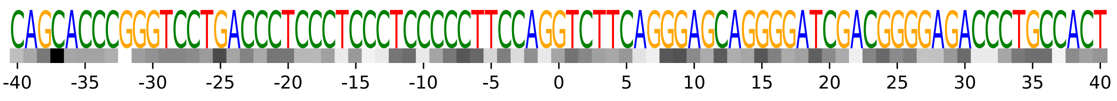
5.3.2 Incorporating Conservation Scores for Splice Prediction

It was unclear which conservation score data set would be most appropriate, therefore four conservation data sets, namely *PhastCons46*, *PhastCons100*, *phyloP46* and *phyloP100* (see section 5.2.1) were evaluated independently.

Annotations were downloaded from UCSC (University of California, b) which provides per-nucleotide conservation scores as *starch* files. These annotations were extracted using BEDOPS (Neph et al., 2012), filling missing values with 0. How the two conservations score methods differ is illustrated in Figure 5.4. The annotation vector is



(A) **PhastCons100 scores are smooth with few details.**



(B) **phyloP100 scores are detailed and rugged.**

FIGURE 5.4: **An acceptor site in *SAMD11* with annotated conservation scores.**

stacked on *gencode.v33grch38.primary* before standardisation and flattening (see step 4b of Figure 3.3). This results in four different new data sets.

For each conservation measure, baseline classifiers were optimised as described in section 2.2.2 and then evaluated using CV as described in section 2.2.4.1. SVCs and Logit are optimised on *subset2.gencode.v33grch38.primary* (see section 3.3.3), the remainder is optimised on the complete corpus. Due to bigger MLPCs (300 and more neurons in a hidden layer) not finishing training due to an unknown error (segmentation fault), the grid search was reduced to the hidden layer configurations $L \in \{(50,),(100,),(100,20),(20,100)\}$.

5.3.3 Training DNA2Vec

The published DNA2Vec model (Ng, 2017a) was originally trained on K-Mers of size [3..8]. Due to consensus sites being commonly referred to as 2-mers, DNA2Vec was retrained to include K-Mers of size [2..6].

The DNA2Vec training code (Ng, 2017b) was adapted to IRIDIS and to include K-Mers of size [2..6]. The code is built on *gensim* (Řehůřek et al., 2011), a state-of-the-art library for Natural Language Processing that Word2Vec is based on as well.

K-Mers are generated from UCSC hg38 (Human Reference Genome p12). The genome is sliced into fragments based on gap characters. Fragments are reverse-complemented with a probability of 0.5 per fragment. While iterating over each fragment, the K for each K-Mer is selected uniformly random from [2..6]. This overlaps neighbouring K-Mers and fixes the number of K-Mers to about the length of the sequence.

A two-layer MLP is trained for 100 epochs using the Skip-Gram Method (see Figure 5.2b page 89), i.e. given one K-Mer, the 10 preceding and 10 following K-Mers are predicted.

Two Word2Vec models were generated. One that maps K-Mers to 100-dimensional space and one to output two dimensions. The latter is used for visualisation of the model only, while the former is used as input for further experimentation.

Both models are expected to return mappings for all possible K-Mers. For $K \in [2..6]$ there are $\sum_{k=2}^{6} 4^k = 5,456$ distinct K-Mers to be found. If K was too big, not all possible K-Mers can be found, however with a $K < 8$ (which has been published by the original authors), this should not be a problem.

Both DNA2Vec models are to be visualised in a plot; the 2-dimensional space directly and the 100-dimensional space using PCA and t-SNE (see section 1.3.2 on page 21).

All K-Mers consisting of only one distinct base (i.e. AA, GGGG, TTT, etc., which will be referred to as *pure* K-Mers) are labelled, plus some more to illustrate the vector space and dimensionality reduction.

5.3.4 Application of DNA2Vec to the Splice Site Dataset

The data pipeline for baseline classifiers from chapter 3.3.1.2 (page 39) is to be changed from one-hot encoding to DNA2Vec encoding. Each data point spans over 81 nucleotides, and there are millions of data points. Since there is no intuitive way of separating "words", i.e. where each K-Mer starts and stops, the most thorough approach to build all possible sub-sequences followed by a feature selection was implemented. This work introduces the notion of a *KI-Mer*: A sequence slice of a specific length K at a specific offset I. Notice that a KI-Mer does not refer to a *specific* nucleotide sequence since it is applied across all data points in the dataset; it merely functions as a reference to a region relative to a potential splice site. This is due to the RFE algorithm being a global feature selection algorithm.

In a window of 81 nucleotides there are $81 - (K - 1)$ KI-Mers, resulting in $\sum_{k=2}^{6} 82 - k = 390$ KI-Mers per data point. Due to each KI-Mer consisting of 100 dimensions of 32bit floats, the nucleotide data of a single splice site would require $390 * 100 * 32\text{bit} = 156\text{KB}$ of data before compression.

The base data of *gencode.v33grch38.primary* consists of 373,420 splice sites (see table 3.2) and ten times more non-splice sites, totalling in $373,420 * 11 = 4,107,620$ sites. Encoding each site with 156KB would result in a data set of over 640GB for nucleotide data alone, excluding metadata and labels. Compression of the *hdfpy* format will decrease the size, but the data set will still be very big. This calculation did not even include considerations for baseline classifiers, which will hardly scale to such a big dataset. It was therefore decided to use *subset15.gencode.v33grch38.primary* with its 6,328 splice sites ($6,328 * 11 = 69,608$ sites in total), resulting in $6,328 * 11 * 156\text{KB} \approx 10.9\text{GB}$ of data.

5.3.5 Correlation Analysis of KI-Mers

Important KI-Mers need to be found for feature selection. Before conducting a lengthy RFE search, correlation analysis was conducted for preliminary assessment of data quality. This helped track errors in the data pipeline and is to give some intuitions about data usability.

The data set (*subset15.gencode.v33grch38.primary*) is sliced into KI-Mers, meaning that the algorithm only looks at one KI-Mer at a specific offset across all 69,608 data points at a time. Each slice is standardised by subtracting the mean and dividing by standard deviation, and a linear SVC is fitted on the binary classification tasks acceptor/rest and donor/rest. Feature importance for all 100 dimensions is estimated by probing the weights (see section 2.2.5.1 on page 29), and the importance of each KI-Mer is estimated by taking the maximum value of its 100 feature importances.

5.3.6 RFE Analysis of KI-Mers

This experiment was conducted after the correlation analysis produced sensible results and all data pipeline mistakes were fixed.

Hyper parameters for RF20 were found as described in section 2.2.2 on the complete data corpus (*subset15.gencode.v33grch38.primary*).

These parameters were then applied to RF500 in a RFE analysis, subject to two binary problem formulations (acceptor / rest and donor / rest) and one multi-class classification for all three classes (acceptor / donor / neither). Parallel to the correlation analysis, the data set is sliced into KI-Mers, however RFs require no feature standardisation and the analysis was conducted on all KI-Mers at the same time, rather than investigating each independently. The data corpus was split into train and test, according to partition 1 of table 3.1 (page 42). The classifier is fit on train data and tested on test data as described in section 2.2.1. Feature importances were estimated and the least important KI-Mer was removed at a time.

The removal of the weakest KI-Mer could not be implemented by the RFE function provided by *scikit-learn*, which only considers each feature separately. This is not flexible enough to select for individual KI-Mers, and a custom RFE algorithm was implemented. Both implementations are equivalent except for what happens after probing for feature importance: The custom RFE allows groupings of features by maxing over all 100 dimensions per KI-Mer while the library function only selects feature-wise. Therefore, the new RFE process 1) fits the model on all features; 2) estimates feature importance by probing the model; 3) calculates KI-Mer feature importance by taking the maximum feature weight of its 100 dimensions, 4) removes the KI-Mer of lowest importance with all its 100 dimensions; and 5) repeats elimination until one KI-Mer remains.

Due to the significant computational effort and IRIDIS not accepting jobs that run longer than 60 hours, the state of the algorithm after each elimination is saved to disk, allowing to resume computation if needed. The complete process (fit, test, probe, eliminate) is repeated until one KI-Mer remains and the elimination path is equal to a ranking in ascending order. Train and test AUC-PRs are plotted for analysis.

5.3.7 Training and Testing on Splice Site Recognition using top KI-Mers

The top 50 KI-Mers found through RFE analysis were used to encode *subset15.gencode.v33grch38.primary* and *subset2.gencode.v33grch38.primary*. On this data, hyper parameters were found as described in section 2.2.2: Linear SVC, RF20, and MLPC were optimised on *subset15.gencode.v33grch38.primary* and the remainder on *subset2.gencode.v33grch38.primary* due to time constraints. Models were then evaluated using CV as described in section 2.2.4.1 (page 28).

Because some SVCs with non-linear kernels and Logit timed out, the feature grid for SVCs was reduced to $C \in \{0.0001, 0.01, 1, 100\}$ and $\gamma \in \{0.0001, 0.01, 0.1, 1, 10\}$; the search over C parameters for Logit was reduced to $C \in \{0.001, 0.1, 10\}$

5.3.8 Application of Semantic DNA Representation to Variant Annotation

To evaluate if the DNA2Vec encoding improves annotation of splice disruptions, an experiment similar to chapter 4 was conducted.

First, all base algorithms were trained on all chromosomes of *subset15.gencode.v33grch38.primary* in DNA2Vec encoding, filtered to the 50 selected KI-Mers. The code to predict variants, described in section 4.3.2 on page 73, was adapted to encode sequences in semantic DNA2Vec space and, again, filter to the 50 features. This experiment was only conducted on the dataset of primary isoforms.

5.4 Results

5.4.1 Incorporating Protein Regulation for Splice Site Recognition

The resulting hyper parameters are shown in table 5.1. Notice that this AUC-PR is not a final result as it was not measured on the proper CV table and has potential data leakage between the train and test partitions. All measures decreased compared to the dataset without annotated ESE/ESS sites (table 3.6 on page 54), it was therefore decided not to continue with the CV analysis. Per-nucleotide annotation of protein binding sites appears to not help baseline classifiers recognise splice sites based on raw sequence input.

Algorithm	Best Parameters				AUC-PR	
					Measure	Improvement
SVC-linear	$b = true$	$C = 0.001$			73.02%	-20.51
SVC-RBF	$b = true$	$C = 10$	$\gamma = 0.001$		75.33%	-18.48
SVC-poly	$b = true$	$C = 0.001$	$\gamma = 0.1$	$d = 2$	60.59%	-28.84
Logit	$l = l_1$	$C = 0.1$			73.47%	-19.35
RF	$d_m = 22$				90.60%	-5.26
MLPC	$L = (50,)$				95.46%	-2.59

TABLE 5.1: **Hyper parameters found for baseline algorithms on *gencode.v33grch-38.primary* with annotated ESE/ESS motifs and the improvements against the base experiment.** Negative improvements represent worse results.

5.4.2 Incorporating Conservation Scores for Splice Site Recognition

The resulting hyper parameters are shown in table 5.2. Again, the AUC-PR is not a final result as it was not measured on the proper CV table, and again it was decided not to proceed with CV because the interim AUC-PRs were worse than the ones without conservation score annotations (table 3.6 on page 54). Generally, 100 vertebrate species performed better than 46 species throughout, and there is no consistent or significant difference between *PhastCons* and *phyloP*. Models did not grow more complex compared to the configurations found before (i.e. max-depth d_m of the RF), so it appears as if the annotations are just unnecessary.

Algorithm	Best Parameters				AUC-PR Measure	AUC-PR Improvement
PhastCons100						
SVC-linear	$b = true$	$C = 0.001$			76.78%	-16.75
SVC-RBF	$b = true$	$C = 10$	$\gamma = 0.001$		78.63%	-15.18
SVC-poly	$b = true$	$C = 0.01$	$\gamma = 0.01$	$d = 3$	65.09%	-24.34
Logit	$l = l_1$	$C = 0.1$			79.65%	-13.17
RF	$d_m = 20$				90.28%	-5.58
MLPC	$L = (50,)$				95.10%	-2.95
PhastCons46						
SVC-linear	$b = true$	$C = 0.001$			75.63%	-17.90
SVC-RBF	$b = true$	$C = 1$	$\gamma = 0.0001$		77.86%	-15.95
SVC-poly	$b = false$	$C = 0.01$	$\gamma = 0.1$	$d = 3$	64.23%	-25.20
Logit	$l = l_1$	$C = 0.1$			78.33%	-14.49
RF	$d_m = 18$				90.19%	-5.67
MLPC	$L = (50,)$				94.92%	-3.13
phylop100						
SVC-linear	$b = true$	$C = 0.001$			76.32%	-17.21
SVC-RBF	$b = true$	$C = 1$	$\gamma = 0.0001$		78.42%	-15.39
SVC-poly	$b = true$	$C = 0.1$	$\gamma = 0.01$	$d = 2$	65.31%	-24.12
Logit	$l = l_1$	$C = 0.1$			79.66%	-13.16
RF	$d_m = 19$				90.22%	-5.64
MLPC	$L = (50,)$				95.02%	-3.03
phylop46						
SVC-linear	$b = true$	$C = 0.001$			75.08%	-18.45
SVC-RBF	$b = true$	$C = 10$	$\gamma = 0.001$		78.18%	-15.63
SVC-poly	$b = false$	$C = 1$	$\gamma = 10$	$d = 2$	64.97%	-24.46
Logit	$l = l_1$	$C = 0.1$			78.49%	-14.33
RF	$d_m = 21$				90.35%	-5.51
MLPC	$L = (50,)$				95.04%	-3.01

TABLE 5.2: **Hyper parameters found for baseline algorithms found on *gencode.v33-grch38.primary* and *subset2.gencode.v33grch38.primary* with annotated conservation scores and their improvements to the base experiment.** Negative improvements represent worse results.

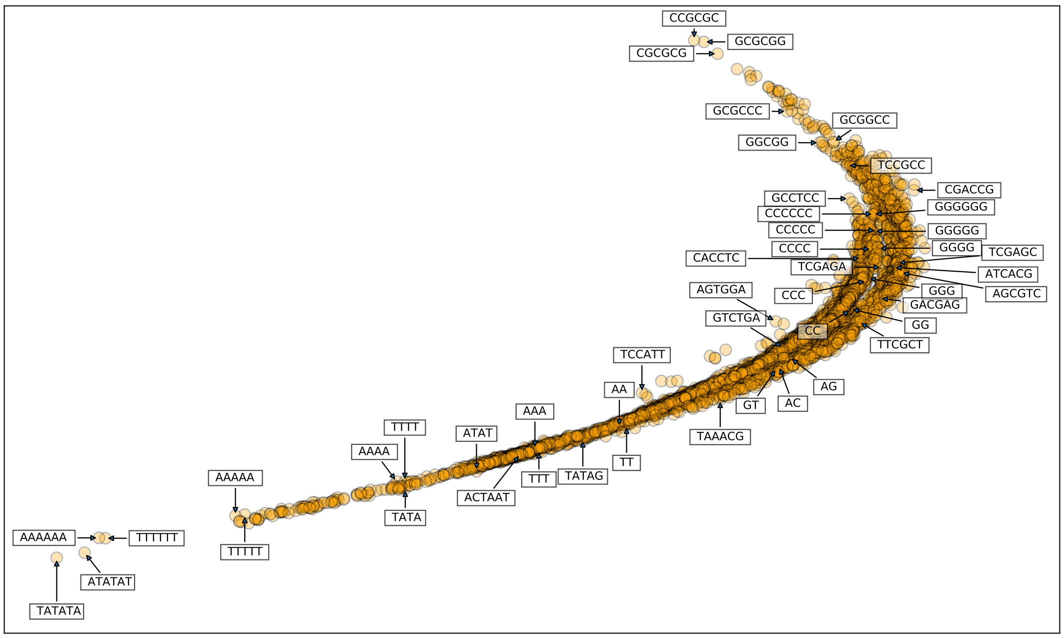
5.4.3 Analysis of Trained DNA2Vec Semantic Space

After 3-4 days of training each, the two generated DNA2Vec models contain all possible 5,456 K-Mers. The 2D representation of the derived semantic space (Figure 5.5a) forms a non-linear curve ranging from A/Ts to C/Gs with mixed sequences in between. The distribution overall is not exactly symmetric; the A/T portion is longer and thinner than the G/C part, however it is notable that the model learned the overall reverse-complementary relationship between A/Ts and C/Gs. Many reverse-complemented strands are found next to each other, probably caused by the data pipeline choosing to reverse-complement randomly. The TATATA/ATATAT K-Mers are separated substantially at the bottom-left. The first assumption that this is a semantic property due to TATA boxes starting transcripts did not hold up since variations of the TATA box are found throughout the plot and do not stand out similarly.

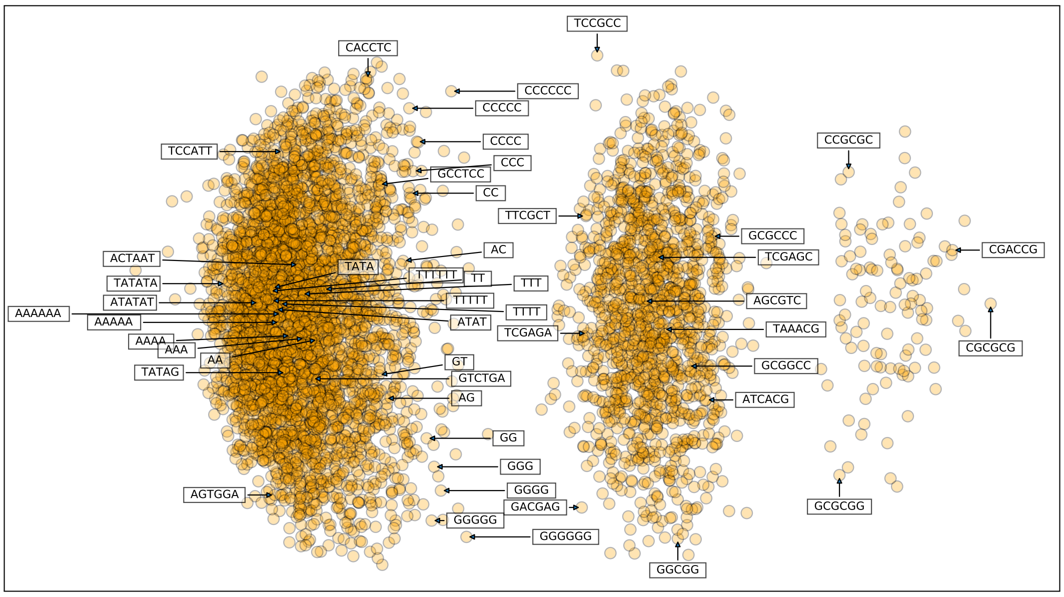
The PCA representation of the 100 dimensional space (Figure 5.5b) is not intuitive. Three main distributions are found, however there is no obvious interpretation for this. Pure K-Mers are all contained in roughly the same neighbourhood within the left distribution. As and Cs are close to each other, but do not have the same relationship as Cs and Gs.

The 100D space is better represented by t-SNE (Figure 5.5c), showing that similar sequences were mapped to similar regions in high dimensional space. All pure K-Mers are within direct neighbourhood and except for CC, all were mapped to almost the same coordinates. CC is not too far away from the other C K-mers. All TATA-Boxes were mapped to the same neighbourhood and are close to the pure A K-Mers; the TATAG K-Mer lies further down on the Y axis range where some Gs are observed.

From these visualisations we can infer that DNA2Vec learned two syntactical relationships of K-Mers: 1) Similar K-Mers are mapped close to each other in vector space, and 2) Reverse-Complemented K-Mers retained some relationship between each other. Concatenation was tested on a sample basis and held up as described in Ng (2017a).

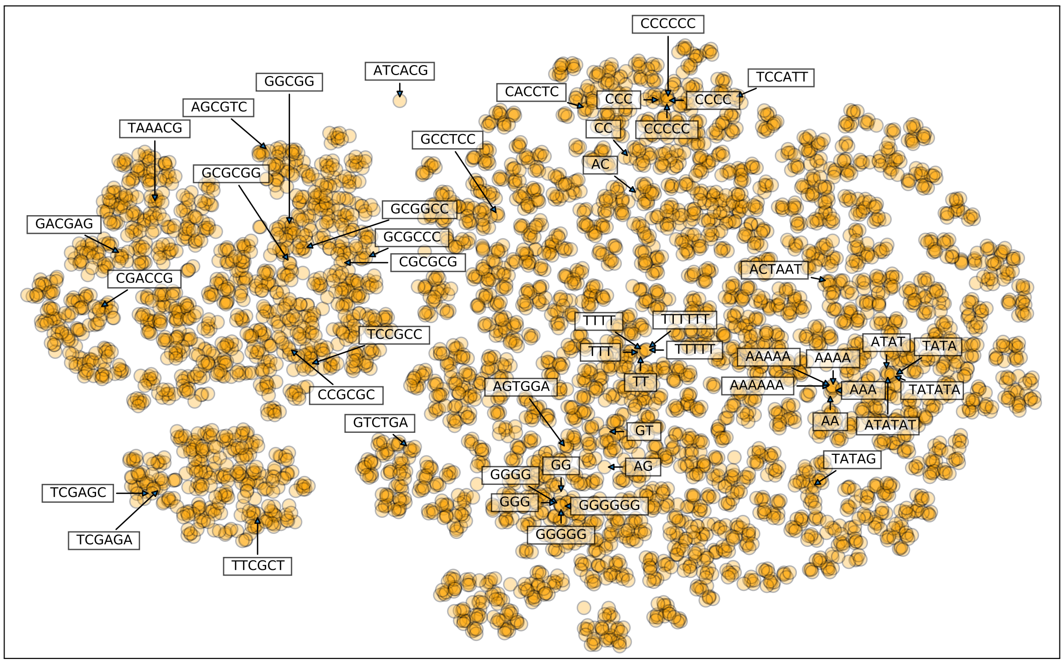


(A) 2D Vector Space of all K-Mers. When mapping to two dimensions, the K-Mers form a non-linear curve. At the end at the bottom-left, only Ts and As can be observed. The further "up-stream" (meaning moving along the distribution), the fewer As and Ts and the more Cs and Gs can be observed, and the distribution ends in only Cs and Gs. Reverse-Complements are generally very close together.



(B) 100D Vector Space of all K-Mers, reduced to 2D using PCA. Three clusters emerged, however there are no obvious patterns to be found. All K-Mers consisting of the same nucleotide are in the left distribution and reasonably close to another. The K-Mers consisting of only Cs are at the top-right of the first distribution, and mirrored to the bottom there are the K-Mers consisting of only Gs. This relation does not hold for all As to Gs.

FIGURE 5.5: **Continuous vector space**; continued on next page

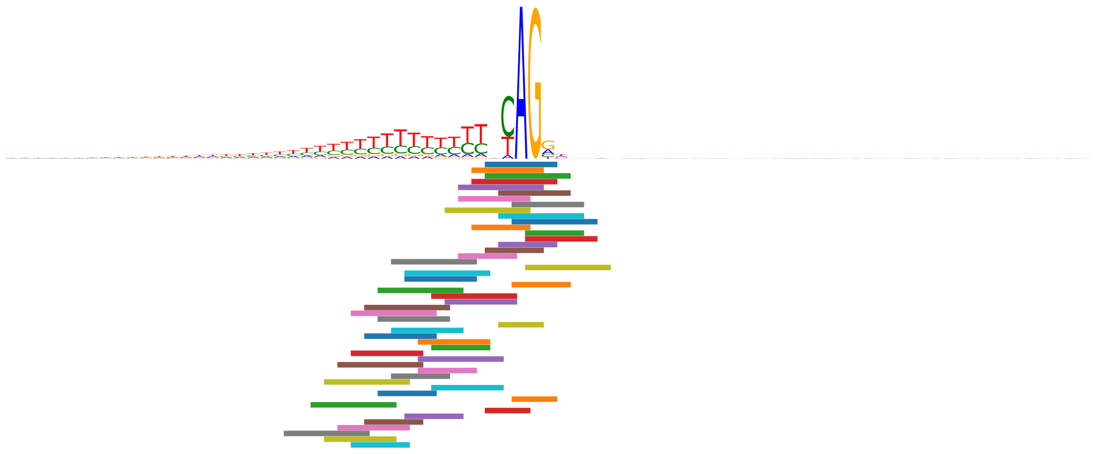


(C) 100D Vector Space of all K-Mers, reduced to 2D using t-SNE. The K-Mers that consist of only As lie very close to another, as do the Ts, Gs and most of the Cs. The TATA-Variants are close to the As. Overall, similar K-Mers are grouped together.

FIGURE 5.5 (cont.): **Continuous vector space.** 3,456 K-Mers have been mapped to vector space and visualised in three different ways. A subset of K-Mers has been annotated manually; all three visualisations have the same set of annotations.

5.4.4 Ranking of KI-Mers using Correlation Analysis

Figure 5.6 shows the top 50 KI-Mers ranked by their correlation (most significant at the top); for visual aid they are aligned to the logo visualisation obtained in section 3.3.5 on page 42. KI-Mers around the consensus site are ranked highest, followed by those in the polypyrimidine tract of acceptor sites. There are only very few KI-Mers outside of the PWM logos within the top 50 features. Many KI-Mers are overlapping for both acceptors and donors. This is due to the correlation analysis analysing each KI-Mer independently at a time and the significant overlap between KI-Mers during generation.


(A) Top 50 KI-Mers Acceptor versus Rest

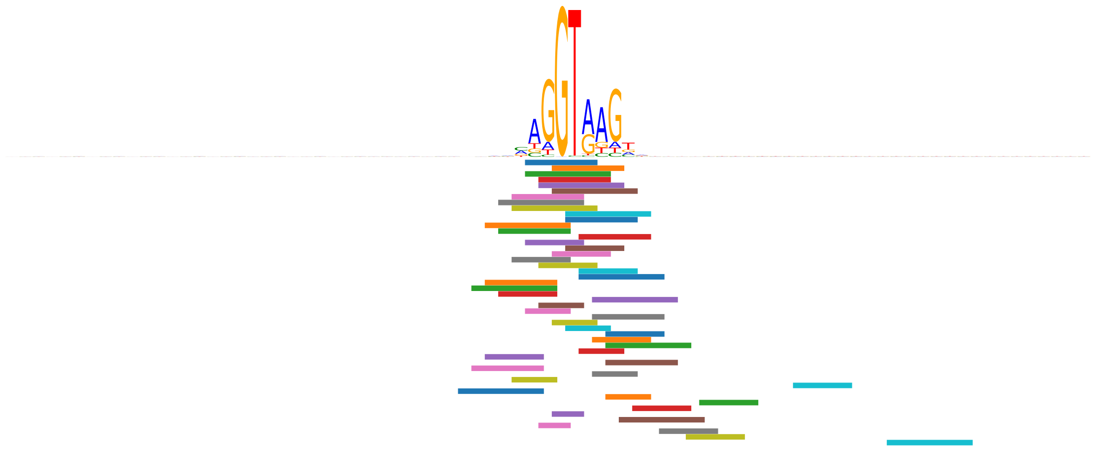

(B) Top 50 KI-Mers Donor versus Rest

FIGURE 5.6: **The top 50 KI-Mers found for the binary classification tasks acceptor/rest and donor/rest according to their individual linear correlations.** KI-Mers are sorted by their importance in descending order from top to bottom. KI-Mer colouring does not bear information. (a) The KI-Mers with highest correlation are at the consensus site with generally high Ks. There are no 2-mers found. KI-Mers with lower correlation are mostly within the polypyrimidine tract. (b) Donor sites also have the most significant KI-Mers at their consensus site, also with generally high Ks. The consensus 2-mer is ranked 45th. There are some KI-mers to the right outside of the motifs visible in the logo.

5.4.5 Ranking of KI-Mers using RFE Analysis

Before feature selection, the max-depth found for RF20 is 46 with an intermediate AUC-PR score of 93.73%. This is substantially deeper than the depth of 20 and 2.13 percentage points worse than for the one-hot encoded base experiment.

Using this architecture, the top 50 KI-Mers were ranked by their importance in descending order using RFE; the results are plotted in Figure 5.7. The two binary classification tasks were aligned to the logo visualisation obtained in section 3.6. The top 50 KI-Mers found for the acceptor classification look similar to the ones found in the correlation analysis, i.e. all KI-Mers found are within the consensus region and T/C trail. The top 50 KI-Mers for the donor class spread further out and cover regions with no known motifs, this does however not hold up in the multi-class case where all top 50 KI-Mers found are within known patterns.

Figure 5.8 shows the convergence of train and test data during RFE for the multi-class case. The train and test values are stable for a big number of KI-Mers, with the train score higher than the test score, as expected. This stability holds until about 50 KI-Mers, selecting fewer decreases the test and train scores, indicating that not enough features are being used. This strengthens the trust that the 50 KI-Mers found (Figure 5.7c) represent an appropriate selection of features.

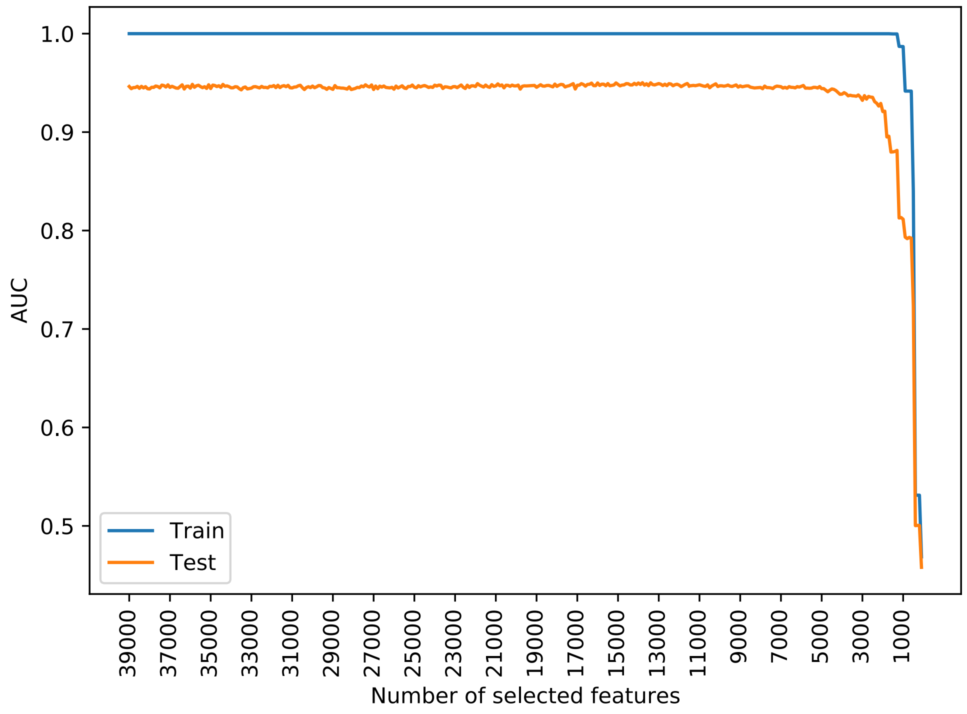
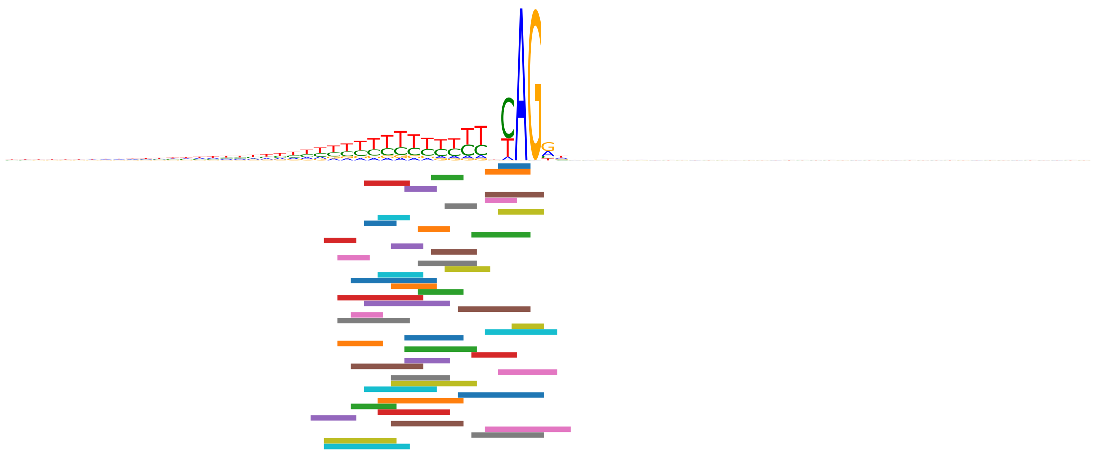
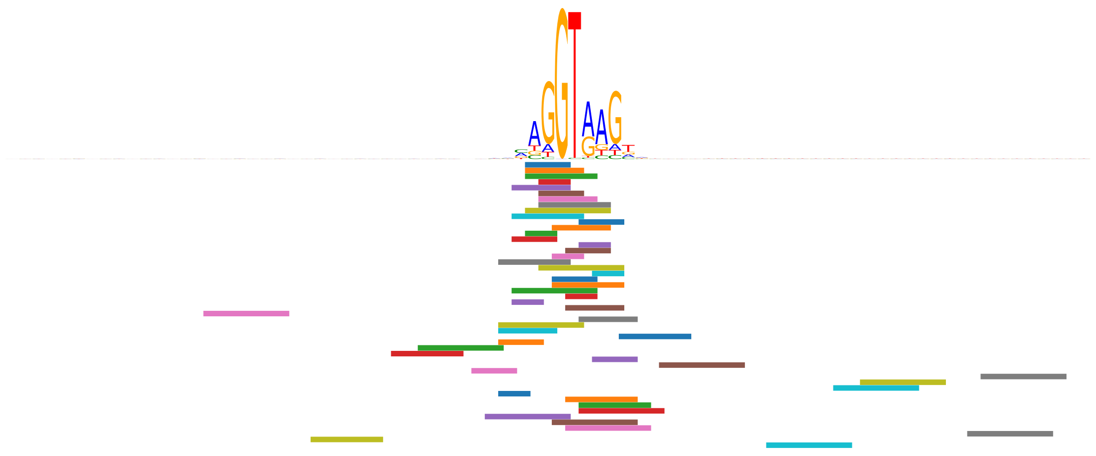


FIGURE 5.8: **Convergence of train and test AUC-PR during recursive feature elimination for the multi-class case.** 100 features correspond to one KI-Mer. The test score did not drop until after 5000 features (50 KI-Mers).


(A) Top 50 KI-Mers Acceptor versus Rest


(B) Top 50 KI-Mers Donor versus Rest

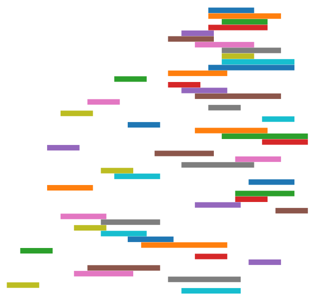

(C) Top 50 KI-Mers Donor for all three classes

FIGURE 5.7: **The top 50 KI-Mers found using RFE analysis.** KI-Mers are sorted by their importance in descending order from top to bottom. KI-Mer colouring does not bear information. (a) The G from the acceptor consensus site is not included within the top 5 KI-Mers, the features found rather focus on the preceding nucleotides. The polypyrimidine tract is covered extensively. Only three of the KI-Mers within the top 50 utilise nucleotides of the exon. (b) The main focus of KI-Mers around donor sites is the consensus site and its immediate surroundings. There are some KI-Mers lying relatively far away from the site within both exon and intron, more within the intron, but with significantly less overlap than around the site itself. (c) None of the outlying KI-Mers found for donor sites are found in the top 50 of KI-Mers for all three classes, the KI-Mers rather cover the site itself and the preceding nucleotides.

5.4.6 Prediction of Splice Sites using DNA2Vec representation

The hyper parameter and their intermediate AUC-PR is documented in table 5.3. The intermediate AUC-PRs improved for SVCs and Logit compared to when run on the one-hot encoded dataset, the scores for the remaining algorithms decreased; RBF with a polynomial kernel did not complete within time.

Algorithm	Best			AUC-PR	
	Parameters			Measure	Improvement
SVC-linear	$b = false$	$C = 0.0001$		95.62%	**+2.09**
SVC-RBF	$b = true$	$C = 10$	$\gamma = 0.0001$	96.00%	**+2.19**
Logit	$l = l_2$	$C = 0.001$		95.04%	**+2.22**
RF20	$d_m = 42$			93.76%	-2.10
MLPC	$L = (500,)$			93.92%	-4.13

TABLE 5.3: **Hyper parameters found for DNA2Vec encoded data and how much they improved compared to one-hot encoded data in percentage points.** Linear SVC, RF20, MLPC were optimised on *subset15.gencode.v33grch38.primary* and the remainder on *subset2.gencode.v33grch38.primary*. Negative improvements represent worse results.

Using these hyper parameters found, the actual CV experiment (table 5.4) revealed that the ability to recognise splice sites improved for SVCs and Logit when trained on the semantic space compared to the one-hot encoded data. The general stability of the models decreased (higher standard deviation). Judging by these results, splice site recognition by SVCs improved; whether these improvements hold up for the variant classification case is determined in the next experiment.

5.4.7 Prediction of Aberrant Splicing using DNA2Vec representation

Table 5.5 shows how base classifiers improved predicting aberrant splicing on the variant dataset when being trained on the semantically encoded data. While there are gains for three classifiers, none of the baseline algorithms can compete with the CNNs from chapter 4 with measures of 90% and more. The gain of a few percentage points on some algorithms, without making them competitive, hardly outweighs the considerably higher technical depth of introducing DNA2Vec, which will also make any potential future feature analysis harder.

Model	Acceptor and Donor		Acceptor		Donor	
	mean $\overline{m}$	std. σ	mean $\overline{m}$	std. σ	mean $\overline{m}$	std. σ
AUC-PR						
Logit	93.215%	2.636	91.039%	1.677	95.390%	1.275
RF20	93.597%	2.505	91.186%	0.919	96.007%	0.301
MLPC	93.919%	2.493	91.543%	0.700	96.295%	0.806
RF500	94.772%	1.899	93.051%	0.899	96.492%	0.699
SVC-RBF	95.600%	1.875	93.811%	0.716	97.389%	0.344
SVC-Linear	95.769%	1.835	94.020%	0.651	97.519%	0.435
AUC-PR difference to One-Hot encoded base experiment						
Logit	**+0.035**	+0.905	-0.420	+1.540	**+0.489**	+1.052
RF20	-2.177	+1.111	-3.205	+0.668	-1.149	+0.257
MLPC	-4.113	+2.013	-6.014	+0.603	-2.211	+0.795
RF500	-1.966	+0.940	-2.736	+0.739	-1.197	+0.646
SVC-RBF	**+1.524**	+0.336	**+1.258**	+0.447	**+1.789**	+0.179
SVC-Linear$^{\Delta}$	**+1.693**	+0.296	**+1.467**	+0.382	**+1.919**	+0.270

$^{\Delta}$ SVC-Linear did not complete in the base experiment. This comparison is against the SVC-RBF from the one-hot encoded experiment.

TABLE 5.4: **Cross-Validated AUC-PRs measured on recognition of splice sites trained on semantic space and how their mean and standard deviation changed compared to the One-Hot encoded data.** Non-linear SVCs and Logit were evaluated on the *subset2.gencode.v33grch38.primary*, the rest on *subset15.gencode.v33grch38.primary*. Positive values in the mean reflect improvements in predictions, negative values in the standard deviation represent predictions becoming more stable.

	AUC-PR		Optimal	Accuracy	
	Measure	Improvement	Threshold	Measure	Improvement
MLPC	78.13%	-8.07	0.799	73.86%	-5.30
Logit	81.19%	**+4.90**	0.361	77.09%	**+2.23**
RF20	82.16%	-5.62	0.350	76.84%	-4.22
SVC-RBF	82.37%	**+3.89**	0.427	77.92%	**+2.24**
SVC-Linear	82.88%	**+4.40**$^{\Delta}$	0.286	78.08%	**+2.40**$^{\Delta}$
RF500	84.08%	-5.12	0.478	78.16%	-3.4

$^{\Delta}$ SVC-Linear did not complete in the base experiment. This comparison is against the SVC-RBF from the one-hot encoded experiment.

TABLE 5.5: **Application of the baseline classifiers trained on semantic space to variant data and comparison to one-hot encoded experiment.** All were trained on *subset15.gencode.v33grch38.primary*.

5.5 Discussion

5.5.1 Incorporation of Protein Regulation for Splice Site Recognition

Protein binding sites for ESE and ISE were annotated to help baseline classifiers find splice sites, however, the results have shown that this annotation strategy did not improve splice site recognition.

It is however not entirely surprising that annotation of regulatory elements did not help ML. Abrahams et al. (2021) classified *Nonsense-associated Alternative Splicing* (NAS) using MMSplice on data annotated with *INT3* ESE motifs and concluded "that presence of ESEs at the site of the mutation is not a good predictor of which nonsense mutations do or do not induce NAS in this dataset".

The cause of why the annotation process presented in this work failed is likely due to the modelling of protein regulation being too simple; it only roughly models the underlying biology. Firstly, it does not distinguish between different protein binding motifs and assumes all ESE and all ESS motifs to be equally important. This is clearly not the case. It was also noted that protein may either inhibit or excite splicing depending on context. Both of these factors may be mitigated by annotating each protein on its own separate vector; this would however inflate the data significantly. Our annotation further allows overlapping motifs. In the real world, protein binding to the same motif prevent each other antagonistically. This model assumes protein bindings to be independent from each other and modelling protein-protein interactions is a very complex task that could be introduced into this domain in its own research context. Further, this algorithm only annotates *potential* binding sites and does not model actual protein concentrations in tissue or developmental stage. It would be very interesting to instead look at local splicing in a certain tissue and only annotate binding sites of protein that are actually present in this environment. Finally, only motifs within the 81 nucleotide windows around a splice site are given to the classifier. While regulatory protein binding close to a splice site are often more important, further interaction with regulatory elements up- and downstream were not modelled.

There are other potential reasons why this experiment did not work apart from the simplifications of the annotation process. The ESE/ESS motifs could be of low quality or inappropriate for the global way the algorithm looks at splicing. Through the design of the data pipeline, some of the models such as the RFs could also have picked up binding motifs without explicit annotation, making the information redundant and confusing. Maybe the presence of ESE/ESS motifs are not that important for finding global primary splice sites. While even canonical splice sites are often associated with regulatory sites, it is foremost alternative splicing that needs weaker splice motifs to be strengthened by protein regulation.

5.5.2 Incorporation of Conservation Scores for Splice Site Recognition

It is surprising that the incorporation of conservation scores did not improve classification as it is part of competing models. SQUIRLS (Danis et al., 2021) for example includes the mean phyloP measure of the *ref* allele region, which is included in this training data. MMSplice on the other hand (Cheng et al., 2019) reports how their model, trained on sequence data without conservation scores, outperformed HAL and SPANR which in term outperformed conservation-score based tools CADD and PhastCons in Kircher et al. (2014). They also report phyloP and CADD to have good exonic performance "but close to random in the evaluated intronic variants" (Cheng et al., 2019). As negative samples in the test dataset are sampled uniformly random, it has an even split between exonic and intronic data points.

Further, it should be emphasised that while splice sites are often conserved more highly, the dataset at hand is only respective to the human genome. Even if a significant amount of splice sites were conserved and similar to other vertebrates, evolutionary drift and selection may introduce differences in both the splicing process and its motifs and regulatory elements.

5.5.3 Application of Semantic Encoding to Splice Prediction

The semantic space was visualised and it was found, that the model learned about reverse-complemented sequences and arranged the sequences semantically sensible. The t-SNE representation of the 100 dimensional space was less confusing than the PCA plot that had a three layer architecture. Literature suggests that t-SNE is indeed more appropriate than PCA for application to Word2Vec encoded data due to its neighbour preserving feature (i.e. Komenda et al. (2016); Malmqvist et al. (2021)), so the three layer architecture in the PCA plot doesn't need to be over-interpreted.

Two methods of deriving the importance of KI-Mers were evaluated: A correlation analysis where each KI-Mer is evaluated independent of the rest, and RFE that removes only the worst one and therefore evaluates KI-Mers in relation to each other. This allows the RFE to ignore strongly correlated and overlapping features. As a result, the most important acceptor sites (Figure 5.7a) are significantly more spread out than the ones found in the correlation analysis. This difference is not that significant for donor sites (Figure 5.7b).

Donor site analysis resulted in some unexpected KI-Mers for both experiments. Correlation analysis found two intronic and RFE found at least two exonic and five intronic KI-Mers where the logo visualisation shows no obvious patterns. Initially it was assumed that this analysis found some regions with novel patterns. After analysing the multi-class case and RFE convergence (Figures 5.7c and 5.8 respectively), this assumption did not hold: The AUC-PR on test data for 50 nucleotides did not drop, indicating that the top 50 KI-Mers contain all important KI-Mers, and within these top 50 KI-Mers there are no KI-Mers spanning over nucleotides outside of the known motifs.

Unfortunately, the semantic space was ultimately found not to help baseline classifiers to predict variant effects on splicing, despite some intermediate results showing minor improvements.

5.6 Conclusion

The data pipeline for baseline classifiers was extended to include protein regulation or conservation scores. Neither of these produced better classification results.

DNA2Vec achieves sensible representation of K-Mers in vector space. Building KI-Mers for every position inflates the dataset substantially, and feature selection was able to bring the number of KI-Mers down to 50 without sacrificing model performance. Unfortunately, while some scores improved both when recognising splice sites and classifying variants, baseline classification could not compete with deep learning models; baseline classifiers seem to not be able to predict splicing based on DNA sequence and need manually engineered features as done in SQUIRLS.

Objective 3 (improve splice site recognition through data engineering) was reached on an implementational level, but the resulting algorithms were found not to be useful in clinical practice.

Chapter 6

CI-SpliceAI: Software Engineering of an Annotation Tool

6.1 Introduction

Chapter 4 found the best ML model of those investigated to be a CNN with the SpliceAI architecture, trained on the collapsed isoform dataset. It outperforms all other models, and more importantly, the original SpliceAI models. At this stage, the model consists of five files, each representing the weights of one CNN, which will hardly be of use for biomedical colleagues in this domain. In order to facilitate usage of these models for clinicians and bioinformaticians, a software package needs to be developed. This software is called CI-SpliceAI, and has been published with the journal PLOS One (Strauch et al., 2022).

For this publication, the tool should be compared against other application from this field with a focus on differences to the original SpliceAI algorithm.

6.1.1 Differences to Previous Analyses

The focus of this chapter is on software engineering of a splice disruption annotation tool to compete with SpliceAI. Therefore there are important differences to the methods from chapter 4.

While chapter 4 focused on the ML part of a binary classification, this chapter focuses on the annotation part. A good annotation tool encapsulates a ML model into a usable interface that outputs which genes are affected and how. Chapter 4 applied all ML models in a custom data pipeline, i.e. without using the SpliceAI python annotation library, to focus only on the ML part. This chapter instead develops a new annotation module and compares it to existing tools in their out-of-the-box configuration, as most people would. For SpliceAI specifically, this means that areas outside of annotated genes are masked and certain MNVs are not annotated by their module.

Overall, the research objectives in this chapter is not ”Which ML algorithm is the best?”, which was already answered in chapter 4, but ”How can we transfer the best model found into production” and ”How does it compare to competing variant annotation tools?”.

6.1.2 Aims and Objectives

The overarching aim 4 is to allow users of varying technological backgrounds to use CI-SpliceAI, and to hold all licences and copyright on the code base for both academic and commercial use. CNNs were previously trained using the published SpliceAI train code (McRae et al., 2019b) which allows academic, but not commercial use. To allow potential commercialisation, the training code is to be reimplemented.

This is to be achieved through 1) reimplementation of the ML training code so no licencing will be held by McRae et al.[1]; 2) implementation of a python package that allows variants in *Variant Call Format* (VCF) to be classified using both CPU and GPU computation, assuming the user has appropriate computational background; and 3) by implementation of a freely accessible website to run VCF files with minimal technological background.

The train and classification code are made open source to facilitate adaptation in the scientific community. The website should be freely accessible with a non-commercial clause in its terms and conditions.

To ensure availability of the website, appropriate usage limits, spam prevention, and security hardening need to be implemented, as well as analytics and tracking about its usage.

In order for a publication to find impact, the new tool needs to be compared against popular tools in the field. It was chosen to compare it with SQUIRLS (Danis et al., 2021), MMSplice (Cheng et al., 2019), MES (Yeo and Burge, 2004), and SpliceAI (McRae et al., 2019a). These tools were already described in sections 1.1.2 (page 4) and 4.2.1 (page 66).

[1]This might change as the original authors have patents pending on the SpliceAI architecture which, if accepted, might cover significant parts of this re-implementation

6.2 Background

From a user's perspective, SpliceAI consists of 5 main components: 1) The training code (McRae et al., 2019b, SpliceAI train code folder); 2) the SpliceAI command line module (McRae et al., 2019a); 3) pre-computed scores for SNVs and MNVs (McRae et al., 2019b, genome_scores_v1.3 folder); 4) the website SpliceAI lookup (TGG, 2022); and 5) an Ensembl VEP (McLaren et al., 2016) plugin (Lemos, 2019).

6.2.1 SpliceAI Training Code

The SpliceAI training process was already described in section 2.2.4.2; this process was re-implemented.

Besides a 1-to-1 reimplementation, some avenues for improvements were found. The training code uses the deprecated python 2 dependency, which was upgraded. Further it did not use the keras logging abilities which would help monitoring algorithmic convergence. The original training code did not include the data pipeline to derive the splicing tables; due to the nature of the modifications to CI-SpliceAI, this needs to be changed so that the codebase encapsulates all steps to produce new models. Lastly, the original SpliceAI model is trained on only 19 chromosomes, excluding the test fold of chromosomes 1,3,5,7, and 9. For clinical application, it would be desirable to instead train on all chromosomes.

The GENCODE version can further be updated to GRCh38.v37, which at the time of writing is the latest version.

6.2.2 SpliceAI Python Annotation Module

The SpliceAI command line module is a python module published to *Python Package Index* (PyPI) (PackagingWG, 2018). A python module allows installation via the *pip* command, which gives access to its functionality to both python code and the command line. Publication to PyPI allows anybody to install this module.

The module comes with two splicing tables for GRCh37 and GRCh38 respectively. These tables lists gene symbols, their primary transcript and matching start and stop coordinates, and all splice sites contained within this isoform. Users can choose which table to use or even reference one of their own.

The command line tool accepts input files as VCF and annotates them in the same format. Internally, the module extracts the reference sequence from the reference genome in a window of configurable size (up to 4,999 nucleotides on either side) around the position of interest plus a fixed 5,000 nucleotides of context on either side. Gene annotations are found by filtering the splicing table to overlapping primary isoforms; if more than one gene is found, the first is chosen; if none overlap the algorithm does not annotate the variant. All nucleotides outside of this primary isoform are encoded as unknown ("N"). The variant sequence is derived by applying the variant, generating a second input. The five CNN models are run on both one-hot encoded sequences, the δ-score matrix is calculated, and the position and score of the most significant losses and gains for acceptor and donor sites are returned in the output annotation file. Details on how this is done exactly and how to cope with insertions and deletions is described in section 4.3.2 (page 73). Variants where both reference and variant sequences are longer than one nucleotide are not annotated; the reason for this is unclear.

The python module can run on either CPU or GPU (on *Compute Unified Device Architecture* (CUDA) supported graphics card); users however must set up CUDA themselves which is a non-trivial task and requires expertise.

Input annotations are accepted for both builds, GRCh37 and GRCh38. This is done by including splice sites for both builds, and requires the user to also point the algorithm to a matching reference genome. Lastly, the module allows *masking* scores to only include gains in splicing for non-canonical sites and losses of splicing for canonical sites. This reflects clinical practice where losses of non-splice sites and gains of known splice sites are sometimes prioritised down.

There are some possible technical improvements on this. If the sequence extracted overlaps with more than one gene, the better strategy would be to run predictions on all genes and either return all effects, or only the most significant one. If there are no annotated genes within the sequence, the algorithm should run on the forward and backward sequence without any masking of regions outside of the gene, and again either return both or only the more significant effect. The masking of scores to only account for losses of canonical and gains of non-canonical sites is also implemented suboptimally: Instead of masking the δ-score matrix before the final annotation is completed, which would allow the most significant disruption that is not masked to be returned, the implementation first calculates the most significant changes and then masks it, which will not return the next-best score.

6.2.3 Pre-Computed Splice Variant Scores

The authors also pre-computed scores for 1 base insertions and 1-4 base deletions within all genes for both masked and unmasked scores on GRCh37. Scores were also lifted to GRCh38 and published. All files together make up over 364GB of gzipped data.

These pre-computed scores are in VCF format and serve as a simple lookup table, so that the same variants do not need to be re-computed repeatedly. They were calculated for a window size of 50 bases around a splice site, which was the value recommended in the original publication (Jaganathan et al., 2019).

Their python module today defaults to a bigger window size of 500 nucleotides around the variant, rendering the pre-computed scores of limited use. Also the lifting process might introduce problems when using GRCh38. Lastly, the pre-computed scores cannot cover all potential variants, so tools using this lookup table cannot perform as well as applications running SpliceAI from scratch.

6.2.4 SpliceAI Online Lookup

The SpliceAI Lookup website, accessible at spliceailookup.broadinstitute.org, was not released with the original SpliceAI publication, instead is was developed and is maintained by the Broad Institute. It is open sourced on github (TGG, 2022).

It allows submission of one variant at a time, batch processing is not supported. Variants can be looked up in the pre-computed scores discussed earlier, or are calculated in real time. Builds GRCh38 and GRCh37 (through lifting) are supported, as well as optional masking of scores and configuration of the maximum distance. To prevent excessive or malicious use, users are only allowed to calculate 4 variants and lookup a total of 15 variants per minute.

Under the hood, this website uses the SpliceAI pre-computed scores and python module, with all of their advantages and disadvantages. Results are cached in a redis database to prevent re-calculation of the same input. The website itself is hosted on a python flask server; the server generating ML predictions is therefore identical with the one hosting the website.

Using this architecture has some drawbacks. If the ML backend crashes due to overload, the website serving cached results will go offline too. All unseen predictions are created ad-hoc, there is no queueing mechanism, which is not scalable and limits its use. It therefore cannot provide batch inference. From a user perspective, entering one variant at a time is impractical. On the other hand, the website is easy to use and greatly facilitates usage of the SpliceAI models without any technical background.

6.2.5 Splicing VEP Plugins

The SpliceAI VEP plugin is a lookup adapter to find variants from the precomputed scores. It is therefore limited to the drawbacks of this collection.

In contrast, the MMSplice VEP plugin (Hasan, 2018) requires the MMSplice python module to be installed. It works as an adapter between the VEP framework (programmed in perl) and the python backend, and runs predictions in real time. This however requires the user to set up the python tool anyway, at which point they should be technologically capable of running it without VEP. This plugin is also not offered by the official VEP web interface, as they seem to not run any third-party applications outside of their perl framework.

Lastly, one could in theory write a VEP plugin which sends each variant to the SpliceAI lookup website. This however would violate their scope; the lookup website must not be queried programmatically.

6.3 Methods

6.3.1 Training Data Pipeline Reimplementation

The main data source was upgraded to *gencode.v37GRCh38* (in this chapter only referred to as *collapsed* data) and re-implemented using multi-processing to significantly speed up both generation of the splicing table and the ML dataset itself.

The splicing table was implemented to contain collapsed splice sites as described in section 3.3.2 (page 41). Sites annotated as both acceptor and donor sites (i.e. chr7:44122282) were filtered out to improve data quality.

The splice annotation table has two functions. First it is used to train the ML model. Second, it is used in the variant annotation module (see section 6.3.3). To allow both builds, GRCh38 and GRCh37, the splice annotation table was created based on both builds without lifting.

6.3.2 Training and Testing Methodology on Splice Sites

The SpliceAI training code (Jaganathan et al., 2019) was reimplemented and improved on.

The batch size during training was increased from 6 to 32 slices per GPU to speed up training on the IRIDIS system. When a gene has an uneven number of slices and therefore cannot be distributed on the two GPUs, SpliceAI drops the last slice. The new implementation instead drops a random slice to not bias against the end of a gene.

While the final model (i.e. the one to be published) is trained on all chromosomes, an additional model was trained only on the same subset of chromosomes that SpliceAI was trained on. This was only done to facilitate fair comparisons between the two

SpliceAI models. This secondary model is tested on the remainder of chromosomes (1,3,5,7 and 9).

6.3.3 Design of a Python Variant Annotation Module

The annotation module is the main piece a user interacts with to predict the effect of a variant on splicing. It uses the splice site annotation table described in section 6.3.1. From a technological standpoint it is a python module with a command line interface as its primary touchpoint.

The general interface of the SpliceAI python module (McRae et al., 2019a) was imitated to ease adaptation, but the whole module code was newly developed and improved on.

Users can input VCF variants and results are annotated in an output VCF file. Many of the general processes are the same as with SpliceAI, for example it supports both GRCh37 and GRCh38 and it allows configuration of the maximum distance from the variant.

When more than one gene overlaps with the variant, it does not choose the first gene as SpliceAI does, instead it runs on all overlapping genes. Similar to SpliceAI, the strandedness of gene annotations in the splicing table affects whether the sequence is reverse complemented, and regions outside of gene annotations are masked as 'N'. This last detail can be disabled when running the tool ("–outside" flag). Additionally, when the extracted DNA sequence from the reference genome overlaps with no gene, it runs on both the forward and backward strand instead. The annotation therefore documents how a *region* is affected, which can either be a gene or the forward or backward strand if no genes were found in the vicinity.

By default, only the annotation of the most significant region is returned; this can be changed to include all annotations with the "–all" flag.

The optional masking of splice sites to only contain gains of novel and losses of existing sites has been improved by masking the δ-score matrix rather than the final output, allowing the next-highest annotation to be returned rather than capping the output to zero.

The CI-SpliceAI module also allows configuration of the batch size. This is important when running the model on a GPU - bigger batch sizes improve speed through parallel computation, whereas batch sizes that are too big cause the GPU to run out of memory and crash the application. The ideal batch size is subject to the memory capabilities of each user, therefore it makes sense to give users a parameter to tune it to their needs. As the max-distance from the variant and the length of reference/alt sequences directly affects the size of each single variant, it would not make sense to use a classical batch size that quantifies the number of samples. A batch of 100 samples of each 50 nucleotides each side of the variant (plus context) is vastly smaller than a batch of 100 samples with 5,000 nucleotides of either side (plus context). It was therefore decided to define the batch size in MB so that this dependency is accounted for. The application therefore aggregates variants until the next variant would accumulate more input than the batch size, processes the full batch, and starts a new one. On the CPU, the batch size does not really bring a merit.

Lastly, the CI-SpliceAI module was also designed in such a way that it can be used and extended from python code. This made it easier to extend on this module for the next components.

6.3.4 Extending the Splice Variant Dataset

The variant dataset described in chapter 4 was further enlarged with two recent sources following the methods outlined to include even more variant data. No further disputes between sources were found.

6.3.4.1 Incorporating Ellingford et al.

Table 1 of Ellingford et al. (2019) published 21 variants and their functional assessment of splice disruption which have been extracted manually. Splicing defects were assessed through RNA expression in lymphoblast cell cultures, whole blood, and cell based minigene assays.

6.3.4.2 Incorporating MutSpliceDB

MutSpliceDB (Palmisano et al., 2021) is a freely accessible genome database consisting of, at the time of writing, 86 variants, their RNA sequences, and their effects on splicing. All variants are disruptive. Variants were exported using their web interface.

6.3.5 Annotation of the Exact Variant Effect

To evaluate if ML algorithms can predict the exact position and variant effect, i.e. at which basepair an acceptor/donor is lost/gained, the variant dataset was annotated with four columns. This was only possible when the source commented on the exact effect: For example, if a variant is annotated with ”Exon 2 skipped”, by looking up the transcript and its second exon, the precise position of the acceptor and donor loss can be derived.

The annotation process was done manually. During this process it was ensured that 1) annotations are not contradicting each other, 2) both 1-based and 0-based coordinates are supported, 3) annotations that are obviously referring to another site were corrected, 4) inconsistencies of what constitutes ”upstream” and ”downstream” in Wai et al. (2020) was resolved, and 5) exon skipping of the first and last exon of the transcript was excluded, as it can’t be modelled as an acceptor/donor loss. During this process it was found that intron inclusion refers to partial intron inclusion most if not all of the time, however no source specified up to where introns were included. All intron inclusion annotations were therefore removed from the exact variant effect.

Annotations of splice site losses were then checked against GENCODE computationally to ensure that there indeed is an acceptor or donor site present. Motifs of novel junctions were checked for sensible consensus sites.

6.3.6 Comparison to Competing Splice Variant Annotation Software

Five algorithms (CI-SpliceAI, SpliceAI, MES, SQUIRLS, and MMSplice) were tested on the extended variant dataset to measure how well they can predict if variants affect splicing or not.

In contrast to chapter 4, SpliceAI and CI-SpliceAI were run using their respective VCF annotation modules rather than a custom implementation running predictions manually. This represents the way the vast majority of users run the models. As a consequence, regions outside of genes were masked as N and SpliceAI will omit predictions of indels and substitutions with more than one nucleotide in REF and VAR annotations.

SQUIRLS was run directly on all data in VCF format as described in Danis et al. (2021).

Two MMSplice (Cheng et al., 2019) models, *splicing efficiency* and *pathogenicity*, were run through *kipoi* (Avsec et al., 2019), a python manager for genomic models. As described on the kipoi guide, the VCF file was first normalised and left aligned.

MES (Yeo and Burge, 2004) was run twice: 1) As a Ensembl VEP (McLaren et al., 2016) plugin (Shamsani et al., 2019) in a docker (Docker, 2013) container. This plugin checks for splice losses of canonical donor or acceptor sites 9 or 23 bases away from the variant respectively and calculates a reference and variant score. This implementation cannot predict novel or cryptic sites. And 2) as a custom implementation of a sliding window, similar to what was done in Danis et al. (2021), where both acceptor and donor predictors were moved over the variant, resulting in a δ-matrix of acceptor and donor predictions. This δ-matrix can then be compensated for indels and interpreted in the same way as deep learning models (see section 4.3.2 on page 73). The second implementation therefore allows recognition of novel splice sites.

Further, it was evaluated how well algorithms can predict the exact variant effect on the mature mRNA, based on the subset of variants described in section 6.3.5.

The exact variant effect was extracted from all algorithms with a per-nucleotide δ-score for acceptors and donors, i.e. SpliceAI, CI-SpliceAI, and the sliding window MES. The position of the most significant loss or gain in predictions for acceptors and donors is compared to the annotated ground truth. An algorithm with 50% accuracy predicting acceptor losses would therefore have the biggest drop in its acceptor prediction score at the annotated position across half of the variants annotated as acceptor loss.

6.3.7 Generation of Pre-Computed Scores

Similar to the pre-computed variant scores published for SpliceAI, it was evaluated if generation of pre-computed scores for CI-SpliceAI was computationally feasible.

To measure throughput, a program to systematically generate and annotate variants was developed (Figure 6.1). ML input (with 5,000 nucleotides of context) is generated and added to an asynchronous multi-processor queue. Four processes, each assigned to feed one GTX1080 TI, pull from this queue to process and annotate a variant at a time. Annotations are pushed into another asynchronous queue for a process to write the results to a VCF file. This decoupled design allows to scale up very easily as new processes to generate, process or write data can be added as needed.

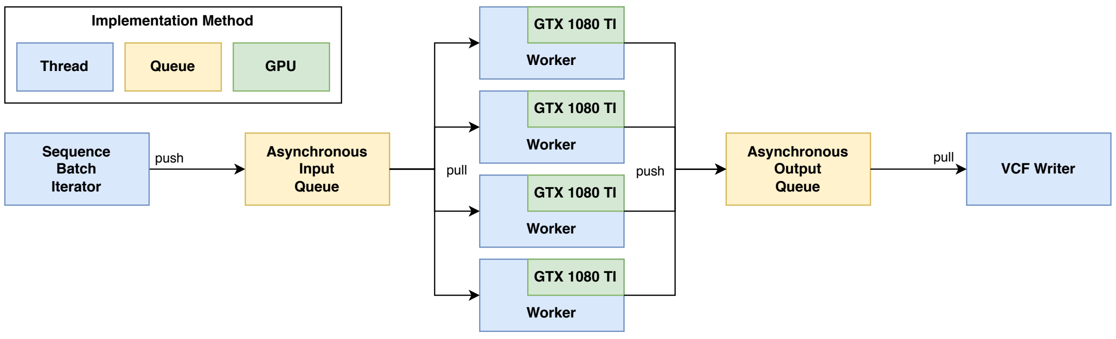


FIGURE 6.1: **Code architecture to pre-compute scores.** The generation of sequences, annotation, and output writing is decoupled to allow maximum throughput. Predictions are calculated on a GPU with its own worker thread.

6.3.8 Design of CI-SpliceAI Web for Online Variant Annotation

The intention behind the CI-SpliceAI Web application is to facilitate usage of CI-SpliceAI without technological knowledge, and to make the tool accessible for academic and non-profit use free of charge.

As described in section 6.2.4, the software design of the SpliceAI lookup tool ties the ML server predicting variants to the web server the user interacts with. It was decided to de-couple these two components, allowing a more resilient architecture. Since the CI-SpliceAI python module allows batch computation on the GPU, it would also make no sense to only allow one input at a time. Instead, the CI-SpliceAI website allows input of VCF files containing hundreds of variants. In term, it needs a queueing mechanism to allow these computations. Another important difference between the SpliceAI lookup and this new site is that the CI-SpliceAI website does not lift genomic data and instead treats variants on GRCh37 completely separate to variants on GRCh38, limiting issues related to lifting but in term requiring more computation in the long run.

Instead of buying dedicated servers to calculate ML predictions, it was decided to use cloud computing resources to batch process predictions. Microsoft Azure (Mund, 2015) and Google Cloud services (Bisong, 2019) were evaluated. Due to both technical and budgetary constraints, it was decided to use Google Cloud services as it was more lightweight and includes a free tier that was deemed sufficient for this project.

6.3.8.1 Tech Stack

The main backend is implemented in PHP (Welling and Thomson, 2003). HTML templates are generated with the PHP templating engine Twig (Ronacher and Potencier, 2019). Styling is mostly done through Bulma (Aubry, 2008) with some minor adjustments in plain CSS. The logic to reload estimates for newly enqueued variants is programmed in plain JavaScript (Flanagan, 2006).

The database is implemented in mySQL (Welling and Thomson, 2003). Tracking is implemented using Matomo (Aubry, 2008), a PHP tracking library that was built around privacy. Matomo and Twig are installed via composer (Adermann and Boggiano, 2012).

Workers on Google Cloud are encapsulated in a docker (Docker, 2013) container that starts a python web service.

6.3.8.2 System Architecture

Figure 6.2 shows the CI-SpliceAI architecture. The user interacts with a frontend made of regular web technologies (HTML, CSS, JavaScript). The frontend allows submission of VCF files which are parsed by the VCF backend. The VCF backend looks up all variants in a cache, which is a relational mySQL database. It immediately returns all previously annotated variants and enqueues new variants to the cache. It further calculates an estimation of when all variants should be calculated by interpolating processing times from the past. This estimate is returned to the user; the frontend refreshes periodically and after the expected time estimate to allow partial analysis if desired.

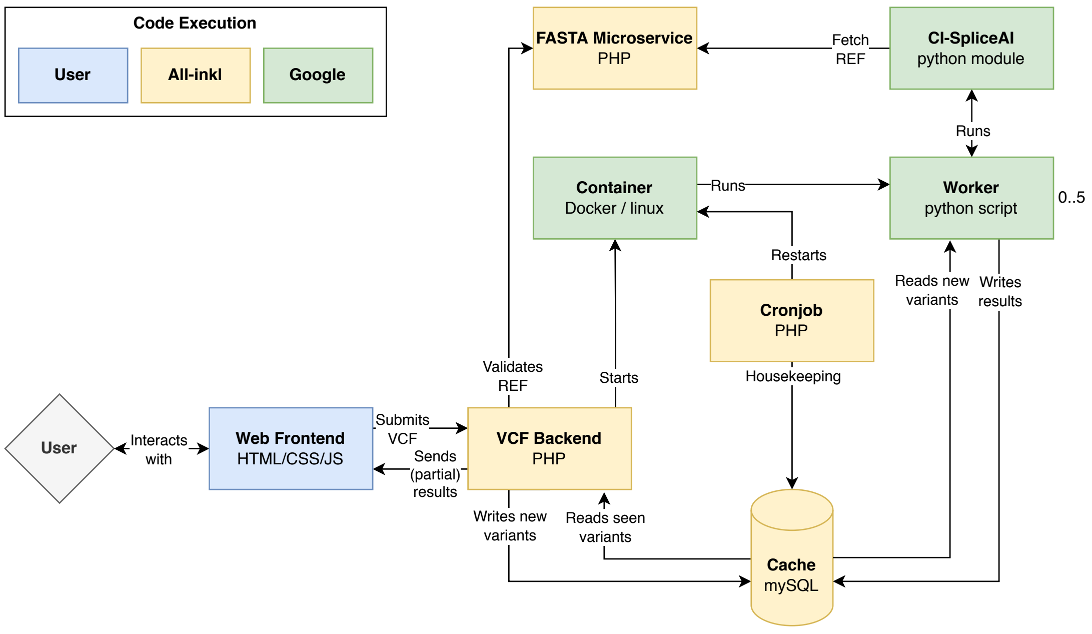


FIGURE 6.2: **CI-SpliceAI Web architecture and where each component is executed.**

The backend then determines the number of new workers to be started, and fires them up on the Google Cloud. Each worker processes one variant at a time by querying them from the database, running the CI-SpliceAI python annotation module, and writing the results back to the cache. Partial results are therefore immediately available to the user if they decide to re-submit. Due to constraints on the Google Cloud, workers do not have their own copy of the reference genome to extract DNA from. Instead, they contact a FASTA microservice which extracts and returns the sequence for them. Google workers run for up to ten minutes or until all variants are processed, and their running status is kept in a database.

Lastly, there are *cronjobs* (code scheduled to be executed regularly) in place. They monitor if workers hang, i.e. do not start or do not end in their expected time. If jobs do not start this is most likely due to budgetary constraints and a message will be shown to the user. If jobs did not exit regularly, variants marked as being processed by this worker need to be unlocked to allow new workers to annotate them.

Apart from the web frontend which obviously runs on the device of the user, there are two service providers involved. All-Inkl (2000) is the service provider hosting all backend and database services except for ML predictions. These in turn are hosted with Google Cloud services.

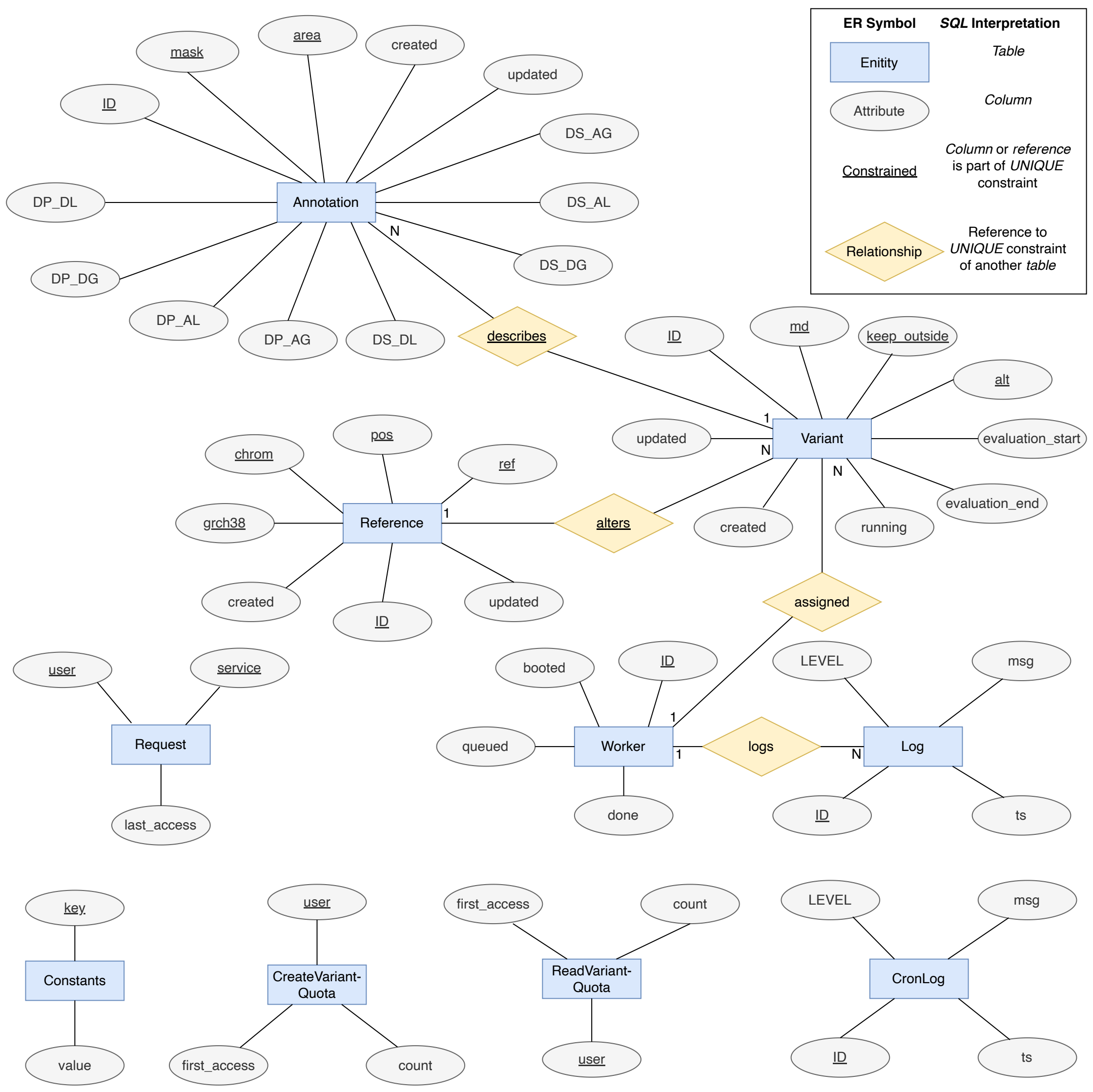


FIGURE 6.3: **Entity-Relationship diagram of the database design of the CI-SpliceAI Web architecture.**

6.3.8.3 Database Design

Figure 6.3 illustrates the database design of the CI-SpliceAI Web application as an entity-relationship diagram (Song and Froehlich, 1994) . It depicts database tables as entities and their columns as attributes. The joined set of underlined attributes and relationships symbolise the identifying unique constraints on a table, except for *ID* which itself is a numerical unique identifier. Relationships allow a column to reference an *ID* field of another table.

The *Reference* table holds the genomic coordinates (*chrom*, *pos*, *ref sequence*, and the boolean *grch38*) which together build the unique definition of a reference.

A *Variant* alters a *Reference* annotation and describes what the reference changes to (*alt*). There can be a number of *Variant* annotations for one *Reference*, but only one *Reference* per *Variant*. One *Variant* specification encapsulates the input to the ML algorithm and therefore also specifies the maximum distance from the variant (*md*) and whether to keep the nucleotides outside of annotated regions (*keep_outside*). Status flags and time stamps help monitor progress and estimate completion.

*Variant*s are *assigned* to a *Worker*; one *Worker* can be assigned multiple *Variant*s but not the other way round. Because workers are started and run asynchronously on Google, they are *queued*, then *booted* and finally *done*; these three fields are timestamps to allow monitoring of jobs. When working, they will update the *evaluation_start* and *evaluation_end* timestamps for each *Variant* accordingly.

When a worker finishes its ML process, it adds *Annotation*s to describe a *Variant*. Because a single *Variant* can affect more than one *area* (i.e. genes), and the δ-score may be *mask*ed, one *Variant* is mapped to multiple *Annotation*s. The *Worker* always enqueues two *Annotation*s per *region*; one *masked* and one without *mask*ing as there is no significant computational overhead once the δ-score is calculated. The *Annotation* then includes all acceptor/donor gains/losses and their position in the usual SpliceAI format. The columns *created* and *updated* are timestamp fields for estimation of time until new variants are calculated.

A *Worker* also *logs* to database by enqueueing *Log* messages (*msg*) at a certain timestamp (*ts*) and log *LEVEL* enum type. A very similar *CronLog* provides almost the same structure bar the *Worker* relationship to allow cronjobs to log info as well.

A simple *key/value* table for *Constants* currently only holds one constant for budget control (see section 6.3.8.7 on page 124).

Lastly, there are three tables to hold user quotas and limits. All three track a *user*'s anonymised IP address. The *Request* table holds the timestamp of when a *user* last accessed a *Service* to prevent malicious request spamming. The tables *CreateVariantQuota* and *ReadVariantQuota* track the number of newly enqueued variants and the number of variants read from the cache respectively in their *count* column. They also track the *first access* as quotas are within a window started from the first request. Quotas and limits are further described in section section 6.3.8.7.

6.3.8.4 VCF Backend

The VCF backend was custom implemented in PHP. It parses an uploaded VCF format of an untrusted source, validates it, and parses it into a structured representation.

It validates the general format, like number of columns, nucleotides consisting of only A,C,G and T, and some limit-related constraints. Any constraint violations are reported back to the user as an error. It also validates the reference sequence by comparing it to the output of the FASTA microservice (which is discussed later in section 6.3.8.6), preventing faulty variant definitions to reach the database and Google Cloud.

The VCF backend also allows configuration of all parameters, such as build (GRCh37 / GRCh38), maximum distance from variant, masking of nucleotides outside of the annotated transcript, filtering to only the most significant offset, and masking of scores towards splicing gains in non-splice sites and loss in splicing of known sites.

After input validation, new variants are enqueued, and variant annotations are returned if possible. The backend will look up all missing annotations and returns an estimate of when the last variant in the batch will be returned. The estimate is calculated by taking the mean annotation time of the past 200 variants times the number of variants in queue before the batch is completed, and dividing it by the appropriate number of workers.

The current configuration allows up to 5 Google workers to run at a time, and starts one worker per 25 variants in the queue. If there are less than the desired number of workers, new workers are started by calling the URL of the Google Cloud service instance.

6.3.8.5 Workers

Workers can be started by calling an URL provided by Google. Additionally, workers were secured with a password to prevent invocation outside of the intended use.

Each worker processes one variant at a time in a first-in first-out queue. Variants are marked in the database so that concurrent workers will not evaluate the same variant. Workers will monitor their run and current evaluation time and shut themselves off early to not overrun the 10 minute mark. It is important to shut down gracefully, if they were to execute for longer than 10 minutes, Google will force a shut down, leaving the currently evaluated variant locked to the ghost worker. This will be caught be a cronjob eventually to prevent indefinitely locked variants.

Currently, the workers are using CPU predictions for ML, as GPU predictions ar enot supported by the free Google Cloud services tier. The design however allows the workers to be adapted to GPU processing quickly; in that case batch sizes of more than one variant can be fetched from the cache.

Each variant is annotated through the CI-SpliceAI python module. The module is always configured to annotate all splice sites, not only the most significant one, and both the δ-scores with and without masking are put into the database. This will prevent the same ML inputs to be run multiple times.

6.3.8.6 FASTA Microservice

The Human Reference Genome for both builds is too big to deploy to the Google Cloud instance directly. Instead, access to FASTA is provided as a REST microservice on the PHP server that also provides the main application.

The microservice was custom implemented in PHP. The service can be queried with genomic coordinates, the desired length and a password and returns the DNA sequence. Because there are no PHP implementations to read FASTA files, the service was implemented without any third party dependencies from the ground up.

The Human Reference Genome FASTA files (Church et al., 2011; Schneider et al., 2017) contain all chromosomes sequentially. Each line has a fixed maximum length plus a carriage return. By indexing the FASTA files first, the start and stop of each chromosome in bytes and their line lengths with and without carriage return can be obtained, allowing to calculate the file seek as follows:

Let the start of the desired chromosome be c and the length of each line with and without line breaks l_b and l respectively (all in bytes), and let the desired position within the chromosome (i.e. start or stop of the region to output) be o. Then the file seek s in bytes is calculated as $s = c + \lfloor \frac{o}{l} \rfloor * l_b + o \bmod l$.

With this logic, the FASTA service can extract DNA very efficiently. It is used to validate reference annotations during variant creation and to return full DNA slices to the Google worker.

6.3.8.7 Limits and Quotas

To prevent excessive and malicious use, user-specific quotas and limits were introduced. These may be subject to change depending on usage and are not directly communicated to the user.

First, VCF submission is limited to once every 10 seconds to prevent malicious spam. Any submission outside of this limit is delayed accordingly. A VCF file may not have more then 2000 variants; bigger submissions are rejected with an appropriate warning.

A user may only submit 1,500 new variants and query only 10,000 variants within 24 hours, starting with the first submission or query. As submission of a new variant requires a read query to establish its novelty first, a submission will increment both quotas. If a quota would be exceeded, a partial result will be returned with an appropriate message.

There is also a quota on Google. The free tier allows 2 million service requests, 360,000 GB-seconds of memory, 180,000 vCPU-seconds of compute time, and 1 GB network egress per month. Anything exceeding these quotas is charged to my credit card every month. To limit spending, a Google budget alert was set up. The alert is connected to a *pub/sub* topic which is then subscribed to call the PHP backend, effectively calling the application every time the budget changes. If the budget is exceeded, the backend writes the date of when the budget replenishes (i.e. the first of the following month) to the database. This causes all currently running Google workers to stop working after the current variant and stops the system from queuing up new workers. It also displays a message to the user when unseen variants are enqueued. As new variants are still added to the database, they will be processed in the next month automatically through the cronjob system. Since going live, the system cost me a few pence per month. A better way with no cost could not be found as Google does not support automatically cutting its services when the free tier is exceeded.

The number of Docker containers on the Google Cloud is managed by Google and scaled up or down according to usage. Currently there are up to five containers configured to run in parallel, one per request.

6.3.8.8 Security and Data Protection

All user-provided input is sanitised to prevent injection of malicious code. Frontend code generated via twig automatically escapes all data to prevent code injections. Database queries are all formulated with placeholders to prevent database injections.

All services that are not supposed to be accessed by a user (i.e. FASTA microservice, Google service invocation URL) are password protected.

As opposed to other popular tracking tools, Matomo is entirely self hosted so no data is processed by third parties. Further all cookie-based tracking was disabled so that no cookie layer was needed. A page with the terms of service to limit the use to academic and non-profit use only was added, also talking about what data is logged.

No identifying data is logged. IP addresses are anonymised by removing the last three digits both in the quota calculations and in the matomo tracking.

6.3.9 VEP Splice Disruption Plugin Development

The SpliceAI VEP plugin uses pre-computed scores to look them up offline. If pre-computed scores were feasible to compute, this would be the desired method. However, as will be described in section 6.4.5, calculation of pre-computed scores is infeasible.

Running predictions in real time on the computer as MMSplice does is not an optimal solution. Installation of the required python framework is non-trivial; and doing so would render the plugin useless as the main annotation logic can be run instead without the need for the VEP plugin.

The last option would be to send variants from the plugin to the website, re-using already existing tooling. However, unseen variants could potentially take days to compute, which is not what VEP is designed for; it expects real-time computation.

CI-SpliceAI was therefore not released as a VEP plugin.

6.4 Results

6.4.1 Overview of the Aggregated Splice Variant Dataset

The curated dataset contains 1,316 functionally validated variants, 388 of which have their exact effect on the mature mRNA annotated with basepair resolution. Figures 6.4 and 6.5 visualise the data and its biases.

Figure 6.4 shows that there is almost a 50/50 split of strandedness and between affecting vs non-affecting variants. 8% of variants are published in multiple sources, all with consensus. Most variants are SNVs, and most of MNVs are deletions. The data is skewed toward certain chromosomes, mostly due to many sources investigating breast cancer.

The variants are distributed across splice sites (Figure 6.5). As expected, variants nearer to a splice site are more frequently disruptive. The dataset includes some deep intronic variants that affect splicing and variants near splice sites that do not, both of which are challenging to predict.

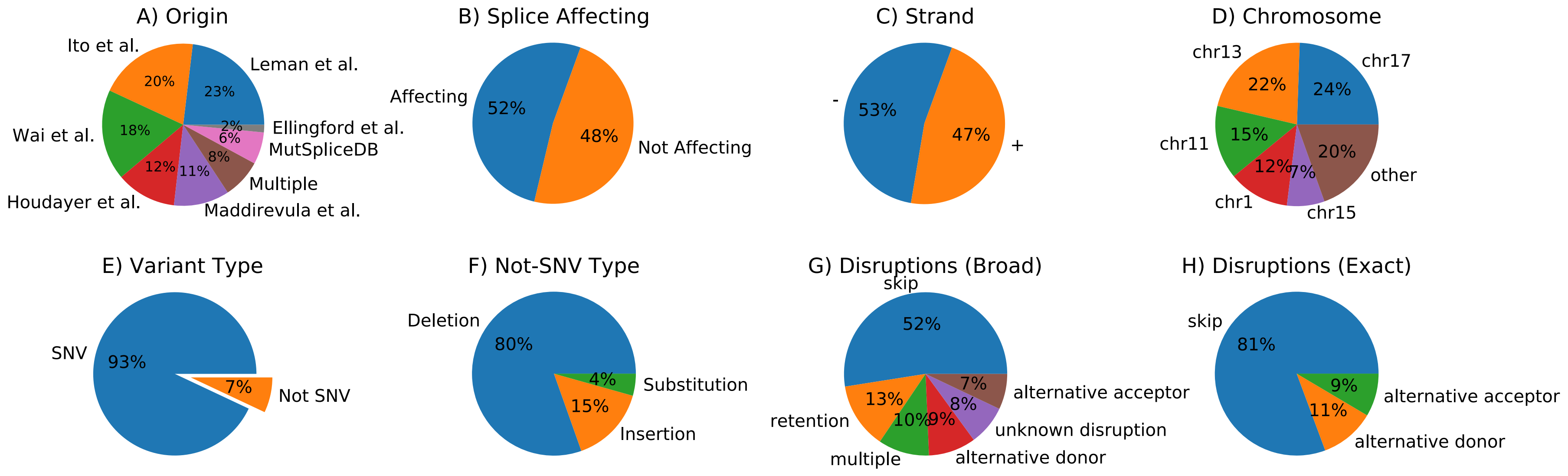


FIGURE 6.4: **The final variant data set consists of 1,316 unique variants.** (a) There is an 8% overlap, mainly due to Houdayer et al. (2012) citing Leman et al. (2018), all with consensus. (b) The data set has an almost 50/50 split between affecting/not affecting and (c) strandedness. (d) Chromosomes 1, 11, 13, 15, and 17 make up 80% of the variant data set. This could produce biased results. (e) The vast majority of variants are SNVs; The 7% MNVs are biased towards deletions (f). (g) The 1,316 variants and their broad effect on the mature mRNA is a superset of (h) where, for 388 variants, the exact position of the skipped acceptor or donor and alternative sites are known with base-pair resolution.

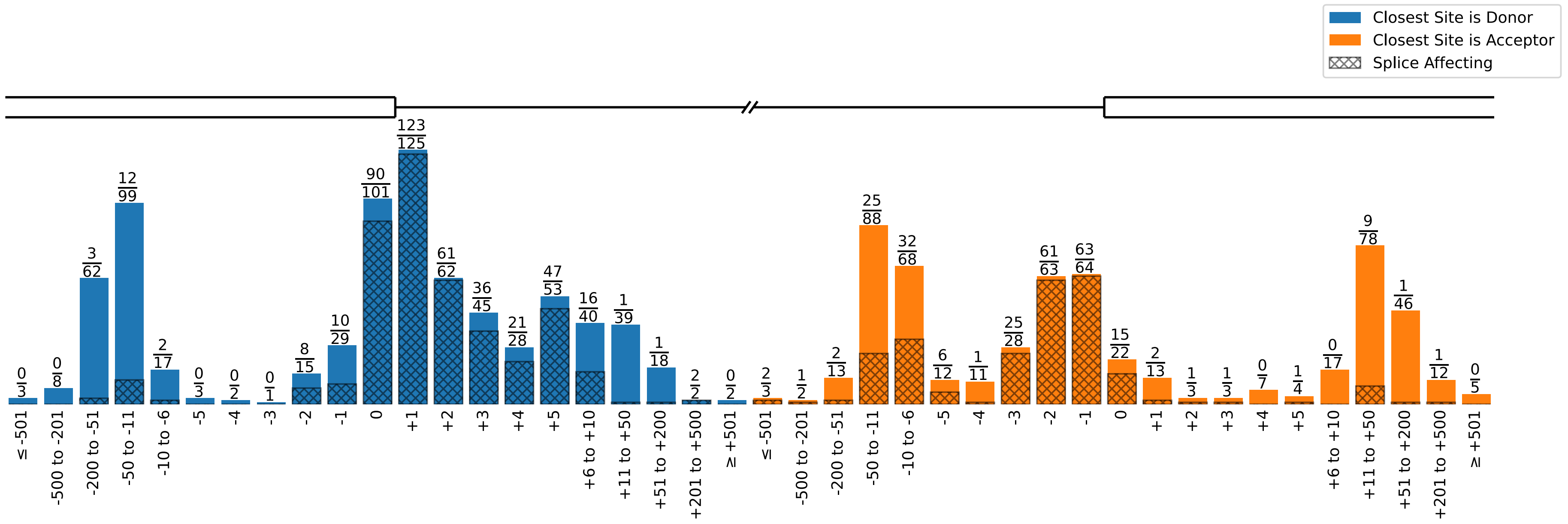


FIGURE 6.5: **All 1,316 variants in relation to their closest splice site.** Their binary effect on splicing is indicated by the shaded area and numerator in the fraction. 57% of sites are closer to a donor than an acceptor site. 23% of variants are in the consensus motif (equally split between acceptors and donors); 98% of variants within consensus regions are splice affecting.

6.4.2 Comparison of Training Data to SpliceAI

The collapsed isoform training set contains more splice sites, genes, and chromosomes than the original SpliceAI training data (Table 6.1). When filtered to the same chromosomes, the number of genes and transcripts in CI-SpliceAI is smaller due to the newer GENCODE version containing fewer transcripts of low quality. The SpliceAI training data is slightly skewed towards acceptor sites and the new one is slightly skewed the other way.

	No. Chroms	No. Genes	No. Splice Sites	Proportion Acceptor/Donor
SpliceAI (Train)	19	13,385	391,515	2.1% more acceptor sites
CI-SpliceAI (Filtered)	19	13,240	301,835	3.5% more donor sites
CI-SpliceAI (All)	24	18,580	428,475	3.4% more donor sites

TABLE 6.1: **Numeric comparison between the original SpliceAI training set and the novel collapsed dataset.** The novel collapsed training dataset used for training the final model (bottom row) includes all chromosome, more genes, and more splice sites than SpliceAI. For comparison only, if filtered to the same chromosomes as SpliceAI (middle row), the collapsed dataset would have slightly fewer genes and splice sites than SpliceAI. While the SpliceAI dataset has slightly more acceptor than donor sites, mine has slightly more donor sites.

18% of start and stop annotations of genes (primary transcript for SpliceAI, all isoforms for CI-SpliceAI) and 58% of splice sites overlap between the lifted SpliceAI training set and ours when filtered to the same chromosomes.

6.4.3 Performance Comparison of Aberrant Splice Annotation Tools

The models trained on the train split recognises unseen splice sites on chromosomes 1,3,5,7 and 9 with 94% average precision. This is comparable to what was previously achieved on *gencode.v33grch38.collapsed* using the train and test code from SpliceAI (section 3.4.5 on page 54).

Table 6.2 shows how well all algorithms predict splice disruptions on the binary classification task. Only three algorithms, the sliding MES, SQUIRLS and CI-SpliceAI annotate all variants, the original SpliceAI and MMSplice reach 99% coverage and MES run via VEP annotates just over half. Missing annotations were handled as if they annotated no splicing effect, which is how *in-silico* systems are utilised when filtering variants. Despite missing almost half of all annotations, MES as run on VEP returns scores of around 90%. The sliding MES returns the worst scores followed by SQUIRLS. CI-SpliceAI returns the highest scores with a lead of over a percentage point in all measures above the second-best algorithm, SpliceAI.

	Coverage	AUC-PR	Optimal Threshold	Accuracy (on opt. thresh.)
MES (Sliding)	**100%**	55.68%	2.500	53.42%
SQUIRLS	**100%**	91.32%	0.074	85.64%
MES (VEP)	58%	92.52%	2.109	86.40%
MMSplice (Splicing Efficiency)	99%	93.03%	1.119	87.23%
MMSplice (Pathogenicity)	99%	94.13%	0.961	88.53%
SpliceAI	99%	96.21%	0.300	90.88%
CI-SpliceAI	**100%**	**97.25%**	0.190	**92.17%**

TABLE 6.2: **Annotation coverage and predictive performance of all algorithms annotating if splicing is disrupted or not, on all 1,316 variants.** Coverage refers to how many variants were annotated.

Figure 6.6 visualises the precision-recall curves of all algorithms which shows how well the algorithms investigated can balance these two measures by adjustment of the threshold. The area under the curve is highest for CI-SpliceAI and its curve is smooth, indicating good adaptability to different thresholds. In contrast, the plateaued high recall values on the MES (VEP) curve for example shows that it cannot detect some splicing defects without classifying everything as disruptive. This is due to it only being applied on an existing splice site with a small context size, causing it to miss deeper variant effects and pseudo exon creation. For every algorithm (except for the sliding window MES), the area under the curve is high. The sliding MES performs barely better than random chance and its curve is very rugged, indicating unstable and unusable results. The deep learning curves are significantly more smooth which should facilitate balancing of sensitivity and specificity, with CI-SpliceAI scoring higher than SpliceAI for almost all thresholds.

The area under the *Precision Recall* (PR) curve describes how well the algorithms perform under all possible thresholds. This is however not how clinicians would use the algorithm: For variant diagnostics, a researcher is not interested in every possible threshold. Instead, the threshold will be set to trade off false positives and false negatives as the study requires. This means that even if the gap between CI-SpliceAI and SQUIRLS is only 6 percentage points *for all thresholds*, at high recall there is a significant gap in precision. For example, at a recall of ~ 0.96, the difference in precision between the two algorithms is 19 percentage points.

Performance on predicting the exact effect of a variant on the mature mRNA is documented in table 6.3. CI-SpliceAI ties with SpliceAI when predicting donor gains and performs at least one percentage point better modelling acceptor gains and splice losses. The sliding MES again not produce usable results. None of the other algorithms models splicing on a per-nucleotide basis which is needed for this analysis.

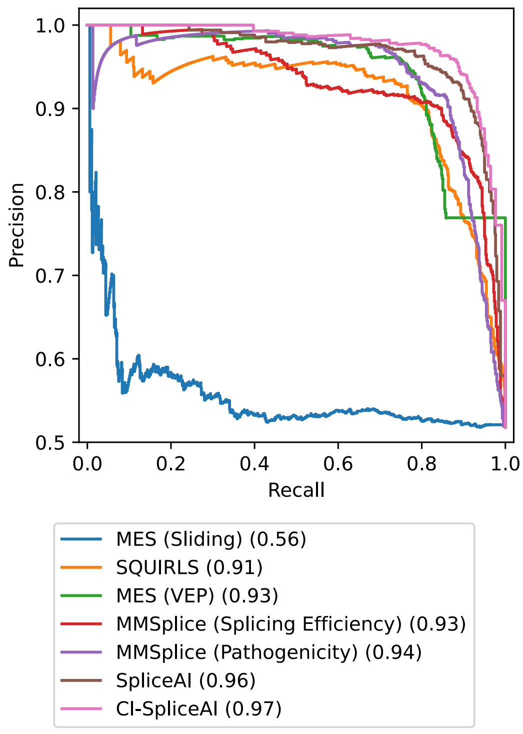


FIGURE 6.6: **Precision-Recall curves of all algorithms on all 1,316 variants.** The area under each curve is indicated in the legend. CI-SpliceAI outperforms all competitors at almost every position. Its shape is evenly round.

	Acceptor Gain	Acceptor Loss	Donor Gain	Donor Loss
MES (Sliding)	0.00%	1.16%	2.33%	2.25%
SpliceAI	87.50%	77.10%	**79.07%**	78.93%
CI-SpliceAI	**93.75%**	**78.55%**	**79.07%**	**82.02%**

TABLE 6.3: **How well algorithms predict the exact variant effect on the mature mRNA, based on the subset of 388 variants**

Figure 6.7 compares how well SpliceAI and CI-SpliceAI (trained on the 19 training chromosomes) can predict splice sites on the gene *CFTR* (comparable to Figure 1B in Jaganathan et al., 2019), located on chromosome 7 which is excluded from the training data. SpliceAI predicts a range of splice sites that are not contained within the validated isoform data, including one exon, and misses a few, with 15 mispredictions in total, half of which are false positives. CI-SpliceAI, when being trained the same subset of chromosomes, mispredicts seven sites with no false positives, and when trained on the whole dataset, it still misses four. The new algorithm performing better is not entirely surprising as it was trained on only junctions in GENCODE, whereas the original SpliceAI was also trained on novel junctions from the GTEx cohort. Due to the higher concentration of junctions in the SpliceAI data (table 6.1), SpliceAI recognises many more false positives than CI-SpliceAI. When the CI-SpliceAI model is trained on all chromosomes, splice site recognition improves further, however the model does not learn every single splice site.

The predictive error, i.e. the absolute difference between prediction and ground truth, is compared between SpliceAI and CI-SpliceAI and visualised in Figure 6.8. It shows that predictions improved (compared to the original SpliceAI algorithm) on all positions, the model did not get biased towards acceptors, donors, or consensus regions, independent of the distance to their closest respective variant. It also shows however that there are data points where predictive quality decreased for some variants. The majority of predictions produced already small errors originally due to the high performance, and the majority of them improved even more. Confidence improved for a magnitude of correctly classified variants, with some significantly high improvements. A cluster of highly confident mis-predictions was identified where CI-SpliceAI is even more certain of the wrong label; this might indicate systematic flaws in both deep learning models or even in the ground truth.

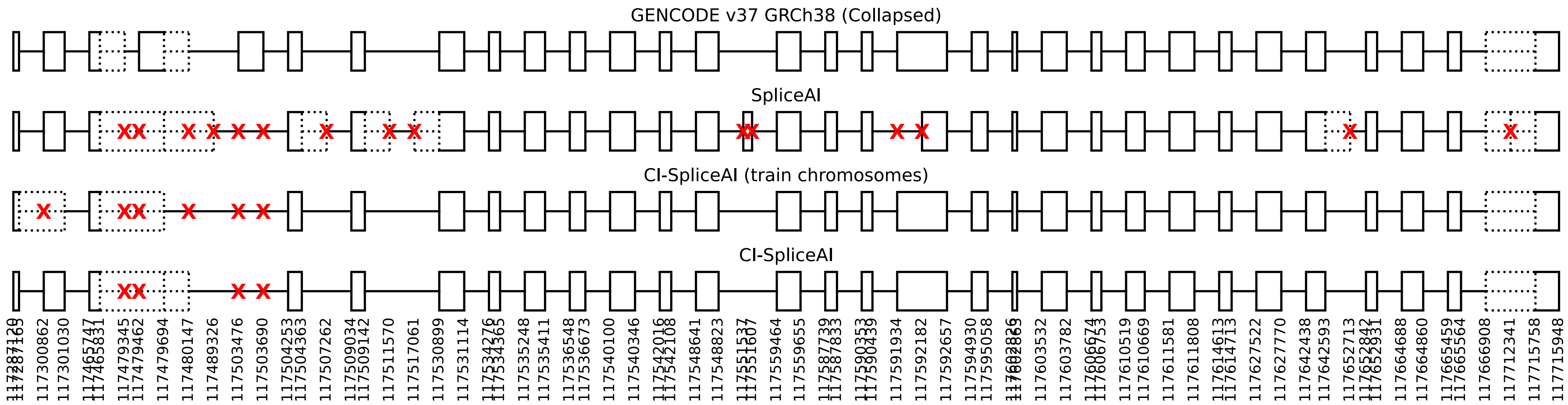
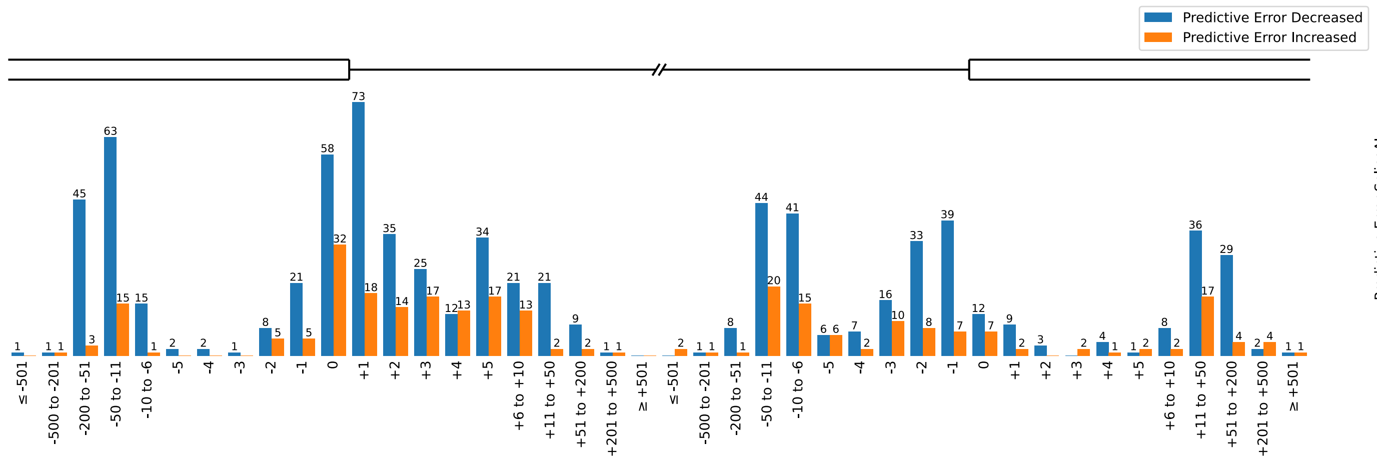
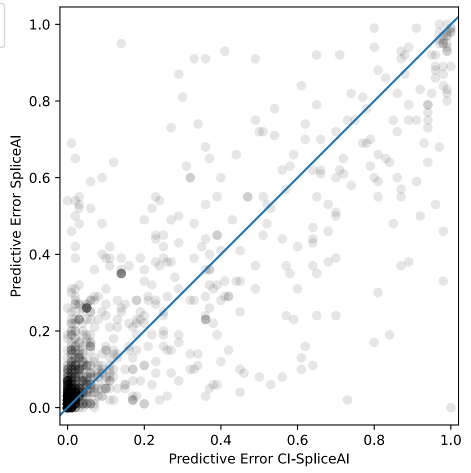


FIGURE 6.7: **Predictions of splice sites in *CFTR* in comparison to the ground truth from GENCODE.** Mispredictions are marked with a red X. The original SpliceAI algorithm misses one exon, adds one extra, and mispredicts 15 sites in total, many of which are false positives. When trained on the training chromosomes of collapsed data, CI-SpliceAI misses two exons, does not add any extra exons, and mispredicts 7 sites in total. Extending training to all chromosomes (including this gene) causes only one missing exon prediction and the overall error decreases to 4 mispredicted sites in total.



(A) Offset of predictive error



(B) Magnitude of predictive error

FIGURE 6.8: **Comparison of the predictive error between SpliceAI and CI-SpliceAI on all 1,316 variants.** The predictive error is the absolute difference between the most significant predicted annotation score and the ground truth. 78% of predictions in CI-SpliceAI have a different output than when annotated with SpliceAI, out of which 73% have a smaller predictive error. (a) Predictions for CI-SpliceAI improved relative to SpliceAI for almost every position; there is no obvious bias where predictions worsened at a specific distance from a splice site, neither for the site type nor the distance from it. (b) The magnitude of the change in predictive error. Variants on the identity line did not change their score; points lying above the identity line improved, points below worsened. The big cluster at the bottom left corner represents points where our algorithm shows improved confidence in correct predictions. A small cluster of mispredicted variants of high confidence have worsened (top right corner).

6.4.4 Installation and Usage of the Annotation Module

The python annotation module improves on some small flaws of the original module. It also supports batch predictions on the GPU by automatic padding of sequences of different length and dynamic batching based on data size, which should increase its applications significantly. It does not only annotate VCF files, it can also be used programmatically in a python script.

The python module for variant annotation can be installed with the command

```
pip install cispliceai[cpu]
```

The ”[cpu]” suffix installs tensorflow without GPU support. Alternatively, the user can install and set up CUDA and a compatible tensorflow version with GPU support manually and omit the CPU suffix. An in-depth guide on how to install and use the module was published to `https://ci-spliceai.com/install` and PyPI.

At the time of writing, the CI-SpliceAI python module was downloaded 1,118 times from PyPI (Flynn, 2022).

6.4.5 Infeasibility of Pre-Computed Scores and VEP Plugin

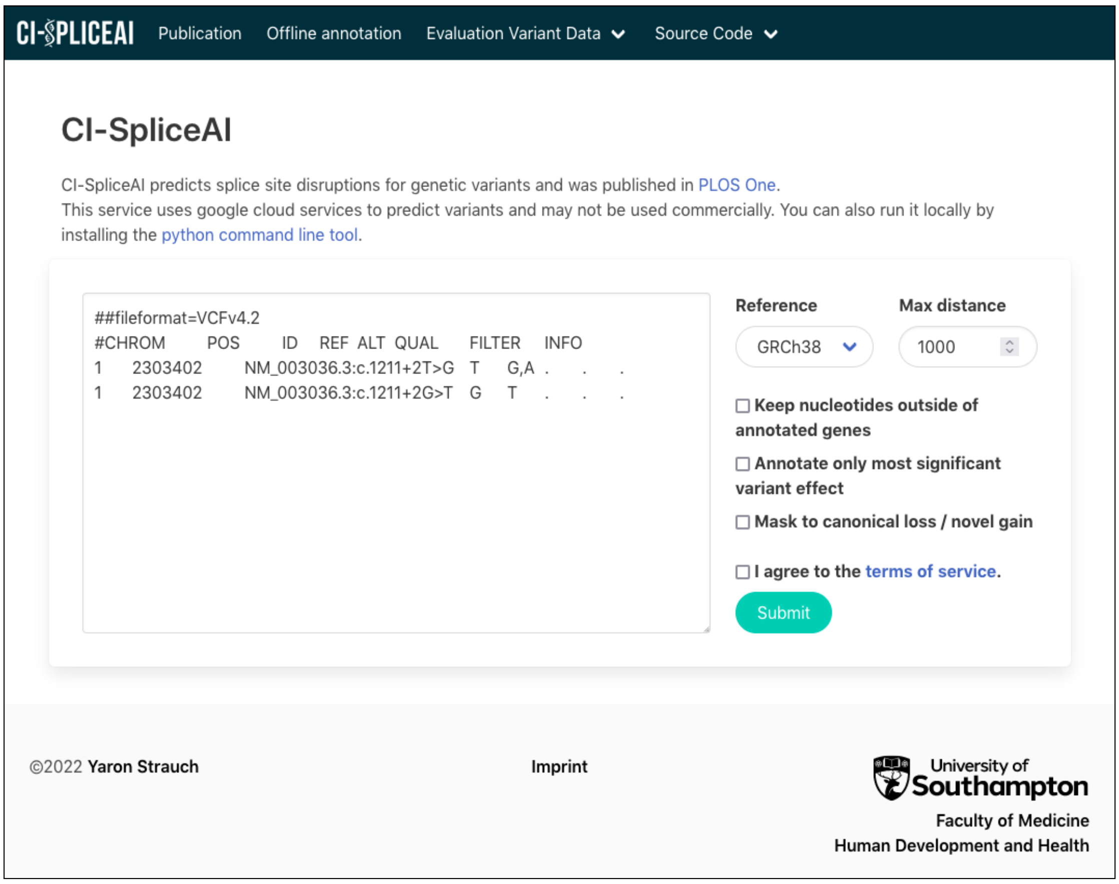
The prototypical implementation has shown that pre-computing all scores is infeasible on the IRIDIS infrastructure. Within 60 hours, the algorithm annotated 8.5 million variants using four GPUs, equal to 35,617 variants per GPU hour. All 1 base insertions and 1-4 base deletions make up 12.6 billion variants. It was estimated that even with a potential 3% performance gain (by optimising the prototypical implementation) and using every single GPU on the IRIDIS supercomputer, calculating the same volume of variants as published by the SpliceAI authors would take over a year. Even in a more realistic framework, such as only annotating SNVs in protein coding regions, eight GTX1080 and four Tesla V100 were estimated take over 100 days.

As this would be infeasible, it was decided to not pre-compute any scores. This also means that no VEP plugin was developed.

6.4.6 Usability and Throughput of CI-SpliceAI Web

6.4.6.1 Frontend Concept

The website (Figure 6.9) is designed with focus on the VCF input. A big, central text area in the middle allows copy and paste of VCF files. All command line parameters are be accessible through form elements such as buttons, checkboxes, and dropdowns.



CI-SpliceAI

CI-SpliceAI predicts splice site disruptions for genetic variants and was published in PLOS One.
This service uses google cloud services to predict variants and may not be used commercially. You can also run it locally by installing the python command line tool.

```
##fileformat=VCFv4.2
#CHROM    POS      ID    REF  ALT  QUAL    FILTER    INFO
1    2303402    NM_003036.3:c.1211+2T>G   T   G,A  .   .   .
1    2303402    NM_003036.3:c.1211+2G>T   G   T    .   .   .
```

Reference: GRCh38
Max distance: 1000

☐ Keep nucleotides outside of annotated genes
☐ Annotate only most significant variant effect
☐ Mask to canonical loss / novel gain
☐ I agree to the terms of service.

©2022 Yaron Strauch

FIGURE 6.9: **The CI-SpliceAI Web main page**. Users can submit a VCF file and choose parameters as needed.

A main navigation links to all code repositories, the PLOS One publication, and an installation guide. Before submitting any variants, the user must consent to terms and conditions underlying the academic non-profit scope of the project.

Faulty inputs and errors due to quotas are communicated transparently to the user. A simple footer links to all uni resources.

The website works on all resolutions and is fully responsive even on mobile devices, although it's improbable anybody would want to use it on a mobile phone. It was tested to work on the latest desktop versions of Firefox, Chrome, Safari, Edge, and on the latest mobile Chrome and Safari (Android and iOS respectively).

All error cases relating to the Google backend are being handled, i.e. if the system runs out of money or if Google workers fail, an appropriate message is shown to the user.

6.4.6.2 Website

The CI-SpliceAI Web page is accessible under `https://ci-spliceai.com`.

In the current configuration, CI-SpliceAI Web can process between 23 and 100 variants per minute, depending on the requested maximum distance from a variant. This can easily be increased by scaling the number of concurrent workers up.

Based on the monthly Google limits on request count, computation, storage, and memory, CI-SpliceAI Web should be able to process between 70,000 and 300,000 variants per month, again depending on the requested maximum distance form a variant. There is however another constraint on network egress, which is hard to estimate due to HTTP traffic commonly using a gzipped protocol; so far this constraint did not constitute a bottleneck of the application.

In the 20 weeks following the release of CI-SpliceAI, 1,075 unique anonymised IP addresses visited the website. In this period, 49,790 variant annotations have been queried from the database plus another 5,596 where the input reference did not match the Reference Genome. Nobody reached the maximum daily quota of read or create variant quotas, one submission exceeded the restriction of 2,000 variants per VCF input. The average computation time on Google is 5 seconds per variant (between 3 and 23 seconds in the extremes). At the time of writing, the database consists of 16,346 annotations describing 7,494 variant inputs for machine learning. Variants affect 6,471 reference annotations.

There was a bug with certain deletions (those using "." as ALT annotation) that caused these variants to not being computed and, after a week, clogged the whole queue. This bug started on the 09/10 and was fixed 9 days later after a user reached out. To prevent future issues from being undetected this long, error monitoring and reporting was set up on the Google cloud. Excluding these 9 days and the pre-publication period, the total waiting for a variant annotation from submission to insertion into the database is 1:36 minutes on average, ranging from 3 seconds to 24 minutes in the extremes.

6.5 Discussion

The variant dataset of 1,316 variants is one of the biggest datasets of its kind to date. 388 variants were additionally annotated with the exact effect on the mature mRNA with basepair resolution, which allows measurement of how well the underlying process is modelled by the algorithm. To my knowledge, this kind of dataset and analysis is completely novel and shows that the deep learning predictions from sequence alone can do more than just binary annotation, in contrast to MMSplice, MES (through VEP), and SQUIRLS.

The CI-SpliceAI software package should allow the wider scientific community to use the application and hopefully prove useful for their research. All code to train, test, and apply the application was re-implemented and open sourced.

The new CI-SpliceAI model outperforms all competitors in all measures, except at predicting donor gains where it tied with SpliceAI. It is important to highlight that the final SpliceAI model was not trained on all chromosomes and excluded chromosomes 1,3,5,7 and 9. It was verified that re-training SpliceAI on all chromosomes improved performance by only 0.05 percentage points on the average-precision, which is negligible.

MES performed competitively when used as intended, as a VEP module, even though it only annotated 58% of variants. The algorithm design however does not allow recognition of non-canonical and novel splice sites. It was evaluated if an application of the algorithm around each variant would compensate that. While in theory this allows MES to recognise gains of novel sites, in practice its predictive utility for both the binary and exact classification tasks was so poor that using MES in this way is not advisable. It shows that 9 or 23 bases around a site in question are not sufficient to predict splice sites and how genetic variants affect mature mRNA.

The train code is technologically equivalent to the SpliceAI training code, except for some technical optimisations. However there are no new algorithmic choices concerning the CNN architecture. Optimising the underlying ML model is an important avenue to be further investigated, which was unfortunately not possible due to limited experience in the design of deep learning algorithms and time constraints.

For ease of use, a website with batch interference on the Google Cloud was implemented. This website is more versatile than the one provided by the Broad Institute as it allows many variants to be queried or submitted in a batch. It should also be much more resilient due to the decoupling of the prediction logic from all the other administrative tasks. A quick extension could be to also allow predictions using SpliceAI. Depending on usage, an email notification system could be added to let users know when their results are ready. This is currently not needed as waiting times observed are very short.

Unfortunately, it was too computationally expensive to pre-compute scores, and the integration into Ensembl VEP was therefore not possible. The website should be enough of a replacement as it accepts VCF files and is even easier to use than VEP.

6.6 Conclusion

Supervised ML algorithms can never be better than the ground truth they were trained on. Research into the quality of training data is time consuming but an important step for ML research.

CI-SpliceAI, trained on recent GENCODE data, collapsed splice sites annotated by humans rather than including novel sites from GTEx data, was shown to outperform all of its competitors.

To measure application to the clinical setting, the dataset from chapter 4 was further enriched with two new sources to a total of 1,316 variants, on which CI-SpliceAI was not only the best algorithm predicting if variants disrupt splicing, but also outperfomed or tied SpliceAI on a subset of 388 variants where the exact effect on the mature mRNA was known.

CI-SpliceAI was published in PLOS One (Strauch et al., 2022), and all of its train, test, and variant data is available to the general public open source. To help adaptation, a new website was created where users can annotate variants online free of charge, with a Google Cloud backend and a database cache allowing submission of new and querying known variants. Objective 4, publishing CI-SpliceAI, is considered resolved.

Chapter 7

Conclusions

7.1 Summary of Investigations

Splicing was shown to not only be an inherently complex task from a biological standpoint that can cause versatile disease, but understanding and modelling it through computational methods was shown to be of comparable difficulty. Chapter 1 set the scene for this work by establishing how computational investigation of splicing disease can help speed up diagnosis at larger scale for personalised medicine. Statistical and ML techniques were introduced and used throughout this project. Concrete applications of these techniques and strategies supporting development, data, and ethics were laid out in chapter 2, connecting the theoretical background with practical strategies.

Chapter 3 investigated how to predict mature mRNA from DNA sequence. Investigations were conducted from the ground up, starting with curation of splice data, statistical analyses and visualisations thereof, leading to trained ML models to predict splicing from raw DNA sequence. Two major dataset variations were created and investigated, one representing strong splice sites from primary isoforms and one incorporating a collapsed representation from all isoforms curated by researchers. Shallow models were trained and functioned as a baseline for comparison.

To evaluate clinical utility, all models were compared in a separate experiment on patient variant data in chapter 4. To do so, variant data and their effect on splicing was aggregated from literature. All ML algorithms were evaluated and it was found that deep learning models are significantly better than shallow learners.

Chapter 5 investigated if shallow learning models working on raw sequence data can be improved through feature engineering. Annotation of conservation scores and protein binding sites did not improve predictions. Neither did a new encoding of DNA sequences in semantic space, where similar sequences are encoded close to each other, which was hoped to compensate for the limited context sizes of basline classifiers.

With this knowledge, the best model found in chapter 4 was made production-ready and compared to competing algorithms in the field in chapter 6. This novel deep learning model, named CI-SpliceAI, is a variant of SpliceAI where only the train data was replaced. While the authors of SpliceAI took primary isoforms and enriched them with novel splice sites from GTEx data, CI-SpliceAI is trained on the collapsed representation of HAVANA annotations which arguably are of higher quality. A whole software package around CI-SpliceAI was created and published to hopefully encourage other researchers to evaluate it on their data. The software package supports offline annotations via command line or python scripts. A website with corresponding infrastructure and software engineering was created where researchers can submit variant data for free. Splice annotations are generated in the cloud and cached to a database free of charge and without registration. To further test its predictive capabilities, a subset of variants was annotated with their exact variant effect on the mature mRNA, and it was shown that the new CI-SpliceAI perfomed better in predicting the variant effects on the mature mRNA, or at least tied with SpliceAI in one measure.

7.2 The Relevance of Weaker Splice Sites

The initial statistical investigations conducted in chapter 3 did not find big differences between the dataset consisting of primary isoforms and the one that collapsed all manually curated isoforms into one. However when applying supervised ML, the differences became very much apparent and it was shown that classification of weaker splice sites was a harder task to achieve, which is something reported by the SpliceAI authors as well.

This is an example of an ongoing discussion in the ML field. Lower scores do not always indicate lower applicability. It relates to a fundamental decision every ML researcher needs to face: When should we focus our research to increase performance, and when do we need to re-focus our experiments towards application and accept negative results. The fail-fast mentality introduced by Silicon-Valley certainly has its benefits for rapid prototyping and translation to production, but the wider ML research community needs to be careful not to overly focus on preliminary results and abort investigation when negative results occur.

Despite objectively worse prediction scores on the dataset containing weaker splice sites, the deep CNNs trained to recognise all splice sites (including weaker ones) outperformed all other models in the variant data experiment of chapter 4. The models' ability to predict aberrant splicing is a better indicator for clinical utility and to determine if the algorithms model splicing correctly. This result shows that higher performance scores need to be contextualised and might not always tell the whole story. By training on harder examples, performance measures went down but clinical utility increased. From a biological perspective, this makes sense: Recognition of weaker splice patterns is intuitively harder, so a reduction in performance of recognising harder splice sites is to be expected. It also makes sense that despite lower scores, clinical utility grows, because variants might affect these weaker transcripts.

Of course the clinical experiment transfers the bias of ”strong sites are easier to recognise” from one evaluation dataset to another. In the same way that recognition of weaker splices is harder, prediction of how weak sites are altered or created by a variant is intuitively expected to be harder as well. If therefore all variants in the clinical data were related to weak splice sites, of course the model trained to recognise those would come out on top. This reduces the confidence in the higher scores and revives the discussion of higher scores versus utility. I tried to mitigate this bias by the law of big numbers and collecting as many variants as I could, so that variants affect both strong and weak isoforms. There also is a likely selection bias in my source material. Scientists are more likely to run functional analysis to determine splice defects on obvious candidates, which might bias my data towards strong splice sites. Nevertheless which way the data is skewed, I would also argue that by collecting data from many studies, my methods are sampling a set of variants of interest to the scientific community, so the data bias might actually in favour of current research needs.

The SpliceAI authors observed the discrepancy of lower scores in splice recognition and higher clinical utility too. In the domain of ML, publication of lower scores is often frowned upon and we as researchers are often forced to re-focus publications accordingly. They did so by splitting the relevant parts of their publication in two: Reporting performance on splice site recognition of the primary isoform, then training on a much more difficult set of sites without reporting their (lower) splice site recognition scores, and applying this model to variants. That way they could always report the higher score. The paper reports on this arguably quite briefly compared to the overall size of their publication, which caused some confusion on my side during re-implementation and when publishing my CI-SpliceAI algorithm. Ultimately I followed the same pattern of reporting measures on two models, which I followed up with a dedicated discussion in the paper.

We can therefore conclude that incorporation of weaker splice sites of high quality is indeed very important to improve clinical utility.

7.3 Deep versus Shallow Learning

To determine if the immense model depth and context size of SpliceAI is actually required for this task, simpler baseline algorithms were trained and tested on reduced data sets using cross-validation (chapter 3). Direct comparison of how well deep and shallow learning can recognise splice sites was not feasible due to the computational complexity and the immense size of the human genome. This is a common problem: Deep learning models can make use of parallel computation on the GPU, allowing to process significantly more data, while shallow models cannot.

Again, the variant analysis described in chapter 4 provided a fair comparison to compare deep and shallow learners. It was found that shallow models are not good at predicting (aberrant) splicing from DNA sequence alone. The deep learning models outperformed shallow learners significantly and were able to predict splice disruptions very accurately. The big context size and model complexity allowed the CNNs to predict even deep intronic variants, and the design of convolutional layers allows the deep learning networks to consider many neighbouring nucleotides at a time and can therefore detect patterns independent of their relative position, such as binding sites.

Novel strategies to close the gap between shallow and deep learning models were explored in chapter 5. Literature shows basically two ways of how to predict splicing disease: Either using raw sequences and deep learning, as SpliceAI does, or by using engineered features and simple algorithms such as the approach taken by SQUIRLS or MES. SQUIRLS uses heavily engineered features where parts of the problem, for example the distance to the next splice site, are already known; researchers have introduced their domain knowledge and arguably parts of the solution to assist simpler algorithms in their predictions. This work evaluated a compromise between both worlds by using DNA sequences annotated with features we know are part of or indicative of splicing without disclosing too much of the solution. Annotation of conservation scores and regulatory binding sites and encoding of the DNA sequences in a semantic space were tried out to bridge the gap. Unfortunately, none of these strategies worked and it seems that one has to decide between deep learning on raw data or heavily engineered data with simpler algorithms.

Which of these two options is preferable is a mix of personal taste and how we focus our engineering towards a desired use case. To me the approach of using raw sequencing data is preferable for annotation of splice disruptions because the algorithm needs to model the splicing process itself. Its ability to predict aberrant splicing directly correlates with how well it understands the mechanism. An algorithm that has never seen a splice variant during development and is not handed parts of the solution is preferable to tools that have been trained specifically on variant data where parts of the solution are in its input, especially considering that in chapter 6 deep learning was observed to actually outperform SQUIRLS.

Inputting parts of the solution is a potential bias and is another example that comparing scores is not the only measure to assess the utility of a tool. Engineering existing splice sites as input to MES or SQUIRLS causes predictive performance to increase, but at a potential price of reducing their utility. This price was measured for MES in chapter 6. By design, MES as run through VEP was not able to classify novel splice sites because it is only run on known junctions from the reference genome. By exploiting parts of the problem through algorithmic design, the VEP implementation of MES is unable to predict novel junctions. When accommodating this by running MES in a window around the variant rather than the known splice site, predictions were comparable to random. MES does not understand or model splicing. The deep learning tools on the other hand seem to do: They outperformed all competitors including shallow learners with highly engineered features.

The deep learning approach also enabled prediction of the exact effect of a variant on the mature mRNA with reasonably high success rates. Excluding SpliceAI and the near-random results for MES (when applied around the variant), none of the competitors could even provide this analysis. The experiment to predict the exact variant effect in this scale is, to my knowledge, completely novel. The 388 annotated variant effects were released to the general public to encourage the community to repeat similar analyses and focus on how to refine algorithms to model splicing this granularly. As a tool for large scale screening and variant prioritisation, based on the experimentation conducted, it can be concluded that CI-SpliceAI might be the best tool currently available.

But even if SpliceAI and CI-SpliceAI are the best tools to predict splice disruptions as argued earlier, the argument made that higher scores are not always equal to better application remains to be made again. While the results suggest that deep learning can model splicing better than shallow learning, it was not possible to demonstrate how. Feature contributions of deep learning were analysed towards both recognition of splicing and annotation of splice variants. The results were so detailed and the big context size made it impossible to understand how the algorithm reached its conclusions. This is the other side of the coin: Deep learning is an inherent black-box algorithm and no insight into the underlying biological mechanism could be derived. If features were already engineered by a human with intent and domain knowledge, even if it contained parts of the solution, investigating feature contributions could return more meaningful insights. Selecting model architectures and designing data pipelines is always subject to the investigations conducted. As a research instrument to generate insights into the underlying biology, deep learning might not be the best approach yet, at least not with the techniques evaluated.

7.4 Standards in Bioinformatics

There is no single standard of how transcripts, genes, or variants are reported and described in the literature. The SpliceAI development team reported splice sites per gene symbol which is ambiguous, rather than utilising a more standardised ID such as the one provided by Ensembl. This complicated merging datasets of different sources and versions such as when annotating paralogs. Gene symbols change on different builds and often a gene has more than one symbol, rendering them as a means of identification less ideal.

For the description of variants, the standard from HGVS unfortunately does not solve this problem. As described in chapter 4, despite the HGVS ID describing variants in a standardised format, it actually encapsulates many sub systems and different ways to describe relative positions. Allowing many different IDs to describe the same variant on the DNA defies the purpose of an ID, which by definition is to identify something uniquely. Variant IDs are also often reported incorrectly due to the complexity of the system. There are efforts to help validation and correction of these IDs, for example VariantValidator, a freely accessible online tool (Freeman et al., 2018). Making such harmonisation tools mandatory upon publication would certainly help studies aggregate data in a more streamlined way.

Similarly, there is not one gold standard for the laboratory experimentation conducted in the data sources. Evidence for or against aberrant splicing in the literature was derived by many different methods, ranging from high quality functional evidence based on sequencing over minigene assays to merely computational indications in one of the sources, which had been removed due to insufficient data quality. Efforts to standardise this aspect, such as the ACMG guidelines, are helpful if applied, however this is not always the case and studies often employ multiple labs with different protocols, making classification under these guidelines more intransparent.

These problems are certainly rooted in the diversity of nations, programs, and corporations where genetic research is conducted in. It is easy to point out these flaws as an interdisciplinary researcher coming from a different field, and even more so without suggesting better strategies. Computer Science however is an example of how a community introduced standardisation, even if, arguably, it might have overshot some times in doing so, on an international scale with many directly competing organisations. In the bioinformatics domain, it would be recommendable for the leading journals in the discipline to come together to decide on one standard so that others would follow for convenience.

Another important technological standard that is currently changing is the sequencing technology itself. One of the popular techniques to date for sequencing data is next generation sequencing, a cost-effective method which sequences millions of short reads in parallel which then have to be re-assembled and aligned to each other (Behjati and Tarpey, 2013). This puzzle process is done *in-silico* and cannot be perfect: Due to many repetitions on the human genome, many short reads will be the same and it's impossible to decide if they constitute duplicate reads of the same area or are actual repetitions on the genome. This invariably leads to regions that are not sequenced my next-generation sequencing and might be important in the overall regulatory processes. This can be solved by long read sequencing techniques like nanopore technology, which in the last few years became reliable enough to potentially become a new gold standard (De Coster et al., 2021). Long read sequencing allows sequencing of heavily repeated regions such as telomere regions, which are impossible to sequence correctly with short reads. The Telomere-to-Telomere consortium is an effort to sequence the full human genome with nanopore technology to derive the first complete human reference genome without gaps and including all repetitions and they recently published the full sequence of the X chromosome (Miga et al., 2020). Substitution of the human reference genome (GRCh38) for a completely sequenced genome derived by this technology will certainly improve data quality and lead to new insights. Judging by how many labs and publications are still using the deprecated GRCh37 genome however, wide-spread adaptation is unlikely to happen timely after a hypothetical full publication of a long-read reference genome.

7.5 Tissue-Specific Splicing

The annotations used throughout this work are based on GENCODE and HAVANA annotations. Identification of the primary transcript from GENCODE was not always straight forward and did not match with the selection from SpliceAI. A more contemporary data set to identify primary transcripts for future work would be the MANE Select transcript, which was not available at the start of research. The selection of the primary isoform through either method is the strictest one and is not specific to alternative splicing. While SpliceAI introduced many novel splice sites from GTEx to account for tissue specific splicing, this work used the more conservative approach of incorporating HAVANA annotations by filtering of the GENCODE source and collapsing them to one pseudo transcript. The results have shown that this more conservative approach seems to be easier for splice site and variant classification, and it improved clinical application. By collapsing them into one, the information on tissue-specific mechanisms however is lost. This might be the biggest drawback of the investigations conducted, and explain some discrepancy to the ground truth in the variant data experiment.

The ground truth to the variant data from chapter 4 is evaluated in different tissues, for example whole blood and kidney. Some splice irregularities might not occur in the tissue where functional analysis was performed on. If a computational model could provide tissue specific predictions, not only could this potential bias be evaluated, it may also be of great clinical utility. Obtaining certain tissue samples such as from the brain is not feasible for diagnosis of many patients. If we could instead sequence their DNA and predict tissue-specific mRNA expression, diagnosis of disease might be significantly easier and precise.

By filtering GTEx junctions to a specific tissue, relevant splice sites could be extracted to form a new training dataset. Instead of the binary annotation format as used throughout this work, splice site strength could be used as ground truth, which in itself might improve predictive accuracy and clinical application. ML algorithms could then be trained to predict splice site strength, one algorithm per tissue-specific dataset. How well these models perform could be evaluated in a common tissue such as whole blood before application to more inaccessible regions.

A second, more sophisticated approach would be to combine isoform predictions into one model instead of separating each tissue. The model architecture would need to have some input to indicate the desired tissue context. Expression of biomarkers of regulatory elements respective to certain tissue may be used to do so, how exactly to derive these needs to be explored.

I believe this is the most promising next step that will help bringing more precise and more relevant diagnosis to patients in the near future.

Appendix A

Code and Data Availability

A.1 Splice Site and Variant Prediction

All data and code used in chapters 3, 4, and 5 are persisted into one project accessible within the university git repository located at `https://git.soton.ac.uk/yls1n18/spliceai`.

Different versions of the codebase are found in git *branches* of this project:

1. Conservation score experiments are found in the branches *feat/phylop46*, *feat/phylop100*, *feat/phast46*, and *feat/phast100*
2. Semantic encoding using the trained DNA2Vec model is saved to the branch *feat/dna2vec*
3. Protein regulation binding sites were incorporated on the branch *feat/ese*
4. Pre-computed scores were evaluated on the branch *feat/precompute*
5. GTEx encoding of the variant pipeline was started at the branch *feat/gtex* (not finished and not part of this thesis)
6. The *master* branch contains all remaining experiments

A.2 DNA2Vec

The code used to train DNA2Vec on IRIDIS (chapter 5) is found at `https://git.soton.ac.uk/yls1n18/dna2vec`.

A.3 CI-SpliceAI

CI-SpliceAI was described in chapter 6.

The intermediary code bases used for peer-review are saved within the university git repository, available at `https://git.soton.ac.uk/ci-spliceai`.

All final data and code for CI-SpliceAI was released to the general public:

1. CI-SpliceAI was published with PLOS One (Strauch et al., 2022)
2. The online annotation website (CI-SpliceAI portal) is available at `https://ci-spliceai.com`
3. Variant data in VCF format is available at `https://ci-spliceai.com/external/?resource=variants/variants.vcf`
4. Variant data with all metadata in CSV format is available at `https://ci-spliceai.com/external/?resource=variants/variants.csv`
5. Code to train CI-SpliceAI is available at `https://ci-spliceai.com/external/?resource=code/train`
6. Code to annotate variants using CI-SpliceAI is available at `https://ci-spliceai.com/external/?resource=code/annotation`
7. Code used to evaluate and compare CI-SpliceAI with competitors is available at `https://ci-spliceai.com/external/?resource=code/comparison`

Glossary

ACMG American College of Medical Genetics 5, 77, 146

ANN Artificial Neural Network xii, 13, 14, 19, 23, 88, 89

AUC-PR Area Under the Precision-Recall Curve xvi, xix, xx, 9, 15, 24, 26, 27, 54, 55, 73, 79, 83, 95–97, 102, 104, 105, 108, 130

CI-SpliceAI Collapsed Isoform SpliceAI xvi, xvii, 8, 26, 84, 109–111, 115–121, 123, 126, 129–139, 142, 143, 145, 150

CNN Convolutional Neural Network xiii, xix, 6, 14, 15, 17, 20, 26, 28, 29, 31, 32, 34, 37, 39, 40, 42, 44, 45, 54, 61–63, 65, 73, 79, 83–85, 104, 109, 110, 112, 138, 142, 144

CPU Central Processing Unit 29, 110, 112, 116, 123, 124, 135

CUDA Compute Unified Device Architecture 112, 135

CV Cross Validation 18, 19, 28, 31, 42, 44, 54, 62, 92, 95, 96, 104

CVT Covariance Type 43, 50

DNA Deoxyribonucleic Acid 4, 5, 7, 8, 15, 17, 20, 23, 24, 29, 34, 36, 42, 73, 74, 83–86, 88, 90, 108, 115, 120, 124, 141, 144, 146, 148, 151

DNA2Vec DNA to vector xx, 8, 26, 86, 88, 90, 92, 93, 95, 98, 104, 108, 149

ESE Exonic Splicing Enhancer xvi, xix, 2, 6, 87, 91, 96, 106

ESS Exonic Splicing Silencer xvi, xix, 2, 87, 91, 96, 106

GENCODE Gene Encyclopedia of DNA Elements xiii, xvii, xix, 7, 15, 17, 30, 32–37, 39–42, 44, 45, 54, 58–60, 62, 65, 83, 84, 111, 117, 129, 132, 133, 139, 147

GPU Graphics Processing Unit xvi, 18, 25, 28, 29, 110, 112, 114, 115, 118, 119, 123, 135, 144

GRCh Genome Reference Consortium human xiii, xix, 32, 34, 35, 42, 45, 58, 59, 69, 71, 72, 76, 77, 111–115, 119, 123, 147

GTEx Genotype Tissue-Expression xiii, 15, 17, 33–35, 41, 45, 54, 62, 65, 83, 132, 139, 142, 147–149

HGMD Human Gene Mutation Database 70, 76, 77

HGVS Human Genome Variation Society 68–72, 146

hnRNP Heterogeneous nuclear ribonucleoprotein 2, 4

HPC High-Performance Computer 25, 26

ISE Intronic Splicing Enhancer 2, 106

ISS Intronic Splicing Silencer 2

Logit Logistic Regression xx, 11, 20, 27, 32, 44, 54, 55, 66, 79, 83, 92, 95–97, 104, 105

MANE Matched Annotation from NCBI and EMBL-EBI 33, 34, 60, 147

MES MaxEntScan 6, 17, 110, 117, 118, 129–131, 137, 138, 144, 145

miRNA microRNA 3

ML Machine Learning xv, 6, 7, 9, 14, 17, 19–21, 23, 25, 26, 31, 32, 34, 36, 39, 40, 62, 65–67, 73, 83, 85, 86, 106, 109, 110, 113, 114, 116, 118–120, 122, 123, 138, 139, 141–143, 148

MLP Multi-Layer Perceptron xii, 14, 92

MLPC Multi-Layer Perceptron Classifier xii, xx, 13, 14, 27, 29, 32, 44, 54, 55, 79, 92, 95–97, 104, 105

MMSplice Modular Modeling of Splicing 6, 66, 87, 106, 107, 110, 114, 117, 125, 129, 130, 137

MNV Multi Nucleotide Variant xv, xvii, 4, 77, 78, 110, 111, 126, 127

mRNA messenger RNA xi, xvii, xx, 1–5, 8, 118, 126, 127, 130, 131, 137–139, 141, 142, 145, 148

NAS Nonsense-associated Alternative Splicing 106

NCBI National Center for Biotechnology Information 30, 33, 68, 70–72

PCA Principal Component Analysis xiii, xiv, 22, 23, 43, 47–52, 88, 93, 98, 99, 107

PPM Position-Probability Matrix 36, 45

PR Precision Recall 130

PWM Position-Weight Matrix xiii, xiv, 6, 30, 36, 37, 42, 43, 46, 52, 53, 55, 100

PyPI Python Package Index 111, 135

RBF Radial-Basis Function xi, 10, 27, 54, 55, 79, 96, 97, 104, 105

ReLU Rectified Linear Unit xii, 13, 16

RF Random Forest xiv, xx, 13, 20, 27, 32, 44, 54–56, 62, 66, 79, 94–97, 102, 104–106

RFE Recursive Feature Elimination xvi, 20, 21, 28, 29, 32, 44, 55, 63, 93–95, 102, 103, 107, 108

RNA Ribonucleic Acid 1–3, 5, 33, 34, 36, 68, 70, 71, 74, 116, 152

RNN Recurrent Neural Network xii, 13, 14

SNV Single Nucleotide Variant xv, xvii, 4, 68, 73, 74, 77, 78, 111, 126, 127, 135

SQUIRLS Super-quick Information Content and Random Forest Learning for Splice Variants 6, 7, 66, 83, 85, 87, 107, 108, 110, 117, 129, 130, 137, 144, 145

SR serine/arigine-rich 2

SVC Support Vector Classifier xi, xix, xx, 10–12, 20, 27, 32, 40, 44, 54, 55, 61, 79, 92, 94–97, 104, 105

SVD Singular Value Decomposition 22, 23

t-SNE T-distributed Stochastic Neighbour Embedding 23, 93, 98, 100, 107

UCSC University of California, Santa Cruz 26, 30, 61, 63, 91, 92

VCF Variant Call Format xvii, 110, 111, 113, 115, 117–119, 122–124, 135–138, 150

VEP Variant Effect Predictor xv, 70, 72, 77, 78, 111, 113, 114, 117, 125, 126, 129, 130, 135, 137, 138, 145

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016.

Liam Abrahams, Rosina Savisaar, Christine Mordstein, Bethan Young, Grzegorz Kudla, and Laurence D Hurst. Evidence in disease and non-disease contexts that nonsense mutations cause altered splicing via motif disruption. *Nucleic acids research*, 49(17): 9665–9685, 2021.

Nils Adermann and Jordi Boggiano. Composer - a dependency manager for php, 2012. URL `https://getcomposer.org/`. Accessed March 2022.

All-Inkl. Domains, webspace, domain webhosting, server-hosting provider all-inkl, 2000. URL `https://all-inkl.com/`. Accessed February 2022.

Mohamed Aly. Survey on multiclass classification methods. *Neural Netw*, 19:1–9, 2005.

Matthieu Aubry. Matomo analytics - the google analytics alternative that protects your data, 2008. URL `https://matomo.org/`. Accessed March 2022.

Ziga Avsec, Roman Kreuzhuber, Johnny Israeli, Nancy Xu, Jun Cheng, Avanti Shrikumar, Abhimanyu Banerjee, Daniel S Kim, Thorsten Beier, Lara Urban, et al. The kipoi repository accelerates community exchange and reuse of predictive models for genomics. *Nature biotechnology*, page 1, 2019.

Christina B Azodi, Jiliang Tang, and Shin-Han Shiu. Opening the black box: Interpretable machine learning for geneticists. 2020.

Joey Azofeifa. azofeifa/pwm_logo, 2017. URL `https://github.com/azofeifa/PWM_logo`. Accessed January 2020.

David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Müller. How to explain individual classification decisions. *The Journal of Machine Learning Research*, 11:1803–1831, 2010.

Pierre Baldi. Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pages 37–49, 2012.

Imad A Basheer and Maha Hajmeer. Artificial neural networks: fundamentals, computing, design, and application. *Journal of microbiological methods*, 43(1):3–31, 2000.

Sam Behjati and Patrick S Tarpey. What is next generation sequencing? *Archives of Disease in Childhood-Education and Practice*, 98(6):236–238, 2013.

Dennis A Benson, Mark Cavanaugh, Karen Clark, Ilene Karsch-Mizrachi, David J Lipman, James Ostell, and Eric W Sayers. Genbank. *Nucleic acids research*, 45(D1):D37–D42, 2017.

Ekaba Bisong. An overview of google cloud platform services. *Building Machine Learning and Deep Learning Models on Google Cloud Platform*, pages 7–10, 2019.

Douglas L Black. Mechanisms of alternative pre-messenger rna splicing. *Annual Reviews of Biochemistry*, 72:291–336, 2003.

Alexander JM Blakes, Htoo A Wai, Ian Davies, Hassan Ebrahim Moledina, April Ruiz, Tessy Thomas, David Bunyan, N Simon Thomas, Christine P Burren, Lyn Greenhalgh, et al. A systematic analysis of splicing variants identifies new diagnoses in the 100,000 genomes project. *medRxiv*, 2022.

Hervé Bourlard and Yves Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics*, 59(4):291–294, 1988.

Rita Dias Brandão, Kees van Roozendaal, Demis Tserpelis, Encarna Gómez García, and Marinus J Blok. Characterisation of unclassified variants in the brca1/2 genes with a putative effect on splicing. *Breast cancer research and treatment*, 129(3):971–982, 2011.

Emanuele Buratti, Martin Chivers, Gyulin Hwang, and Igor Vorechovsky. Dbass3 and dbass5: databases of aberrant 3′-and 5′-splice sites. *Nucleic acids research*, 39(suppl_1): D86–D91, 2010.

M Burset, IA Seledtsov, and VV Solovyev. Analysis of canonical and non-canonical splice sites in mammalian genomes. *Nucleic acids research*, 28(21):4364–4375, 2000.

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

Latarsha J Carithers, Kristin Ardlie, Mary Barcus, Philip A Branton, Angela Britton, Stephen A Buia, Carolyn C Compton, David S DeLuca, Joanne Peter-Demchok, Ellen T Gelfand, et al. A novel approach to high-quality postmortem tissue procurement: the gtex project. *Biopreservation and biobanking*, 13(5):311–319, 2015.

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.

Nicolas Charlet-B, Gopal Singh, Thomas A Cooper, and Penny Logan. Dynamic antagonism between etr-3 and ptb regulates cell type-specific alternative splicing. *Molecular cell*, 9(3):649–658, 2002.

Jun Cheng, Thi Yen Duong Nguyen, Kamil J Cygan, Muhammed Hasan Çelik, William G Fairbrother, Julien Gagneur, et al. Mmsplice: modular modeling improves the predictions of genetic variant effects on splicing. *Genome biology*, 20(1):1–15, 2019.

Deanna M Church, Valerie A Schneider, Tina Graves, Katherine Auger, Fiona Cunningham, Nathan Bouk, Hsiu-Chuan Chen, Richa Agarwala, William M McLaren, Graham RS Ritchie, et al. Modernizing reference genome assemblies. *PLoS Biol*, 9(7): e1001091, 2011.

Andrew Colette. Hdf5 for python, 2014. URL `https://docs.h5py.org/`.

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.

RGH Cotton and O Horaitis. Human genome variation society. *e LS*, 2001.

Nello Cristianini, John Shawe-Taylor, et al. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

Daniel Danis, Julius O B Jacobsen, Leigh C Carmody, Michael A Gargano, Julie A McMurry, Ayushi Hegde, Melissa A Haendel, Giorgio Valentini, Damian Smedley, and Peter N Robinson. Interpretable prioritization of splice variants in diagnostic next-generation sequencing. *American journal of human genetics*, 108(9): 1564—1577, September 2021. ISSN 0002-9297. doi: 10.1016/j.ajhg.2021.06.014. URL `https://doi.org/10.1016/j.ajhg.2021.06.014`.

Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240, 2006.

Wouter De Coster, Matthias H Weissensteiner, and Fritz J Sedlazeck. Towards population-scale long-read sequencing. *Nature Reviews Genetics*, 22(9):572–587, 2021.

François-Olivier Desmet, Dalil Hamroun, Marine Lalande, Gwenaëlle Collod-Béroud, Mireille Claustres, and Christophe Béroud. Human splicing finder: an online bioinformatics tool to predict splicing signals. *Nucleic acids research*, 37(9):e67–e67, 2009.

Docker. Empowering app development for developers, 2013. URL `https://www.docker.com/`. Accessed February 2022.

Heidi Dvinge. Regulation of alternative mrna splicing: old players and new perspectives. *FEBS letters*, 592(17):2987–3006, 2018.

Ian D'Souza, Parvoneh Poorkaj, Ming Hong, David Nochlin, Virginia M-Y Lee, Thomas D Bird, and Gerard D Schellenberg. Missense and silent tau gene mutations cause frontotemporal dementia with parkinsonism-chromosome 17 type, by affecting multiple alternative rna splicing regulatory elements. *Proceedings of the National Academy of Sciences*, 96(10):5598–5603, 1999.

Jamie M Ellingford, Huw B Thomas, Charlie Rowlands, Gavin Arno, Glenda Beaman, Beatriz Gomes-Silva, Christopher Campbell, Nicole Gossan, Claire Hardcastle, Kevin Webb, et al. Functional and in-silico interrogation of rare genomic variants impacting rna splicing for the diagnosis of genomic disorders. *BioRxiv*, page 781088, 2019.

Genomics England. The 100,000 genomes project. *The*, 100:0–2, 2016.

Ensembl. Ensembl rest api version 13.1, a. URL `https://rest.ensembl.org`. Accessed January 2021.

Ensembl. Ensembl variant recoder, b. URL `https://www.ensembl.org/Homo_sapiens/Tools/VR`. Accessed January 2021.

Erwin Erdem and Kishore Jaganathan. How is the principal transcript determined in the training dataset?, 2021. URL `https://github.com/Illumina/SpliceAI/issues/87`. Accessed January 2022.

William G Fairbrother, Ru-Fang Yeh, Phillip A Sharp, and Christopher B Burge. Predictive identification of exonic splicing enhancers in human genes. *Science*, 297(5583): 1007–1013, 2002.

Peter Flach and Meelis Kull. Precision-recall-gain curves: Pr analysis done right. *Advances in neural information processing systems*, 28, 2015.

David Flanagan. *JavaScript: the definitive guide*. " O'Reilly Media, Inc.", 2006.

Christopher Flynn. Pypi download stats - cispliceai, 2022. URL `https://pypistats.org/packages/cispliceai`. Accessed November 2022.

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):1–11, 2015.

Adam Frankish, Mark Diekhans, Anne-Maud Ferreira, Rory Johnson, Irwin Jungreis, Jane Loveland, Jonathan M Mudge, Cristina Sisu, James Wright, Joel Armstrong, et al. Gencode reference annotation for the human and mouse genomes. *Nucleic acids research*, 47(D1):D766–D773, 2019.

Peter J Freeman, Reece K Hart, Liam J Gretton, Anthony J Brookes, and Raymond Dalgleish. Variantvalidator: Accurate validation, mapping, and formatting of sequence variation descriptions. *Human mutation*, 39(1):61–68, 2018.

Pascaline Gaildrat, Audrey Killian, Alexandra Martins, Isabelle Tournier, Thierry Frébourg, and Mario Tosi. Use of splicing reporter minigene assay to evaluate the effect on splicing of unclassified genetic variants. *Cancer Susceptibility*, pages 249–257, 2010.

Vaishali Ganganwar. An overview of classification algorithms for imbalanced datasets. *International Journal of Emerging Technology and Advanced Engineering*, 2(4):42–47, 2012.

GENCODE. Frequently asked questions - gencode, 2020. URL `https://www.gencodegenes.org/pages/faq.html`. Accessed January 2020.

Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2019.

Hernan D Gonorazky, Sergey Naumenko, Arun K Ramani, Viswateja Nelakuditi, Pouria Mashouri, Peiqui Wang, Dennis Kao, Krish Ohri, Senthuri Viththiyapaskaran, Mark A Tarnopolsky, et al. Expanding the boundaries of rna sequencing as a diagnostic tool for rare mendelian disease. *The American Journal of Human Genetics*, 104 (3):466–483, 2019.

Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1):389–422, 2002.

Muhammed Hasan. Mmsplice vep plugin, 2018. URL `https://github.com/gagneurlab/MMSplice_MTSplice/tree/master/VEP_plugin`. Accessed February 2022.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

Naozumi Hiranuma. Integrated gradients, 2018. URL `https://github.com/hiranumn/IntegratedGradients`. Accessed November 2020.

Ourania Horaitis and Richard GH Cotton. The challenge of documenting mutation across the genome: the human genome variation society approach. *Human mutation*, 23(5):447–452, 2004.

Claude Houdayer, Virginie Caux-Moncoutier, Sophie Krieger, Michel Barrois, Françoise Bonnet, Violaine Bourdon, Myriam Bronner, Monique Buisson, Florence Coulet, Pascaline Gaildrat, et al. Guidelines for splicing analysis in molecular diagnosis derived from a set of 327 combined in silico/in vitro studies on brca1 and brca2 variants. *Human mutation*, 33(8):1228–1238, 2012.

Jing Hu, Hao Qian, Yuanchao Xue, and Xiang-Dong Fu. Ptb/nptb: master regulators of neuronal fate in mammals. *Biophysics reports*, 4(4):204–214, 2018.

J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. doi: 10.1109/MCSE.2007.55.

Kaoru Ito, Parth N Patel, Joshua M Gorham, Barbara McDonough, Steven R DePalma, Emily E Adler, Lien Lam, Calum A MacRae, Syed M Mohiuddin, Diane Fatkin, et al. Identification of pathogenic gene mutations in lmna and mybpc3 that alter rna splicing. *Proceedings of the National Academy of Sciences*, 114(29):7689–7694, 2017.

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.

Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

Xueqiu Jian, Eric Boerwinkle, and Xiaoming Liu. In silico prediction of splice-altering single nucleotide variants in the human genome. *Nucleic acids research*, 42(22):13534–13544, 2014.

Auinash Kalsotra and Thomas A Cooper. Functional consequences of developmentally regulated alternative splicing. *Nature Reviews Genetics*, 12(10):715, 2011.

Kirthevasan Kandasamy, Karun Raju Vysyaraju, Willie Neiswanger, Biswajit Paria, Christopher R Collins, Jeff Schneider, Barnabas Poczos, and Eric P Xing. Tuning hyperparameters without grad students: Scalable and robust bayesian optimisation with dragonfly. *J. Mach. Learn. Res.*, 21(81):1–27, 2020.

Konrad J Karczewski, Laurent C Francioli, Grace Tiao, Beryl B Cummings, Jessica Alföldi, Qingbo Wang, Ryan L Collins, Kristen M Laricchia, Andrea Ganna, Daniel P Birnbaum, et al. The mutational constraint spectrum quantified from variation in 141,456 humans. *Nature*, 581(7809):434–443, 2020.

Donna Karolchik, Robert Baertsch, Mark Diekhans, Terrence S Furey, Angie Hinrichs, YT Lu, Krishna M Roskin, Matthias Schwartz, Charles W Sugnet, Daryl J Thomas, et al. The ucsc genome browser database. *Nucleic acids research*, 31(1):51–54, 2003.

Olga Kelemen, Paolo Convertini, Zhaiyi Zhang, Yuan Wen, Manli Shen, Marina Falaleeva, and Stefan Stamm. Function of alternative splicing. *Gene*, 514(1):1–30, 2013.

Niroshika Keppetipola, Shalini Sharma, Qin Li, and Douglas L Black. Neuronal regulation of pre-mrna splicing by polypyrimidine tract binding proteins, ptbp1 and ptbp2. *Critical reviews in biochemistry and molecular biology*, 47(4):360–378, 2012.

Kee K Kim, Yong C Kim, Robert S Adelstein, and Sachiyo Kawamoto. Fox-3 and psf interact to activate neural cell-specific alternative splicing. *Nucleic acids research*, 39 (8):3064–3078, 2011.

Martin Kircher, Daniela M Witten, Preti Jain, Brian J O'roak, Gregory M Cooper, and Jay Shendure. A general framework for estimating the relative pathogenicity of human genetic variants. *Nature genetics*, 46(3):310–315, 2014.

Martin Komenda, Martin Víta, Matěj Karolyi, Vincent Kríž, Andrea Pokorná, et al. Word2vec in practice: A similarity analysis of medical and healthcare disciplines. 2016.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. Unsupervised machine translation using monolingual corpora only. *arXiv preprint arXiv:1711.00043*, 2017.

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.

Raphaël Leman, Pascaline Gaildrat, Gérald Le Gac, Chandran Ka, Yann Fichou, Marie-Pierre Audrezet, Virginie Caux-Moncoutier, Sandrine M Caputo, Nadia Boutry-Kryza, Mélanie Léone, et al. Novel diagnostic tool for prediction of variant spliceogenicity derived from a set of 395 combined in silico/in vitro studies: an international collaborative effort. *Nucleic acids research*, 46(15):7913–7923, 2018.

Diana Lemos. Spliceai vep plugin, 2019. URL `https://github.com/Ensembl/VEP_plugins/blob/release/103/SpliceAI.pm`. Accessed February 2022.

Jung-Chun Lin and Woan-Yuh Tarn. Exon selection in α-tropomyosin mrna is regulated by the antagonistic action of rbm4 and ptb. *Molecular and cellular biology*, 25(22): 10111–10121, 2005.

Gemma Llort, Carmen Yagüe Muñoz, Mercè Peris Tuser, Ignacio Blanco Guillermo, José Ramón Germà Lluch, Allen E Bale, and Mayra Alvarez Franco. Low frequency of recurrent brca1 and brca2 mutations in spain. *Human Mutation*, 19(3):307, 2002.

Núria López-Bigas, Benjamin Audit, Christos Ouzounis, Genís Parra, and Roderic Guigó. Are splicing mutations the most frequent cause of hereditary disease? *FEBS letters*, 579(9):1900–1903, 2005.

Jenny Lord, Giuseppe Gallone, Patrick J Short, Jeremy F McRae, Holly Ironfield, Elizabeth H Wynn, Sebastian S Gerety, Liu He, Bronwyn Kerr, Diana S Johnson, et al. Pathogenicity and selective constraint on variation near splice sites. *Genome research*, 29(2):159–170, 2019.

Mitchell R Lunn and Ching H Wang. Spinal muscular atrophy. *The Lancet*, 371(9630): 2120–2133, 2008.

Sateesh Maddirevula, Hiroyuki Kuwahara, Nour Ewida, Hanan E Shamseldin, Nisha Patel, Fatema Alzahrani, Tarfa AlSheddi, Eman AlObeid, Mona Alenazi, Hessa S Alsaif, et al. Analysis of transcript-deleterious variants in mendelian disorders: implications for rna-based diagnostics. *Genome biology*, 21(1):1–21, 2020.

Lars Malmqvist, Tommy Yuan, and Suresh Manandhar. Visualising argumentation graphs with graph embeddings and t-sne. *arXiv preprint arXiv:2107.00528*, 2021.

William McLaren, Laurent Gil, Sarah E Hunt, Harpreet Singh Riat, Graham RS Ritchie, Anja Thormann, Paul Flicek, and Fiona Cunningham. The ensembl variant effect predictor. *Genome biology*, 17(1):1–14, 2016.

Jeremy McRae, Kishore Jaganathan, Sandeep Aswathnarayana, David A. Parry, and Tor Solli-Nowlan. Illumina/spliceai, 2019a. URL `https://github.com/Illumina/SpliceAI`. Accessed January 2020.

Jeremy McRae, Kishore Jaganathan, Sandeep Aswathnarayana, David A. Parry, Tor Solli-Nowlan, Sofia Kyriazopoulou Panagiotopoulou, et al. Illumina/spliceai train code, 2019b. URL `https://basespace.illumina.com/s/5u6ThOblecrh`. Accessed January 2020.

Scott Menard. *Applied logistic regression analysis*, volume 106. Sage, 2002.

Microsoft Corporation. Microsoft excel. URL `https://office.microsoft.com/excel`.

Karen H Miga, Sergey Koren, Arang Rhie, Mitchell R Vollger, Ariel Gershman, Andrey Bzikadze, Shelise Brooks, Edmund Howe, David Porubsky, Glennis A Logsdon, et al. Telomere-to-telomere assembly of a complete human x chromosome. *Nature*, 585 (7823):79–84, 2020.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Google code archive - word2vec, 2015. URL `https://code.google.com/archive/p/word2vec/`. Accessed January 2021.

Joannella Morales, Shashikant Pujar, Jane E Loveland, Alex Astashyn, Ruth Bennett, Andrew Berry, Eric Cox, Claire Davidson, Olga Ermolaeva, Catherine M Farrell, et al. A joint ncbi and embl-ebi transcript set for clinical genomics and research. *Nature*, 604(7905):310–315, 2022.

Sumit Mund. *Microsoft azure machine learning*. Packt Publishing Ltd, 2015.

NCBI. Ncbi nucleotide (nuccore). URL https://www.ncbi.nlm.nih.gov/nuccore. Accessed January 2021.

Shane Neph, M Scott Kuehn, Alex P Reynolds, Eric Haugen, Robert E Thurman, Audra K Johnson, Eric Rynes, Matthew T Maurano, Jeff Vierstra, Sean Thomas, et al. Bedops: high-performance genomic feature operations. *Bioinformatics*, 28(14):1919–1920, 2012.

Patrick Ng. dna2vec: Consistent vector representations of variable-length k-mers. *arXiv preprint arXiv:1701.06279*, 2017a.

Patrick Ng. pnpnpn/dna2vec, 2017b. URL https://github.com/pnpnpn/dna2vec. Accessed August 2020.

NVIDIA, Péter Vingelmann, and Frank H.P. Fitzek. Cuda, release: 10.2.89, 2020a. URL https://developer.nvidia.com/cuda-toolkit. Accessed June 2022.

NVIDIA, Péter Vingelmann, and Frank H.P. Fitzek. Cuda deep neural network (cudnn), 2020b. URL https://developer.nvidia.com/cudnn. Accessed June 2022.

PackagingWG. Pypi, 2018. URL https://pypi.org. Accessed February 2022.

Alida Palmisano, Suleyman Vural, Yingdong Zhao, and Dmitriy Sonkin. Mutsplicedb: A database of splice sites variants with rna-seq based evidence on effects on splicing. *Human Mutation*, 2021.

Qun Pan, Ofer Shai, Leo J Lee, Brendan J Frey, and Benjamin J Blencowe. Deep surveying of alternative splicing complexity in the human transcriptome by high-throughput sequencing. *Nature genetics*, 40(12):1413, 2008.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

Walter Hugo Lopez Pinaya, Sandra Vieira, Rafael Garcia-Dias, and Andrea Mechelli. Autoencoders. In *Machine Learning*, pages 193–208. Elsevier, 2020.

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.

Fernando Pozo, José Manuel Rodriguez, Laura Martínez Gómez, Jesús Vázquez, and Michael L Tress. Appris principal isoforms and mane select transcripts define reference splice variants. *Bioinformatics*, 38(Supplement_2):ii89–ii94, 2022.

T Pruitt, G Brown, and Maglott D Tatusova, T. The reference sequence (refseq) database, 2002. URL https://www.ncbi.nlm.nih.gov/books/NBK21091/. Accessed January 2021.

Aaron R Quinlan and Ira M Hall. Bedtools: a flexible suite of utilities for comparing genomic features. *Bioinformatics*, 26(6):841–842, 2010.

Madara Ratnadiwakara, Monika Mohenska, and Minna-Liisa Änkö. Splicing factors as regulators of mirna biogenesis–links to human disease. In *Seminars in cell & developmental biology*, volume 79, pages 113–122. Elsevier, 2018.

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.

Radim Řehůřek, Petr Sojka, et al. Gensim—statistical semantics in python. *Retrieved from genism.org*, 2011.

Sue Richards, Nazneen Aziz, Sherri Bale, David Bick, Soma Das, Julie Gastier-Foster, Wayne W Grody, Madhuri Hegde, Elaine Lyon, Elaine Spector, et al. Standards and guidelines for the interpretation of sequence variants: a joint consensus recommendation of the american college of medical genetics and genomics and the association for molecular pathology. *Genetics in medicine*, 17(5):405–423, 2015.

Tabea Riepe and Kishore Jaganathan. Performance gtex model, 2022. URL https://github.com/Illumina/SpliceAI/issues/89. Accessed April 2022.

Lior Rokach. Decision forest: Twenty years of research. *Information Fusion*, 27:111–125, 2016.

Armin Ronacher and Fabien Potencier. Twig - the flexible, fast, and secure php template engine, 2019. URL https://twig.symfony.com/. Accessed March 2022.

Charlie F Rowlands, Diana Baralle, and Jamie M Ellingford. Machine learning approaches for the prioritization of genomic variants impacting pre-mrna splicing. *Cells*, 8(12):1513, 2019.

Tassa Saldi, Michael A Cortazar, Ryan M Sheridan, and David L Bentley. Coupling of rna polymerase ii transcription elongation with pre-mrna splicing. *Journal of molecular biology*, 428(12):2623–2635, 2016.

Valerie A Schneider, Tina Graves-Lindsay, Kerstin Howe, Nathan Bouk, Hsiu-Chuan Chen, Paul A Kitts, Terence D Murphy, Kim D Pruitt, Françoise Thibaud-Nissen, Derek Albracht, et al. Evaluation of grch38 and de novo haploid genome assemblies demonstrates the enduring quality of the reference assembly. *Genome research*, 27(5): 849–864, 2017.

Supriya Sen. Aberrant pre-mrna splicing regulation in the development of hepatocellular carcinoma. 2018.

Jannah Shamsani, Stephen H Kazakoff, Irina M Armean, Will McLaren, Michael T Parsons, Bryony A Thompson, Tracy A O'Mara, Sarah E Hunt, Nicola Waddell, and Amanda B Spurdle. A plugin for the ensembl variant effect predictor that uses maxentscan to predict variant spliceogenicity. *Bioinformatics*, 35(13):2315–2317, 2019.

Claude E Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

Matthew Shirley. pyfaidx 0.1.3 documentation, 2014. URL `https://pythonhosted.org/pyfaidx/`.

Adam Siepel, Gill Bejerano, Jakob S Pedersen, Angie S Hinrichs, Minmei Hou, Kate Rosenbloom, Hiram Clawson, John Spieth, LaDeana W Hillier, Stephen Richards, et al. Evolutionarily conserved elements in vertebrate, insect, worm, and yeast genomes. *Genome research*, 15(8):1034–1050, 2005.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

Ravi K. Singh and Thomas A. Cooper. Pre-mrna splicing in disease and therapeutics. *Trends in Molecular Medicine*, 18(8):472 – 482, 2012. ISSN 1471-4914. doi: https://doi.org/10.1016/j.molmed.2012.06.006. URL `http://www.sciencedirect.com/science/article/pii/S1471491412001013`.

Nayanah Siva. 1000 genomes project, 2008.

Il-Yeol Song and Kristin Froehlich. Entity-relationship modeling. *IEEE Potentials*, 13(5): 29–34, 1994.

Peter D Stenson, Matthew Mort, Edward V Ball, Molly Chapman, Katy Evans, Luisa Azevedo, Matthew Hayden, Sally Heywood, David S Millar, Andrew D Phillips, et al. The human gene mutation database (hgmd®): optimizing its use in a clinical diagnostic or research setting. *Human genetics*, 139(10):1197–1207, 2020.

Gary D Stormo, Thomas D Schneider, Larry Gold, and Andrzej Ehrenfeucht. Use of the 'perceptron'algorithm to distinguish translational initiation sites in e. coli. *Nucleic acids research*, 10(9):2997–3011, 1982.

Yaron Strauch, Jenny Lord, Mahesan Niranjan, and Diana Baralle. Ci-spliceai—improving machine learning predictions of disease causing splicing variants using curated alternative splice sites. *PLOS ONE*, 17(6):e0269159, 2022.

Yaron Leander Strauch. Evolving convolutional neural network topologies for image recognition. Master's thesis, University of Southampton, 2019.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. *arXiv preprint arXiv:1703.01365*, 2017.

Tabula. Tabula: Extract tables from pdf, 2018. URL https://tabula.technology/.

Suryakanthi Tangirala. Evaluating the impact of gini index and information gain on classification using decision tree classifier algorithm. *International Journal of Advanced Computer Science and Applications*, 11(2):612–619, 2020.

TGG, 2022. URL https://spliceailookup.broadinstitute.org/. Accessed February 2022.

Rebecca Truty, Karen Ouyang, Susan Rojahn, Sarah Garcia, Alexandre Colavin, Barbara Hamlington, Mary Freivogel, Robert L Nussbaum, Keith Nykamp, and Swaroop Aradhya. Spectrum of splicing variants in disease genes and the ability of rna analysis to reduce uncertainty in clinical interpretation. *The American Journal of Human Genetics*, 108(4):696–708, 2021.

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.

Santa Cruz University of California. Lift genome annotations, a. URL https://genome.ucsc.edu/cgi-bin/hgLiftOver. Accessed April 2020.

Santa Cruz University of California. Ucsc genome browser download, b. URL https://hgdownload.soe.ucsc.edu. Accessed January 2021.

Jan Van der Lubbe and Hendrik Jan Hoeve. *The R-norm information measure*. Cambridge university press, 1997.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009. ISBN 1441412697.

Guido Van Rossum and Fred L Drake Jr. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

Julian P Venables. Downstream intronic splicing enhancers. *FEBS letters*, 581(22):4127–4131, 2007.

Michel Verleysen and Damien François. The curse of dimensionality in data mining and time series prediction. In Joan Cabestany, Alberto Prieto, and Francisco Sandoval, editors, *Computational Intelligence and Bioinspired Systems*, pages 758–770, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. ISBN 978-3-540-32106-4.

Markus C. Wahl, Cindy L. Will, and Reinhard Lührmann. The spliceosome: Design principles of a dynamic rnp machine. *Cell*, 136(4):701 – 718, 2009. ISSN 0092-8674. doi: https://doi.org/10.1016/j.cell.2009.02.009. URL http://www.sciencedirect.com/science/article/pii/S0092867409001469.

Htoo A Wai, Jenny Lord, Matthew Lyon, Adam Gunning, Hugh Kelly, Penelope Cibin, Eleanor G Seaby, Kerry Spiers-Fitzgerald, Jed Lye, Sian Ellard, et al. Blood rna analysis can increase clinical diagnostic rate and resolve variants of uncertain significance. *Genetics in Medicine*, pages 1–10, 2020.

Eric T Wang, Rickard Sandberg, Shujun Luo, Irina Khrebtukova, Lu Zhang, Christine Mayr, Stephen F Kingsmore, Gary P Schroth, and Christopher B Burge. Alternative isoform regulation in human tissue transcriptomes. *Nature*, 456(7221):470, 2008.

Juan Wang, Jie Zhang, Kaibo Li, Wei Zhao, and Qinghua Cui. Splicedisease database: linking rna splicing and disease. *Nucleic acids research*, 40(D1):D1055–D1059, 2012.

Yang Wang, Xinshu Xiao, Jianming Zhang, Rajarshi Choudhury, Alex Robertson, Kai Li, Meng Ma, Christopher B Burge, and Zefeng Wang. A complex network of factors with overlapping affinities represses splicing through intronic elements. *Nature structural & molecular biology*, 20(1):36–45, 2013.

Zefeng Wang, Michael E Rolish, Gene Yeo, Vivian Tung, Matthew Mawson, and Christopher B Burge. Systematic identification and analysis of exonic splicing silencers. *Cell*, 119(6):831–845, 2004.

Luke Welling and Laura Thomson. *PHP and MySQL Web development*. Sams Publishing, 2003.

Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

Tzu-Tsung Wong. Performance evaluation of classification algorithms by k-fold and leave-one-out cross validation. *Pattern Recognition*, 48(9):2839–2846, 2015.

Andrew D Yates, Premanand Achuthan, Wasiu Akanni, James Allen, Jamie Allen, Jorge Alvarez-Jarreta, M Ridwan Amode, Irina M Armean, Andrey G Azov, Ruth Bennett, et al. Ensembl 2020. *Nucleic acids research*, 48(D1):D682–D688, 2020.

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.

Andy B Yoo, Morris A Jette, and Mark Grondona. Slurm: Simple linux utility for resource management. In *Workshop on job scheduling strategies for parallel processing*, pages 44–60. Springer, 2003.