Modeling the intronic regulation of Alternative Splicing using Deep Convolutional Neural Nets

JOHANNES LINDER
Modeling the intronic regulation of Alternative Splicing using Deep Convolutional Neural Nets

A Deep Learning methodology based on Convolutional Neural Networks for modeling the intronic regulation of Alternative Splicing using synthetic massively parallel DNA libraries.

En studie i hur den introniska regleringen av Alternativ Splicing kan modelleras och förutsättas med djupa neurala nätverk tränade på massivt parallela DNA-bibliotek.

Johannes Linder  jlinder2@kth.se

Master’s Thesis
Computer Science
KTH
June 14, 2015

Supervisor: Jens Lagergren
Examiner: Erik Fransén

In collaboration with
Seelig Lab
Dept. of Electrical Engineering
and Computer Science & Engineering
University of Washington
Supervisor: Georg Seelig
Abstract

This paper investigates the use of deep Convolutional Neural Networks for modeling the intronic regulation of Alternative Splicing on the basis of the DNA sequence. By training the CNN on massively parallel synthetic DNA libraries of Alternative 5'-splicing and Alternatively Skipped exon events, the model is capable of predicting the relative abundance of alternatively spliced mRNA isoforms on held-out library data to a very high accuracy ($R^2 = 0.77$ for Alt. 5'-splicing). Furthermore, the CNN is shown to generalize alternative splicing across cell lines efficiently. The Convolutional Neural Net is tested against a Logistic regression model and the results show that while prediction accuracy on the synthetic library is notably higher compared to the LR model, the CNN is worse at generalizing to new intronic contexts. Tests on non-synthetic human SNP genes suggest the CNN is dependent on the relative position of the intronic region it was trained for, a problem which is alleviated with LR.

The increased library prediction accuracy of the CNN compared to Logistic regression is concluded to come from the non-linearity introduced by the deep layer architecture. It adds the capacity to model complex regulatory interactions and combinatorial RBP effects which studies have shown greatly affect alternative splicing. However, the architecture makes interpreting the CNN hard, as the regulatory interactions are encoded deep within the layers. Nevertheless, high-performance modeling of alternative splicing using CNNs may still prove useful in numerous Synthetic biology applications, for example to model differentially spliced genes as is done in this paper.

Referat

En metod baserad på djupa neurala nätverk för att modellera regleringen av Alternativ Splicing


Uppsatsen jämför modellen med Logistic regression och drar slutsatsen att nätverkets förbättrade prestanda grundar sig i dess förmåga att modellera icke-linjära beroendena i datan. Detta medför dock svårigheter i att tolka vad modellen faktiskt lärt sig, eftersom interactionen mellan reglerande element är inbäddad i nätverkslagren. Trots det kan högpresterande modellering av alternativ splicing med hjälp av neurala nät vara användbart, exempelvis inom Syntetisk biologi där modellen kan användas för att kontrollera regleringen av splicing när man konstruerar syntetiska gener.
Definitions

**Massively Parallel DNA library:** A large collection of unique DNA sequences which have been sequenced (read) in parallel using Next-gen sequencing techniques.

**Diff. splicing:** The differential expression (i.e. the difference in ratio) of alternatively spliced RNA isoforms between cell types.

**Exon:** Nucleotide sequences which are part of the mature mRNA transcript and become translated.

**Intron:** Nucleotide sequences which are cleaved, or spliced, out of an mRNA transcript before translation.

**5’-end:** The left-most endpoint of a sequence which is transcribed from left to right.

**3’-end:** The right-most endpoint of a sequence which is transcribed from left to right.

**SD:** Splice Donor, the 5’-splice site of an intron.

**Motif:** A widespread and conserved nucleotide sequence which is conjectured to have some biological significance.

**RBP:** RNA Binding Protein, a protein which binds to specific sites on an RNA transcript.

**HEK:** Human Embryonic Kidney cells.

**HELA:** Cell line derived from human cervical cancer cells.

**MCF7:** Human breast cancer cells.

**CHO:** Chinese Hamster Ovary cells.

**LR:** Logistic Regression, a machine learning method.

**CNN:** Convolutional Neural Network, a neural network-based machine learning method.

**CE:** Cross-Entropy, a loss function commonly minimized in classification tasks.

**SNP:** Single-Nucleotide Polymorphism, a common DNA sequence variation in which a single nucleotide differs between members of a species.
1 Introduction

Alternative Splicing is an important and common mechanism during DNA transcription in eukaryotic cells which allows a gene to code for multiple proteins by alternatively cleaving out certain regions in the premature mRNA transcript. In human cells, more than 90% of the genes which have multiple exons are alternatively spliced[1]. This phenomenon greatly increases the diversity of the transcriptome and allows the human genome to code for many more proteins than there are protein-coding genes. The process is highly regulated by a large collection of trans-acting proteins which bind to target sites on the premature RNA transcript and either activate or repress the usage of splice sites. These regulatory interactions vary greatly and while efforts have been made in identifying many of them, researchers are still far away from having a model which completely explains and predicts the exact splicing behavior of any gene given the DNA sequence. Alternative splicing is important for researchers to understand, not only because it is a widespread and fundamental part of transcription, but also because misregulation of splicing has been linked to numerous human genetic disorders. Accurately predicting how genetic mutations influence splicing could prove useful in the study of such diseases.

The purpose of this report is to investigate how accurately predictive machine learning methods may model the (possibly non-linear) relationship between regulatory sequence elements and the choice of alternative splice site. The problem is addressed by developing a deep learning system based on Convolutional Neural Networks (CNNs). The CNN learns from a massively parallel library of alternatively spliced DNA containing randomized intronic regions. The report hypothesizes that the location-invariant, yet position-preserving properties of the convolutional layers will allow for accurately predicting the splicing isoform ratio given an input DNA sequence. At the same time, the number of free parameters will be constrained due to the sparse connectivity of the architecture. One of the splicing libraries considered in this report has previously been analyzed using Logistic regression by Seelig Lab at the University of Washington and the report aims to provide a comparative analysis between the developed CNN and the linear Logistic regression model.

Specifically, the following questions are investigated in this paper:

1. How well may a deep Convolutional Neural Network model alternative splicing on a large synthetic data set? How accurately can it predict isoform ratio and how well do predictions generalize across cell lines?

2. Given a CNN trained on synthetic data, how well does its predictions generalize to new, naturally occurring genes?

3. How well can the CNN predict the differential expression of splicing isoforms between cell lines?

The last question touches on an interesting extension of modeling splicing; can a forward engineering approach be taken for building differentially expressed splicing constructs? The applications of such constructs are many. For example, consider having a gene which is spliced in one way for cancerous cells and spliced another way for healthy cells. If such a gene is designed to lyse a cell if expressed in a particular isoform, one would have a way of designing precise cancer treatment delivery systems.
2 Background

In this section, the background and previous work related to this report is presented. First the mechanism of RNA splicing is reviewed. Next, Alternative Splicing is introduced and research concerning its regulation is summarized. Following that is a review of next-generation sequencing and how massively parallel DNA libraries may be built. Finally, the machine learning methods which will be employed in this study to model Alternative Splicing are described.

2.1 RNA splicing

Splicing is a modification of premature RNA where non-protein-coding regions of the transcript, introns, are cleaved out and the protein-coding regions, exons, are joined together. The process occurs shortly after or during gene transcription. Splicing is typically performed on all multi-exon messenger RNAs in eukaryotic cells and is required before the correct protein can be translated[1].

2.1.1 Mechanism

There are several different mechanisms of RNA splicing and their use depends on factors such as the cell type and the gene in question. Less common types include self splicing and splicing by the minor spliceosome.

This section will focus on splicing by the major spliceosome, which is the most common method. A simplified outline of the process is depicted in Figure 1. The major spliceosome is a protein complex made up of 5 different small nuclear ribonucleoproteins (snRNPs) which together catalyze splicing by binding to conserved subregions of the intron and forming the splicing complex. Several other auxiliary proteins are also involved in assembling the complex[2]. Major splicosomal splicing begins with one of the snRNPs, U1, binding to the highly conserved GU-sequence at the 5'-end (left-most part) of the intron. This sequence is referred to as the splice donor. Splicing Factor 1 (SF1) binds to an Adenine base within the intron, termed the branch point. U2AF1 binds to the highly conserved AG-sequence at the 3'-end (right-most part) of the intron, the splice acceptor. Additionally, the protein U2AF2 binds to a CU-rich region downstream of the branch point (called the polypyrimidine tract) and together these proteins form Splicing Complex E[3]. Next, the snRNP U2 binds to the branch point and pushes SF1 away (forming Complex A). Proteins U5, U4 and U2 then bind to the intron, reshaping the complex in such a way that the splice donor at
the 5’-end of the intron ligates to the branch point (Complex B-C). The intron is cleaved at the splice donor and a lariat (loop structure) is formed at the branch point. Finally, the intron is cleaved at the 3’ splice acceptor and the two exons are merged using ATP hydrolysis. The intron lariat is degraded in the nucleus while the splicing complex is recycled[2].

2.1.2 Alternative Splicing

Alternative splicing is a phenomenon encountered in many protein-coding genes within eukaryotic cells, where the alternative choice of splice site on a transcript allows for a single gene to express multiple mature mRNA isoforms. This alternative inclusion or exclusion of (possibly parts of) exons gives a gene the ability to code for many different proteins and so is a major source of transcriptome and proteome variability[1]. In fact, recent studies estimate that more than 90% of human multi-exon genes are alternatively spliced (Wang et al., 2008).

Alternative splicing events are typically divided into 5 general classes[3]:

1. **Exon skipping**: An exon is either entirely spliced out of or kept in the mature transcript. The exon is surrounded by introns which are always spliced out.

2. **Mutually exclusive exons**: Two exons are alternatively spliced in such a way that exactly one of them is always kept in the mature transcript.

3. **Alternative 5’ splicing**: The transcript contains two 5’ splice donors which compete for forming a lariat with the branch point and being spliced. If the splice donor closest to the branch point is chosen, the region between the splice donors is kept as an extension to the 5’ exon.

4. **Alternative 3’ splicing**: Instead of alternative splice donors, the transcript contains multiple 3’ splice acceptors which compete for being chosen as the splice site. This class of splicing alternatively includes an extension to the 3’ exon.

5. **Intron retention**: Intron retention works similar to Exon skipping, but without the alternatively spliced exon being surrounded by introns.

Exon skipping is by far the most common type of alternative splicing encountered in mammalian cells[4].

2.1.3 Regulation of Alternative Splicing

The mechanistic regulation of alternative splicing has been studied extensively the last few decades and researchers have found that the splicing outcome is highly dependent on numerous sequence-specific RNA binding proteins (RBPs), which either enhance or repress the usage of a splice site. While the field is still far away from having a complete explanatory model for reading the "splicing code" of genes, several studies have been conducted to identify many of the cis-regulatory elements (elements which recruit RBPs to the transcript)[2].

Splicing-regulatory elements are commonly divided into four groups according to whether they are located on an exon or an intron and whether they
repress or enhance the usage of a splice site, resulting in the groups ESE (Exonic Splicing Enhancer), ESS (Exonic Splicing Silencer), ISE (Intronic Splicing Enhancer) and ISS (Intronic Splicing Silencer)[4].

Studies have shown that Exonic Splicing Enhancer sites usually bind proteins from the Ser-Arg family. These proteins play an important role in what is called splice-site recognition or exon definition, which is the early stages of splicing when the spliceosome complex searches for the exon boundaries[2]. Ser-Arg proteins enhance splicing by recruiting the U1 snRNP to nearby 5’ splice donors as well as recruiting the U2 snRNP to nearby 3’ splice acceptors (CF Bourgeois et. al., 1999). They have also been found to cooperate with other RBPs by forming larger splice-enhancing complexes[5]. Intronic Splicing Enhancers are more varied and have more specificity than ESEs. For example, T cell-restricted intracellular antigen 1 (TIA1) has been found (Forch P et. al., 2002) to bind U-rich sequences downstream of 5’ splice donors and recruit U1 to its location. Other studies have identified various Ser-Arg proteins as intronic enhancers[6].

Splicing Silencers act oppositely to Splicing Enhancers by inhibiting the recognition of splice sites. Such inhibition may occur in many ways. First, silencer proteins may bind to sequence elements in close proximity to splice sites and physically block them[2]. An example is the Polypyrimidine-tract Binding Protein (PTB) which blocks spliceosome complex subunit U2AF (R Singh et. al., 1995) by binding to the polypyrimidine tract. Other tissue-specific factors such as FOX1 have been found (HL Zhou et. al., 2008) to block Splicing Factor 1 from binding to the branch point. Aside from blocking the actual spliceosome machinery, silencers can suppress enhancer proteins from binding. This can be done by physical blockage, but studies have identified several RBPs which suppress enhancers even when binding 100-200 bp away from them[7]. Furthermore, silencers may cooperate together and interact with numerous other proteins. This enormous network of combinatorial and positional interactions among cis-regulatory elements is the main reason as to why the "splicing code" is such a complex system to fully understand[8].

Traditional attempts at learning about alternative splicing, including the characterization of cis-regulatory elements, used to rely on the study of individual genetic pathways to identify conserved sequence elements. One such commonly studied pathway is the Drosophila Sex Determination (the Sxl and tra proteins). More recent approaches take advantage of high-throughput sequencing (reviewed in section 2.2) and quantitative analysis for determining cis-regulatory elements[9]. Alternative splicing is frequently studied in the field of Biomedicine, focusing on how misregulation of splicing affects human cells. In fact, it is known that at least 15% of all human genetic diseases are caused by near-splice site mutations, altering their use[10].

2.2 Massively parallel DNA analysis

Massively parallel sequencing, or Next-generation sequencing as it is also termed, is a set of high-throughput sequencing techniques which have emerged within the last few years and have allowed researchers to analyze large sets of genomic sequences in parallel. Using these techniques, synthetic libraries can be designed and used for quantitative analysis of many regulatory events. Such a library is used in this paper as the primary data set for analyzing alternative splicing.
2.2.1 Next-generation sequencing

Prior to 2004, the by far most widely used sequencing technique was Sanger sequencing, a method where chain-terminating dideoxynucleotides are successively added to a reaction containing the target DNA template, a DNA primer and DNA polymerase[11]. By adding chain-terminating bases of one type (A, C, G or T) in combination with ordinary, non-terminating bases, DNA fragments will stop being extended at different lengths. These fragments can then be separated using gel electrophoresis and the sequence can be read of the gel.

Next-generation sequencing techniques have emerged as highly cost-efficient methods for large-scale sequencing and, although these techniques differ from each other in implementation, they all share the paradigm of massive parallelization. In Sanger sequencing, each reaction works on a single target DNA template, while in Next-gen sequencing millions or even billions of different input fragments are captured on an array and sequenced in parallel[12].

A commonly used Next-gen sequencing platform is the Illumina MiSeq or HiSeq system, which uses a technique called Reversible termination for sequencing[12]. First, input DNA fragments are anchored to spatially separated adapters on a glass surface (the flow cell). The fragments are then clonally amplified into dense clusters. Finally, the sequencing takes place, where DNA polymerase incorporates fluorescently dyed reversible nucleotide terminators in the chain extension. By analyzing a temporal series of color images taken over the flow cell, the sequences can be determined from the dyed terminators.

2.2.2 Functional analysis of Alternative Splicing

This section reviews some of the most relevant research regarding the functional analysis of alternative splicing, which over the past decade has been fueled by the possibilities of high-throughput sequencing techniques and large-scale genomic analysis for quantitatively studying splicing events.

A study in 2008 conducted a global analysis of the human transcriptome by extracting the polyadenylated RNA from HEK293- and Ramos B cells and using randomly sampled Illumina deep sequencing (shotgun sequencing) to read out the entire RNA library[13]. By mapping the sequenced RNA to the human genome (hg18), the study was able to identify 94,241 splice junctions across the set of expressed mRNAs. Out of them, 4096 were previously unidentified. The study also concluded that Exon skipping is the most common type of alternative splicing within the human genome. Another genome-wide analysis mapped more than 2 million human mRNA and EST sequences (short RNA sequences) onto the human genome in order to characterize the alternative splicing isoforms from genes[14]. Many of the 6201 identified splice forms were novel and appeared to have biologically meaningful functions, for example modulating protein activity by removing specific targeting signals. Their results indicated that a large portion of all alternative splicing events in eukaryotes are concentrated to specific types of molecules or functions, e.g. the nervous- or immune system.

In more recent years, Machine Learning methods have been used to statistically model the splicing-regulatory interactions in eukaryotic cells. An example is the Bayesian Neural Network developed by Xiong et. al at the University of Toronto for predicting tissue-specific splicing-regulation from RNA-extracted features[15]. By processing EST- and cDNA libraries, 3665 Alterna-
tively skipped exons were obtained from 27 different mouse tissues. A set of 1014 hand-crafted features were extracted for each of the Skipped exon events, including short sequence motif counts, exon length and RBP binding site scores. Furthermore, target exon inclusion probabilities were estimated from the RNA library. The Bayesian network architecture uses the input RNA features to model a set of hidden variables which are later used to predict tissue-specific output response variables. The study defines a measure of Splicing code quality based on the log-likelihood of the data set and shows that, by this measure, their Bayesian network outperforms previous models in predicting tissue-specific splicing with an improvement of 52% compared to the state of the art. Furthermore, the Bayesian network achieves a 22% lower classification error when predicting whether exons are mostly included or excluded.

A later report by Leung et. al re-investigates splicing-regulatory events in mouse tissues using a Deep Neural Network and a data set of 11019 alternative exons retrieved from RNA-Seq data. Here, the network model uses not only hand-crafted intronic, exonic and structural RNA features, but also an index encoding of the input tissue types[16]. The model jointly predicts both the magnitude of the exon inclusion ratios for two input tissues as well as the relative difference in exon inclusion ratio. The output neurons encode indexes denoting if the exon inclusion ratios are Low, Medium or High and whether the difference in exon inclusion is Decremental, Incremental or if no change is predicted. The authors show that the deep network surpasses the performance of the Bayesian model using the previously developed Splicing code quality metric.

2.3 Machine learning methods

In this section, the machine learning methods which will be used for modeling alternative splicing are reviewed and discussed. Since the fundamental problem studied in this report is how a model may decide between different splicing outcomes, the section focuses on discriminative models.

2.3.1 Logistic regression

The first model discussed is that of Logistic regression. Consider the task of binary classification: An n-dimensional data point $\vec{x}'$ belongs to one of two possible classes $y' \in \{C_0, C_1\}$ and the task of a (trained) discriminative model $M$ is to predict the most likely class $y'$ for an unseen data point $\vec{x}'$. Logistic regression directly models the posterior probability $P(Y = y | \vec{x}, M)$ using the sigmoid as a link function from $\mathbb{R}^n \rightarrow [0,1]$ as follows[17]:

$$\hat{P}(Y = 0 | \vec{x}, w_0, \vec{w}) = \frac{1}{1 + e^{w_0 + \sum_i w_i \cdot x_i}}.$$

The Logistic regression model $M$ only consists of a scalar bias term $w_0$ and a weight vector $\vec{w}$. Furthermore, Logistic regression assumes the relationship between target class $y$ and input features $x_i \in \vec{x}$ is linear in the model’s weights $w_i \in \vec{w}$, as indicated by the linear combination $w_0 + \sum_i w_i \cdot x_i$ in the formula above. The model does however not necessarily need to be linear in the input features themselves; if non-linear basis functions $h_i(\vec{x}) \in \tilde{h}(\vec{x})$ are used, then the combination $w_0 + \sum_i w_i \cdot h_i(\vec{x})$ is still linear in the model’s weights[17]. Because $\hat{P}(Y = 0 | \vec{x}, w_0, \vec{w}) + \hat{P}(Y = 1 | \vec{x}, w_0, \vec{w}) = 1$, it implies that:
\[ P(Y = 1|\vec{x}, w_0, \vec{w}) = 1 - P(Y = 0|\vec{x}, w_0, \vec{w}) \]

For the training data set \( D = \{D_X, D_Y\} \), the Logistic regression model is optimized with regard to its weights \( w_0 \) and \( \vec{w} \) in order to maximize the log-likelihood of observing the training data[17]:

\[ l(w_0, \vec{w}) = \ln \hat{P}(D_Y|D_X, \vec{x}) = \{\vec{x}^j, y^j \in D \ \text{are I.I.D} \} = \sum_j \ln \hat{P}(y^j|\vec{x}^j, w_0, \vec{w}) \]

\[ = \sum_j y^j \ln \hat{P}(Y = 1|\vec{x}^j, w_0, \vec{w}) + (1 - y^j) \ln \hat{P}(Y = 0|\vec{x}^j, w_0, \vec{w}) \]  

(1)

The negative of the above likelihood function is called the Cross-Entropy error (CE-loss) and so minimizing the CE-loss is the same as maximizing the likelihood of the training data. Several algorithms exist for minimizing the CE-loss, the simplest and most straightforward one being Gradient Descent. The gradient of the CE-loss over the training data with respect to the weights is:

\[ \frac{\partial l(w_0, \vec{w})}{\partial w_i} = -\sum_j x^j_i * (y^j - \hat{P}(Y = 1|\vec{x}^j, w_0, \vec{w})) \]

In Gradient Descent, the weights \( w_0 \) and \( \vec{w} \) are updated with a small step \( \eta \) in the negative direction of the CE-loss gradient:

\[ w_i \leftarrow w_i - \eta \cdot \frac{\partial l(w_0, \vec{w})}{\partial w_i}, \forall i \]

More sophisticated learning algorithms like Coordinate Descent are sometimes used in regularized LR models. The concept of regularization is described next.

**Regularization**

Overfitting is an issue in Machine Learning where a model’s parameters are optimized against the training data to such an extent that it starts to describe more of the random noise or error in the training data rather than the underlying relationship of the data itself. Such a model will not generalize well to new, unseen data and will exhibit poor predictive power[17]. Several methods have been developed to prevent overfitting. Generally, the idea is to penalize overly large model parameters, a strategy called Regularization.

The most common technique is \( L_2 \)-Regularization, where a penalty term \( \lambda \cdot ||\vec{w}||^2_2 \) is introduced in the loss function. Consequently, minimizing the loss also trade-offs minimizing the magnitude of the weights[18]. Because the \( L_2 \)-norm is differentiable, standard Gradient Descent or Quasi-Newton methods can be used for optimization. The gradient of \( L_2 \)-regularized CE-loss over the \( N \) training points with respect to the model’s weights is:

\[ \frac{\partial l(w_0, \vec{w}) + \lambda ||\vec{w}||^2_2}{\partial w_i} = 2 \cdot \lambda \cdot w_i - \sum_{j=1}^N x^j_i \cdot (y^j - \hat{P}(Y = 1|\vec{x}^j, w_0, \vec{w})) \]  

(2)
Another penalizing strategy is $L_1$-Regularization. The $L_1$-penalty, defined as $\lambda * |\vec{w}|_1$, also penalizes large weights when included in the loss function. However, because of the hyper-geometrical properties of the $L_1$-norm it also promotes sparse solutions for $\vec{w}$, which can be an effective strategy at pruning non-correlated input dimensions[18]. Both Coordinate Descent- and Gradient Descent methods may be used for optimization, however in Gradient Descent special care has to be taken to induce sparsity when any component of $\vec{w}$ reaches zero (where the $L_1$-norm is not differentiable). The gradient of $L_1$-regularized CE-loss when $w_i \neq 0$ is:

$$\frac{\partial l}{\partial w_i}(w_0, \vec{w}) + \lambda |\vec{w}|_1 \partial w_i = \lambda * sgn(w_i) - \sum_{j=1}^{N} x^*_j * (y_j - \hat{P}(Y = 1|\vec{x}_j, w_0, \vec{w}))$$ (3)

**Logistic Regression & Bioinformatics**

Logistic Regression has been applied rigorously in Bioinformatics and Computational Biology for various types of sequence classification.

A 2012 study (Simcha et. al) broadly investigates different methodologies for de novo Transcription Factor (TF) motif discovery and prediction of gene expression profiles on the basis of such motifs[19]. One of the techniques investigated was that of Logistic Regression. Three non-synthetic data sets were considered in their experiments: Yeast expression profile clusters, human gene expression data and ChIP-chip TF binding data collected from various sequencing projects. A synthetic data set was also computationally generated using a zero-order Markov model with pre-determined motif insertion which allowed them to analyze motif classification performance on a "gold-standard" library.

The sets of gene expression profiles were partitioned into disjoint clusters of co-expressed genes and the classification task lied in discriminating between gene clusters on the basis of common sequence motifs. For the LR model, the hamming distance between a candidate sequence motif and all sequence motifs of a cluster were used as input. Evaluating the model on held-out data showed that while accuracy was significantly increased on the synthetic data set, it was still quite low with a mean AUROC (Area Under the ROC) of 0.677. The authors draw the conclusion that the lack of high-throughput, aligned heterogeneous data (where expression profile cluster boundaries are less noisy) was the main limitation in performance on the non-synthetic data sets, while employing more feature-rich input could potentially improve accuracy overall.

Logistic Regression has furthermore been used in medical research to identify disease from gene expression profiles. In particular, sparse feature selection by regularization has been extensively studied for the purpose of expression profile classification, as it is vital in such high-dimensional, low sample-rate data sets where the input features typically regard the entire expressed gene set of a sample. As an example, a study in 2004 (Zhou et. al) on cancer disease developed a sparse Logistic Regression classifier in order to predict cancer type from gene expression levels, using a bayesian sampling approach for regularization[20]. Several expression profile data sets for various types of cancer were trained and tested on, and the results showed that the model generalized well to new gene expression data, with high prediction accuracy across all data sets.

A similar study in 2012 by Liang et. al developed a sparse LR model for the purpose of predicting cell cancer types from gene expression data[21]. Again,
because the number of gene members (which is used as input) was much greater than the number of samples, a modified version of $L_1$-regularization was used to prevent overfitting. The model was trained using Coordinate Descent on four publicly available cancer gene expression data sets (Leukemia, Prostate cancer, Colon cancer and DLBCL). The reported results were very good, with an average test set error of 4.5%. The model outperformed other types of LR-based predictive models from similar studies, which according to Liang et al shows the importance of the choice of regularization. Furthermore, because the model obtained sparse solutions, the most influential genes could be identified. Literature studies confirmed that these genes were linked to cancerous disease, an indication that the model bases its decision on meaningful features.

2.3.2 Feed-forward Neural Networks

Logistic regression can be viewed as a specific instance of a more general set of models, namely Feed-forward Neural networks. A Feed-forward neural network layer consists of a set of $n$ inputs $\vec{x} = (x_1, ..., x_n)$, $m$ outputs (or activations) $\vec{a} = (a_1, ..., a_m)$, an $m \times n$-weight matrix $W = \{w\}_{ji}$ and intercept terms $\vec{b} = (b_1, ..., b_m)$[22]. A graphical representation of the architecture, the layer weights and their connection to the input layer is shown in Figure 2. The activations $\vec{a}$ are computed by multiplying the input vector $\vec{x}$ with the weight matrix $W$, adding the intercept $\vec{b}$ and applying a non-linear activation function $g(h)$:

$$a_j = g(h_j), \quad h_j = \sum_i w_{ji} x_i + b_j$$

The neural network can be trained to minimize a loss function $L(\vec{t}, \vec{a}, W, \vec{b})$, where $\vec{t}$ is a vector of target values for $\vec{a}$, using gradient descent[23]: Define

$$\frac{\partial L}{\partial w_{ji}} = \frac{\partial L}{\partial a_j} \frac{\partial a_j}{\partial h_j} \frac{\partial h_j}{\partial w_{ji}} = \left\{ \frac{\partial h_j}{\partial w_{ji}} = x_i \right\} = \delta_j \cdot x_i \quad (4)$$

$$\frac{\partial L}{\partial b_j} = \frac{\partial L}{\partial a_j} \frac{\partial a_j}{\partial h_j} \frac{\partial h_j}{\partial b_j} = \left\{ \frac{\partial h_j}{\partial b_j} = 1 \right\} = \delta_j \quad (5)$$

where $\delta_j = \frac{\partial L}{\partial a_j} \frac{\partial a_j}{\partial h_j}$ is the error signal from the activations. The weights are updated using the Stochastic Gradient Descent equations:

$$w_{ji} \leftarrow w_{ji} - \eta \cdot \frac{\partial L}{\partial w_{ji}}, \quad b_j \leftarrow b_j - \eta \cdot \frac{\partial L}{\partial b_j} \quad (6)$$
In the case of binary Logistic regression, the model is simply a 1-layer, single activation unit neural network using the sigmoid as an activation function. I.e

\[
\hat{P}(Y = 1 | \vec{x}) = a = g(\sum_i w_i x_i + b) = \frac{1}{1 + e^{-\sum_i w_i x_i + b}}
\]

Using the Batch Cross-Entropy loss

\[
L = -\sum_j y_j \ast \ln(a^j) + (1 - y_j) \ast \ln(1 - a^j),
\]

the error signal \( \delta \) becomes:

\[
\delta = \frac{\partial L}{\partial a} \ast \frac{\partial a}{\partial h} = -(y - a) \ast a \ast (1 - a) = -(y - a) \tag{7}
\]

Inserting this value for \( \delta \) into the expressions for \( \frac{\partial L}{\partial w_{ji}} \) and \( \frac{\partial L}{\partial b} \) results in the exact same loss gradients as presented in section 2.3.1 on Logistic regression.

Multi-Layer Feed-forward Networks

The single-layer network described above (including the Logistic regression network model) only models linear relationships between the \( n \)-dimensional input vector \( \vec{x} \) and activations \( \vec{a} \). To model non-linear relationships, multiple layers of non-linear activational units are required. In fact, it can be shown that a Feed-forward network consisting of three layers (including the final output layer) can approximate any function up to an arbitrarily high accuracy (dependent on the number of units in each layer)\[22\]. The network notation can therefore be extended to have an \( m^k \times m^{k-1} \)- weight matrix \( W^k = \{w\}^k_{ji} \) and bias vector \( \vec{b}^k = (b^k_1, ..., b^k_{m_k}) \) for each network layer \( k \), where \( m^{k-1} \) is the number of activation units in layer \( k - 1 \) and \( m^k \) is the number of activation units in layer \( k \) (\( m^0 = n \))[22]. Here superscripts (such as \( k \) in \( m^k \)) denote indices rather than powers. The network architecture is depicted in Figure 3 below.

![Multi-Layer Feed-forward Network Diagram](image)

Figure 3: A multi-layer feed-forward network. The activations from previous layers are passed as input to successive layers. At each layer, the inputs are multiplied with the layer weights and fed through the activation functions[22].

Each layer \( k \) computes its activations \( \vec{a}^k (\vec{a}^0 = \vec{x}) \) according to:
\[ a_j^k = g(h_j^k) \], where \( h_j^k = \sum_i w_{ji} a_i^{k-1} + b_j^k \) \hspace{1cm} (8)

The last layer of activation units is called the Output layer, while the previous layers of activation units are called Hidden layers (and activation units are called Hidden units). Typical choices for the activation function \( g(z) \) are the sigmoid or the hyperbolic tangent[23]. Using the notation from the previous section, the error signals \( \delta^K \) for the output layer are defined as:

\[ \delta_j^K = \frac{\partial L}{\partial a_j^K} \ast g'(h_j^K) \] \hspace{1cm} (9)

where \( L = L(\vec{t}, \vec{a}) \) is some loss function defined for the vector of output layer activations \( \vec{a} = \{a_j^k\} \) and the vector of target values \( \vec{t} = \{t_j\} \). For the hidden layers, the error signals \( \delta^K \) are recursively defined to be:

\[ \delta_j^k = \sum_{u=1}^{m_{k+1}} \delta_{u}^{k+1} \ast w_{uj}^{k+1} \ast g'(h_j^k) \] \hspace{1cm} (10)

To train multi-layered feed forward networks, a generalization of the Gradient Descent algorithm called Backpropagation of error is used. The algorithm consists of first computing the activations \( \vec{a}^K \) of each network layer, and then recursively computing the error signals \( \delta^K \) backwards through the layers[23]. After the error signals have been computed for each layer, the Stochastic Gradient Descent update equations can be applied as usual:

\[ w_{ji}^k \leftarrow w_{ji}^k - \eta \ast \frac{\partial L}{\partial w_{ji}^k}, b_j^k \leftarrow b_j^k - \eta \ast \frac{\partial L}{\partial b_j^k} \] \hspace{1cm} (11)

where \( \frac{\partial L}{\partial w_{ji}^k} = \delta_j^k \ast a_i^{k-1} \) and \( \frac{\partial L}{\partial b_j^k} = \delta_j^k \).

Training is usually stopped when the mean loss on some held-out validation data, as defined below, starts increasing (where \( \vec{a}_{out}(\vec{x}) \) is the output of the neural network for input \( \vec{x} \)):

\[ \tilde{L}(D_{valid}) = \frac{1}{|D_{valid}|} \sum_{\{\vec{x}, \vec{t}\} \in D_{valid}} L(\vec{t}, \vec{a}_{out}(\vec{x})) \]

Note that in these types of feed-forward networks, each layer is fully connected to the activation units of the previous layer, i.e. there is a weight \( w_{ji}^k \) between each unit \( i \) of layer \( k-1 \) and each unit \( j \) of layer \( k \).

### 2.3.3 Deep Neural Networks

Deep Feed-forward neural networks, i.e. networks with many connected layers of activation units, have more expressive power than shallower network architectures and can potentially model non-linear relationships in the data using fewer units in total compared to an equally well-performing shallow network. In fact, a shallow network may require exponentially as many activation units as a deeper network would in order to approximate certain functions[24].

11
However, the training of deep feed-forward networks comes with several issues. The first and foremost difficulty during training is the so-called Vanishing gradient problem. Sepp Hochreiter shows in his 1991 thesis that the error signals $\delta^k$ shrink exponentially as they are backpropagated through the layers[25]. This can be seen by observing the recursive definition of $\delta^k$: At each layer $k$, $\delta^k_j$ is multiplied by the derivative of the activation function, $g'(h^k_j)$. For activation functions such as the sigmoid or the hyperbolic tangent, $g'(h^k_j)$ is always smaller than 1, resulting in an exponential decrease due to repeated multiplication of this term. One proposed solution to this problem is to pre-train each network layer on its own using unsupervised learning. A different technique which has shown much promise in deep learning is to switch the activation functions of the network to Rectified Linear Units (ReLUs), defined as $g(x) = \max(x, 0)[26]$. The gradient $g'(x)$ is either 0 or 1, which promotes network sparsity when $x \leq 0$ and eliminates the Vanishing gradient problem when $x > 0$.

Another issue with deep networks regards the large number of hyperparameters that needs to be tuned during training (the number of layers, the number of units in each layer, the learning rate for each layer and the initial weights), and the choice of them can have a large impact on the final performance. Cross-validating to find the optimal parameters is often infeasible due to the size of the parameter space and the time it takes to train the network each time, however this problem has lately become somewhat relieved with the ability to speed up network training using the large processing power of GPUs[27].

Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a special type of deep neural network which differs quite a bit from the fully connected feed-forward networks discussed so far. CNNs exploit a sparse connectivity architecture combined with weight sharing for units within the same network layer, replicating the behavior of receptive field cells in the visual cortex. This allows neurons in the network to respond to patterns in the input data regardless of exact position (they are locally invariant). CNNs have been shown to perform excellently on various visual recognition tasks, a famous example being the LeNet-5 model which was able to classify handwritten digit images from the MNIST dataset to a very high accuracy[28]. The LeNet-5 model is shown in Figure 4.

![Illustration of the LeNet-5 CNN](image)

Figure 4: Illustration of the LeNet-5 CNN, as depicted in LeCun’s classic paper "Gradient-based learning applied to document recognition"[28].

Typically, A CNN consists of a number of Convolutional layers connected in sequence to a fully connected feed-forward output layer. The task of the convolutional layers is to learn an abstract representation of the input signal (which
is typically a 2D image, but may very well be a 1D signal) which can be fed to
the fully connected layer as input features. Each convolutional layer produces
a set of output signals or feature maps (which are 2D if the input signal is 2D)
that are used as input signals to successive layers[29].

Consider the case when the input is an \( n \times m \) 2D signal \( X = \{ x_{ij} \} \). Each
convolutional layer \( l \) has \( K_l^{l-1} \times K_l \) weight matrices (called kernels or filters)
\( W^{l,k_l^{l-1}} = \{ w_{ij} \}^{l,k_l^{l-1}} \) of size \( u_l \times v_l \) and \( K_l \) intercept biases \( b^{l,k_l} \), where
\( K_l^{l-1} \) is the number of output feature maps at layer \( l - 1 \) and \( K_l \) is the number
of feature maps at layer \( l \). This means there is a unique kernel for each input
feature map - output feature map combination. The activation (or output) maps
\( A^{l,k_l} \) of each convolutional layer \( l \), for each kernel \( k_l \) of the layer, are computed by
convolving the input signals \( A^{l-1,k_l^{l-1}} \) from the previous layer with each kernel,
adding the bias of each kernel and applying a non-linear activation function
\( g(x) \)[29]:
\[
\begin{align*}
  a^{l,k_l}_{ij} &= g(h^{l,k_l}_{ij}), \\
  h^{l,k_l}_{ij} &= \sum_{k_l^{l-1}} (W^{l,k_l,k_l^{l-1}} \otimes A^{l-1,k_l^{l-1}})_{ij} + b^{l,k_l}
\end{align*}
\]
(12)
where \( A^{0,1} = X \), and the convolution operator \( \otimes \) is defined as:
\[
(f_{n \times m} \otimes g_{n \times m})_{ij} = \sum_{u=i-n/2}^{i+n/2} \sum_{v=j-m/2}^{j+m/2} f_{u,v} \ast g_{u-v,m-v}
\]
It is common to reduce the dimensionality of the output feature maps of each
convolutional layer by applying a non-linear down-sampling. This provides
a form of translation invariance to the input for later layers, whose neurons
may then cover a greater receptive field. Common down-sampling techniques
are MaxPooling and MeanPooling. MaxPooling works by partitioning the feature
map into small grids and outputing only the maximum value of each grid
square[28]. The final output of each convolutional layer is hence:
\[
\begin{align*}
  o^{l,k_l}_{ij} &= \maxpool(A^{l,k_l})_{ij}
\end{align*}
\]
(13)
The final layer (or layers) of CNNs is typically fully connected feed-forward
networks, whose activations are computed exactly as described in the previous
sections.

Similar to other feed-forward neural networks, CNNs are trained using Back-
propagation of error and Gradient Descent. For the fully connected layers of
the CNN, the error signals \( \delta_l \) for each layer \( l \) are computed exactly as described
in the previous sections. For each convolutional layer \( l \) and output feature map
\( k_l \), the error signals \( \delta^{l,k_l} \) (which are matrices) are computed as[29]:
\[
\begin{align*}
  \delta^{l,k_l} &= \text{upsample}(\sum_{k_l^{l+1}} W^{l,k_l^{l+1},k_l} \times \delta^{l+1,k_l^{l+1}}) \ast g'(H^{l,k_l})
\end{align*}
\]
(14)
where \( H^{l,k_l} = \{ h_{ij} \}^{l,k_l} \) is the convolved input feature map used as input
to compute the activation \( A^{l,k_l} \) for layer \( l \) and output feature map \( k_l \). The function \text{upsample}() is the inverse operation to the pooling function used in
the convolutional layer, i.e. it upsamples the \( \delta^{l,k_l} \) matrix to the dimensionality
of the output feature map before it was pooled. If MaxPooling is used, the
upsampling operation simply writes each error signal to the position of each unit which was chosen as max, while zero is written to the positions of the non-chosen units. Also note that the last convolutional layer will receive its forward error signal $\delta^{l+1,k+1}$ (which is used in the recursive computation of $\delta^{l,k}$) from the fully connected output layer(s).

Finally, the gradient of the loss function with respect to each kernel $W^{l,k,l-1}$ and each intercept $b^{l,k}$ is computed as:

$$\nabla_{W^{l,k,l-1}}L = A^{l-1,k,l-1} \otimes \text{rot90}(\delta^{l,k})$$

$$\frac{\partial L}{\partial b^{l,k}} = \sum_{i,j} (\delta^{l,k})_{ij}$$

where rot90() rotates the input matrix by 90 degrees and $\otimes$ is the convolution operator. These gradients can then be used in a Gradient Descent optimization procedure to train the convolutional layers similar to how fully connected feed-forward network layers are trained.

3 Methodology

The fundamental problem investigated in this report is how Alternative Splicing may be accurately modeled and predicted given only the DNA sequence. To this end, a deep learning methodology based on Convolutional Neural Networks is developed whose machine learning task is to predict splicing isoform ratios from input sequences. This type of model is suitable mainly for two reasons: First, a high-capacity model which can learn non-linear functions between sequence elements is desirable in order to maximize prediction accuracy. At the same time, the sparse connectivity of the CNN can substantially reduce the number of free parameters. Second, the convolutional layers provide the model with both location-invariance and position-specific preservation, which could be very beneficial for detecting important regulatory elements across the sequence.

The network is trained using massively parallel DNA libraries of alternative splicing events. Questions we want to ask about this model are: How well does it predict splicing isoform ratios on new, previously unseen DNA sequences of the data set? And, how well does the model generalize across cell lines? Also, as the reader will become aware, the DNA sequences of the data sets are all synthetic alterations based off the same original gene. An important consideration is how well the model generalizes splicing prediction for completely different genes.

In this section the models and methods developed are presented, starting by introducing the synthetic DNA libraries and their structure. Next, the machine learning methodology is described, focusing mainly on the Convolutional Neural Network architecture. However, a Logistic regression model which will be used for performance comparisons with the CNN is developed as well. The LR model is replicated from previous work conducted on one of the synthetic DNA libraries, originally done by Alexander Rosenberg at Seelig Lab.

Finally, as an extension of the methods developed here, the problem of finding differentially expressed alternative splicing isoforms across cell lines is investigated. To this end, a search method is developed which utilizes CNNs trained on cell line-specific data in order to find maximal differential splicing.
3.1 Data sets

This section describes the DNA library data sets used in the analysis. Two types of Alternative Splicing is investigated in the report: Alternative 5’-splicing and Alternatively skipped exons.

3.1.1 5’-alternative splicing library

The data set used for analyzing Alternative 5’-splicing is a massively parallel DNA library constructed by Seelig Lab at the University of Washington. It has previously been analyzed using Logistic regression. The library was constructed by introducing two fully degenerate (mutagenized) 25 base-pair regions into the intron of a small mini-gene (Citrine fluorescent protein), giving rise to a large set of mutated gene-variants. The library is shown in Figure 5 below.

The intron separating the two exons of the Citrine gene contains three 5’ splice donors, each of which competes for forming a lariat with the branch point and splicing according to the process described in section 2.1.1. The two randomized regions lie in between the 5’ splice donors without overlapping them. The gene system can lead to 4 distinct RNA isoforms as a result of splicing at either of the splice donors (SD$_1$, SD$_2$ or SD$_3$) or not splicing at all. Throughout the rest of the methodology section, the splice donor usage ratios for SD$_1$, SD$_2$ and SD$_3$ are for each library member $j$ denoted $\vec{P}(j) = (P_1(j), P_2(j), P_3(j))$.

Seelig Lab synthesized the library by transfecting and growing the mini-gene into cells, after which the resulting DNA and expressed RNA was extracted and sequenced in parallel (reviewed in section 2.2.1). The RNA reads could then be associated to the originating DNA sequence and the isoform ratio of each splicing outcome could be quantified for every mini-gene variant. Furthermore, the library was constructed for a range of different cell lines. In this report, library instances for the cell lines HEK293, HELA, MCF7 and CHO are used. The final library contains 265,000 unique gene variants and has on average 54.4 RNA reads per member and cell line. Because the randomized regions are located within the intron of the mini-gene, the variability of isoform ratio among library members solely comes from introducing new binding sites for Intronic Splicing Enhancers and Silencers (as defined in section 2.1.3), or from introducing new splice sites.

Figure 6a-6c displays histograms of the usage ratio for each splice donor among the HEK library members, showing that the expressed isoform ratios are quite diverse. The library instances of other cell lines are similarly distributed.
Figure 6: (a) - (c) The distribution of splice donor usage ratios across the HEK library instance (the histograms are thresholded at 20,000) and (d) the distribution of the maximum differential expression of isoforms spliced at $SD_1$ for library members across HEK, HELA, MCF7 and CHO.

Furthermore, there is notable variability in the differential expression of isoforms across cell lines, as shown in Figure 6d. The histogram plots the maximum differential expression of $SD_1$ usage between cell lines HEK, HELA, MCF7 and CHO. Maximum diff. expression of $SD_1$ is for a sequence $j$ defined as $\Delta \bar{P}_1(j) = \max(\bar{P}_1(j)) - \min(\bar{P}_1(j))$, where $\bar{P}_1(j) = (P_{1_{HEK}}^1, P_{1_{HELA}}^1, P_{1_{MCF7}}^1, P_{1_{CHO}}^1)$.

### 3.1.2 Skipped exon library

The second synthetic DNA library analyzed in this report regards Alternatively skipped exons (see section 2.1.2 for a definition). It too was constructed by Seelig Lab at the University of Washington. The library is illustrated in Figure 7. Two 25bp randomized regions are introduced in the introns surrounding an alternatively spliced-out exon. The regions do not overlap any splice- donor or acceptor. As shown in the figure, only two splicing outcomes are possible: a shorter isoform where the exon is spliced out together with the introns, or a longer isoform where only the introns are spliced.
Each library member $j$ has an exon-inclusion ratio $P_1(j)$ and an exclusion ratio $P_0(j) = 1 - P_1(j)$. Instances of the library exist for three different cell lines: HEK, HELA and MCF7. The library contains 1,400,000 unique members. Figure 8a plots a histogram of the exon inclusion ratio for every library member of the HEK instance, while Figure 8b plots a histogram of the Maximum differential expression between cell lines (defined in section 3.1.1). As can be seen, the histogram peaks are much more accentuated and less dispersed than in Figure 6, which is due to the Skipped exon library having fewer RNA sequence reads per library member (2.0 reads on average per member and cell line) than the Alternative 5’-splicing library (54.4 reads on average).

3.2 Logistic regression model

Logistic regression is a linear model used to predict a response variable in the range $[0, 1]$, often thought of as the probability of a data point belonging to some target class. This type of model is directly applicable to our task of modeling splicing isoform ratio: The target ratios can be considered the relative frequency, or probability, of a sequence being spliced at one of the splice donors.

LR was introduced in section 2.3.1 and in section 2.3.2 it was conceptualized as a single-layer feed-forward network. The feed-forward network notation will be used here to describe the model architecture. The LR model will serve as a baseline for performance comparisons with the CNN model (presented in section 3.3). This is interesting because Logistic regression is literally the least complex
type of neural network while a deep CNN is one of the most complex ones. Developing an LR model thus allows us to observe if the added complexity of a deep CNN brings more explanatory power to alternative splicing than a simpler network. The developed LR model is similar to the one previously built by Seelig Lab for analyzing the Alternative 5'-splicing library and a comparative analysis is a good way of extending the machine learning aspect of their work.

For this part of the analysis, only the first splice donor (SD₁) usage ratio is regressed in the Alt. 5'-splicing library. I.e. we do binary logistic regression where a sequence splices at SD₁ with ratio $P₁$ or anywhere else with ratio $1 - P₁$. Obviously, the regression is binary for the Skipped exon library as well.

3.2.1 Feature representation

Before doing regression, the library DNA sequences must first be transformed into a suitable representation of numerical features. Since the model should be identical to the one Alexander Rosenberg at Seelig Lab used in his analysis of the Alt. 5'-splicing library, the feature extraction which he developed is used.

For input sequence $S(i)$, the features are taken to be the occurrence count of each n-mer motif in the randomized regions of the sequence. To enforce some location specificity, we separate the count of motifs for each randomized region. The model consequently assumes that the exact location of a motif within each 25bp randomized region is irrelevant. Formally, for each sequence $S_C^C(i)$ in the library $D_C$ of cell line $C$, perform the following transform:

$$\vec{x}_i = (c^k_{AAA...}, ..., c^k_{TTT...}, c^k_{AAA...}, ..., c^k_{TTT...})$$

$c^k_{XXX...} = \text{count of subsequence XXX... in randomized region } k.$

With regard to the size of the DNA libraries, the length of n-mers considered is set to $n = 6$. The input dimensionality of the LR model becomes 8192 ($4^n$ motif counts per randomized region). Note however that the input vectors are extremely sparse; each sequence can have at most 20 unique motif occurrences per randomized region. The feature extraction is illustrated in Figure 9.

Figure 9: The occurrence counts of 6-mer subsequences within each randomized region are used as features for the Logistic regression.

To motivate this feature extraction, consider instead the most basic representation where a small identity vector $\vec{e} = (e_A, e_C, e_G, e_T)$ is constructed for every base-pair in the sequence, encoding a 1 for the base at the current position. The complete input $\vec{x}$ is the concatenation of all base-pair identity vectors. The model would from the training data learn about important sequence motifs by increasing the corresponding base-pair weights at specific positions. Then, if the same motifs reappear in new data but are shifted by as little as 1bp, the model will likely perform poorly if the specific arrangement has not occurred during training. This is completely alleviated by Rosenberg’s feature representation.
3.2.2 Model architecture

The architecture of the LR model is depicted in Figure 10. It is a single-layer, single-output unit feed-forward network with activation function \( g(z) = \frac{1}{1+e^{-z}} \). For each 8192-dimensional input vector \( \vec{x}_j \) (the feature vector of sequence \( j \)), the network approximates the usage ratio of \( SD_1, P^C_1(j) \) as:

\[
\hat{P}^C_1(j) = \frac{1}{1 + e^{-\sum_i w_i \times x_{ji} + b}}
\]

where \( \vec{w} = (w_1, ..., w_{8192}) \) is the weight vector and \( b \) is the intercept bias. The model consists in total of 8193 free parameters.

3.2.3 Training

The model is trained by minimizing the L2-regularized Cross Entropy loss as defined in section 2.3.1 (Equation 1-2), using batch gradient descent. In neural network notation, the unregularized loss over the data set \( D^C = \{(\vec{x}_j, P^C_1(j))\} \) is defined as:

\[
L(D^C) = -\sum_j P^C_1(j) \ast \ln(\hat{P}^C_1(j)) + (1 - P^C_1(j)) \ast \ln(1 - \hat{P}^C_1(j))
\]

The error signal \( \delta(j) \) (see section 2.3.2) for a data point \( (\vec{x}_j, P^C_1(j)) \) with regard to Cross Entropy loss was derived in Equation 7 to be:

\[
\delta(j) = \frac{\partial L(D^C)}{\partial \hat{P}^C_1(j)} \ast \frac{\partial \hat{P}^C_1(j)}{\partial \sum_i w_i \ast x_{ji} + b} = -(P^C_1(j) - \hat{P}^C_1(j))
\]

The gradient of the regularized loss with respect to \( \vec{w} \) and \( b \) are computed according to the batch equivalent of Equation 4-5:

\[
\frac{\partial (L(D^C) + \frac{1}{2} \ast \lambda \ast ||\vec{w}||_2^2)}{\partial w_i} = \lambda \ast w_i + \sum_j \delta(j) \ast x_{ji}
\]

\[
\frac{\partial (L(D^C) + \frac{1}{2} \ast \lambda \ast ||\vec{w}||_2^2)}{\partial b} = \sum_j \delta(j)
\]

The weights are updated with a small step in the negative direction of the loss gradients (Equation 6). This is done until the loss function converges. The optimization is performed using the Python library Scipy[30]. Specifically, the library’s implementation of the Limited Memory BFGS quasi-newton line search is used. The BFGS procedure is given the loss function and gradients and then finds the minimum of the loss. Training is done on 90% of the data set, while 10% is kept for testing. The regularization parameter \( \lambda \) is set using 10-fold cross-validation on the training data. The 8192-dimensional input vectors are stored in sparse matrix format throughout the computation.
3.3 Convolutional Neural Network model

The primary model investigated in this report is a deep Convolutional Neural Network. CNNs were introduced at the end of section 2.3.3 and the same notation will be used here to describe the architecture and how training is done.

3.3.1 Feature representation

Different from fully connected neural networks, CNNs operate on what should be considered a signal stream rather than a feature vector. That is, fully connected neural nets consist of activation units which are bound to all inputs of the feature vector. Every unit has a weight which is specific to each feature in the input. Convolutional layers on the other hand, as described in 2.3.3, utilize weight sharing by sliding a small (trainable) filter of weights across the input vector (or 2D input map, as CNNs are often used on images) and convolving each overlaid region of input with the filter.

The concept of weight sharing is very appealing to our task of learning about regulatory elements contained in the randomized regions of the DNA library; we require a model which can recognize regulatory elements regardless of exact position. Position should only play a minor role in prediction. Fortunately, this is exactly what the convolutional layers provide: The filters will slide across the input DNA sequence, responding to specific motifs they have been trained to be sensitive towards. However, every overlaid region of the input signal relates to a specific location in the convolved output signal, meaning positional information remains preserved. With these properties in mind, we encode the features as a 2D-map of base-pair occurrences (illustrated in Figure 11).

For every sequence $S^C(j)$ in the library $D^C$ of cell line $C$, construct for each randomized region $r$ a 2D input signal $X^r_j$ according to:

$$(X^r_j)_{ik} = 1, \text{ if } R^r_i = b_k$$

$$(X^r_j)_{ik} = 0 \text{ otherwise}$$

where $R^1$ is the first randomized region, $R^2$ is the second randomized region and $\vec{b} = (b_1, b_2, b_3, b_4) = (A, C, G, T)$. The randomized regions are padded with 5 bps on each side from $S^C(j)$, turning each input signal $X^r_j$ into size $4 \times 35$.

3.3.2 Model architecture

The CNN network design is fairly complex in comparison to the LR network presented in section 3.2, consisting in total of 7 hierarchies with separate spatial columns in the first two layers. The first three layers are shown in Figure 12.

Layer 1 consists of two separate convolutional layers ordered as columns, which are fed the input signal $X^1_1$ of the first randomized region and $X^2_2$ of the second randomized region respectively. The activations of the two columns are computed according to Equation 12 in section 2.3.3. The height of the
kernels \( W^{1,k,1} \) of Layer 1 is set to 4, meaning they will occupy the entire height of the input signal and produce a 1D output signal post-convolution. The width, which effectively decides how long motifs the model will consider, is determined by cross-validation. Also, keep in mind that the layer has \( K^1 \) different kernels and so will produce \( K^1 \) output signals to the next layer. Layer 2 also consists of two separate convolutional columns, each of which is fed the set of output activations \( A^{1,k^1} \) from the respective previous column of Layer 1. As shown in the figure, the input signals are 1D and the kernels \( W^{2,k^2,k^1} \) must then have height 1. Similar to Layer 1, the kernel width is tuned with cross-validation. Before sending the output to Layer 3, the activation signals from the two columns are concatenated horizontally, producing one single output signal. Layer 3 is a MaxPooling layer (reviewed in 2.3.3). It receives the set of concatenated activations \( A^{2,k^2} \) from Layer 2 and outputs a subsampled signal according to Equation 13. The MaxPooling operation takes the maximum value from small subregions of the original signal. Consequently, high-magnitude activations which are spatially far apart are brought closer together, further improving location invariance and providing a kind of summative abstraction. The subregion size (which max-values are chosen from) is set to 2.

Figure 12: The first three layers of the CNN model used to predict Alt. splicing isoform ratio, including two convolutional layers and one MaxPooling layer.

Layers 4-7, which are described next, are shown in Figure 13. Layer 4 is the last convolutional layer of the network. Similar to Layer 2, it takes the activations \( A^{3,k^3} \) from the previous layer (which was a MaxPooling layer) and convolves the signal with 1D kernels. However, since the activations were concatenated after Layer 2, there is only one column in this layer. Layer 5 is yet another 2-sized MaxPooling layer, effectively halving the size of the activations \( A^{4,k^4} \) by subsampling. Layer 6 is the first fully connected layer of the model. It is a standard feed-forward network as defined in section 2.3.2. The layer has a weight \( w_{ji} \) between each hidden unit \( j \) and activation unit \( (A^{5,k^5})_i \) of the previous layer and computes its activations \( \vec{a}_6 \) according to Equation 8. Also, remember that Layer 5 produces \( K^5 \) different output signals \( A^{5,k^5} \) (due to multiple weight kernels), so the actual input to the hidden layer is the concatenation of all \( K^5 \) outputs. The number of hidden units is determined by cross-validation.
Finally, the output layer of the network (shown in Figure 14) is a standard multinomial Logistic regression layer.

It takes the activations $a^6$ from the hidden layer as input and, if analyzing the Alternative 5’-splicing library, estimates the vector of splice donor usage ratios $\hat{P}_C^C(j) = (\hat{P}_1^C(j), \hat{P}_2^C(j), \hat{P}_3^C(j))$ by applying the softmax function (which is the multinomial extension to the logistic, or sigmoid, function):

$$\hat{P}_k^C = \frac{e^{-\sum_i w_{ki}a^6_i + b_k}}{\sum_v e^{-\sum_i w_{vi}a^6_i + b_v}}$$

For the Skipped exon library, which only has the exon inclusion ratio $\hat{P}_1^C(j)$ and exclusion ratio $1 - \hat{P}_1^C(j)$ as output, binary Logistic regression is done by applying the classical sigmoidal activation function ($g(z) = \frac{1}{1+e^{-z}}$).

There are a couple of design choices regarding the network architecture worth discussing, the first and foremost one being the layer configuration. Cross-validating in order to find the optimal combination of convolutional- and MaxPooling layers becomes infeasible when simultaneously cross-validating other parameters. Instead, the model was developed in a test-driven way by trying a few layer alterations and choosing the configuration which performed best on validation data (separate from the test data). Having Layer 1 not be Max-Pooled before connecting to the next convolutional layer is an example alteration which made a large impact on performance. After Layer 1, adding more than two convolutional- and MaxPooling layers made performance suffer, probably due to the resulting signal becoming too short for the fully connected layers.
The non-linear activation functions used in the convolutional layers are Rectified Linear Units (discussed in 2.3.3). This halved the training time while not lowering model performance compared to when using hyperbolic tangents.

Finally, it is worth discussing how cross-validation was done. Even when not considering the number of convolutional layers, the amount of hyper-parameters is very large. For each of the three convolutional layers, the number of kernels \( k^l \) as well as the kernel width \( w^l \) must be tuned. Also, the number of hidden units in Layer 6 has to be set. Cross-validating 7 parameters simultaneously for a network which takes hours to train is not realistic and so a greedy approach was taken: Starting from initial parameters which worked seemingly well on validation data during development, each parameter was individually cross-validated over a suitable range while the other ones remained fixed. This reduced the cross-validation time complexity from \( O(n^7) \) to \( O(n) \) in the number of parameter-choices. 4-fold cross-validation was performed for all parameters. Table 1 and 2 lists the final cross-validated parameters for the number of kernels \( (k^l) \) and the kernel width \( (w^l) \) for the Alternative 5'-splicing library and Skipped exon library respectively. The tables also list the resulting input- \( (A^{l-1}) \) and output \( (A^l) \) sizes for each of the cross-validated layers.

### Table 1: The cross-validated hyper-parameters of the Alt. 5'-splicing CNN. For Layer 6, which is fully connected, \( k^l \) denotes the number of hidden units.

| Layer \((l)\) | \( k^l \) | \( w^l \) | \( |A^{l-1}| \) | \( |A^l| \) |
|-------------|--------|--------|----------------|----------------|
| 1           | 30     | 6 × 4  | 2 × 35 × 4     | 2 × 30 × 1     |
| 2           | 40     | 5 × 1  | 2 × 30 × 1     | 2 × 26 × 1     |
| 4           | 50     | 5 × 1  | 1 × 26 × 1     | 1 × 22 × 1     |
| 6           | 50     | 50 × 11| 50             | 50             |

### Table 2: The cross-validated hyper-parameters of the Skipped exon CNN. For Layer 6, which is fully connected, \( k^l \) denotes the number of hidden units.

| Layer \((l)\) | \( k^l \) | \( w^l \) | \( |A^{l-1}| \) | \( |A^l| \) |
|-------------|--------|--------|----------------|----------------|
| 1           | 40     | 4 × 4  | 2 × 35 × 4     | 2 × 32 × 4     |
| 2           | 60     | 5 × 1  | 2 × 32 × 4     | 2 × 28 × 4     |
| 4           | 70     | 5 × 1  | 1 × 28 × 4     | 1 × 24 × 4     |
| 6           | 200    | 70 × 12| 200            | 200            |

Reading from Table 1, the Alternative 5'-splicing model has in total 51333 free parameters (1500 in Layer 1, 12080 in Layer 2, 10050 in Layer 4, 27550 in Layer 6 and 153 in the output layer). The Skipped exon model has 214951 free parameters (1360 in Layer 1, 24120 in Layer 2, 21070 in Layer 4, 168200 in Layer 6 and 201 in the output layer).

### 3.3.3 Training

The CNN network is trained by minimizing the multinomial version of the Cross Entropy loss (Equation 1 of section 2.3.2) in a mini-batch Gradient Descent
setting. When training the Alternative 5’-splicing network, the loss is defined for mini-batch $\mathcal{B}^C \subset \mathcal{D}^C$ as:

$$L(\mathcal{B}^C) = -\sum_j \sum_i P_C^i(j) \star \ln(\hat{P}_i^C(j)), j \in \mathcal{B}^C, i \in \{0, 1, 2, 3\}$$

where $P_0^C(j) = 1 - P_1^C(j) - P_2^C(j) - P_3^C(j)$. In order to compare the CNN network to the Logistic regression model (section 3.2), a binary version of the model is used as well which only regresses $P_1^C(j)$. Binary Cross Entropy loss is then taken between $P_1^C(j)$ and $(1 - P_1^C(j))$ as was used to train the LR model. The binary Cross Entropy is also used for the Skipped exon model.

The actual Gradient Descent procedure involves computing the gradient of the loss at each layer with respect to the weights and updating them with a small step in the negative direction of the gradient. This means that for the output Logistic regression layer (Layer 7) with weight vector $\vec{w}_7$ and bias $b_7$ (or weight matrix and bias vector respectively in the multinomial case) we compute the gradients according to Equation 4-5 and perform updates according to Equation 6. For the hidden layer (Layer 6), the error signals $\vec{\delta}_6$ are computed according to Equation 10 and updated for the weight matrix $W_6$ and bias vector $\vec{b}_6$ according to Equation 11. Finally, for the convolutional layers (Layers 1, 2 and 4), the error signal maps $\delta_l,k_l$ are computed according to Equation 14. The kernel filters $W_l,k_l,l-1$ and the bias vector $\vec{b}_l,k_l$ are updated with Equation 15.

The CNN is implemented using the excellent Theano Math expression compiler[31]. Theano allows layers to be defined as basic object classes and the activation functions of the layers can easily be linked. Theano also automatically computes the gradients by compiling a computational graph of the linked activation functions and applying a recursive differentiation.

For the Alternative 5’-splicing library, training is done on 80% of the data. 10% of the data is used as a validation set for testing during development and for stopping early (discussed in section 2.3.2). The remaining 10% of the data is used for testing. For the skipped exon library, which is approximately 5 times larger than the 5’-splicing library, 90% of the data is used for training while the last 10% of the data is split equally between the validation and test set.

### 3.4 Biases, non-modeled interaction & noise

In this section the limitations and biases of the modeling inherent to the DNA libraries are summarized, starting with a brief discussion on target label noise. The target ratios of the synthetic DNA library members are, as mentioned in section 3.1.1, estimated by sequencing the RNA transcripts which were expressed from each member gene and computing the relative abundance of each isoform type. This means that the target ratios $\tilde{P}^C(j)$ are noisy estimates of the true library member ratios $P_C^\text{true}(j)$ and the quality of the estimation is directly dependent on the number of RNA isoform sequencing reads which $\tilde{P}^C(j)$ was estimated from. To illustrate this, consider the binary case:

Isoform $I_A$ is expressed and observed with probability $P_A$ and isoform $I_B$ is expressed with probability $1 - P_A$. If we assume $I_A$ is Bernoulli distributed, the unbiased maximum likelihood estimate of $P_A$ is $\hat{P}_A = \frac{N_A}{N_A + N_B}$, where $N_A$ is the number of times $I_A$ is observed and $N_B$ is the number of times $I_B$ is observed.
The standard deviation of the estimator \( \hat{P}_A \) is proportional to \( \frac{1}{\sqrt{n}} \)[32]. Consequently, the quality of the estimated isoform ratios improve on higher RNA read count. The Skipped exon library actually contains many more library members which have few RNA reads than the Alternative 5'-splicing library. These low-quality ratio estimations are however used when training the models, as it has been shown that classifiers may learn from data even when large amounts of noise is present[33]. To maximize testing accuracy, however, the data sets are sorted according to isoform read count prior to training and the highest-quality library members are chosen for the test set.

Finally, it is important to emphasize what is varied in the DNA library and what may be learned by the machine learning models. Remember that the randomized regions of both libraries lie on introns, without overlapping any of the target splice sites. This means that all variation in isoform ratio comes from the inclusion (or exclusion) of Intronic Sequence Enhancers or Silencers. It is widely known (as discussed in section 2.1.3) that Exonic effectors, including the regions directly surrounding the splice sites, are highly influential in the regulation of alternative splicing. This effect is neither varied nor modeled here, and the target ratios will simply be under the effect of a bias from the splice sites and Exonic effectors fixed in the non-randomized part of the mini-gene.

### 3.5 Model evaluation on synthetic data

To evaluate the performance of the developed models, test sets of the synthetic libraries are held out during training and later used to test prediction accuracy. As previously mentioned, 10% of the Alternative 5'-splicing data set is kept for testing (26,500 out of 265,000 sequences). For the Skipped exon library, approximately 5% is used for testing (60,000 out of 1,400,000 sequences).

The metrics used for evaluating prediction accuracy are \( R^2 \) score and Mean absolute error. \( R^2 \), or the Coefficient of determination, measures how much of the variance in the data set is explained by the model. It is defined as:

\[
R^2 = 1 - \frac{\sum_j (\hat{P}(j) - P(j))^2}{\sum_j (\bar{P}(j) - P(j))^2}
\]

where \( P(j) \) is the target ratio for test sequence j, \( \hat{P}(j) \) is the predicted ratio and \( \bar{P}(j) \) is the average ratio of the test set. Mean absolute error is defined as

\[
E_{MAE} = \frac{1}{N} * \sum_j |\hat{P}(j) - P(j)|
\]

which measures average predicted ratio error.

### 3.6 Model evaluation on non-synthetic data

In order to test the ability to generalize across genes, the LR model and the CNN are tested on a set of human Single-Nucleotide Polymorphism (SNP) DNA sequences, which are sequences where a point-mutation commonly occurs at some specific position. If these point-mutations occur on an intron, they may alter the Intronic Splicing Effector binding sites related to nearby alternative splicing events. The models are tasked with predicting the change in expressed Alternative 5'-splicing isoform ratio due to the point-mutations of the SNPs, having previously been trained on the synthetic Alt. 5'-splicing library.

Seelig Lab had prior to this report processed DNA- and RNA sequencing data from the 1000 Genomes Project[34] and the GEUVADIS consortium[35],
which resulted in a data set of approximately 400 heterozygous SNPs which have point-mutations on or between two alternative 5’-splice donors. The MISO software package was then used to estimate the percent usage of each donor[36]. For each SNP-pair (S\textsubscript{wild}, S\textsubscript{mut}), where S\textsubscript{wild} is the wildtype sequence and S\textsubscript{mut} is a point-mutated variant of S\textsubscript{wild}, the models predict:

\[
\Delta \hat{P} = \hat{P}_{\text{wild}} - \hat{P}_{\text{mut}}
\]

The predictions are evaluated in comparison to the target isoform ratio change, \(\Delta P = P_{\text{wild}} - P_{\text{mut}}\), in two ways. First, the percentage of correct direction in change is computed. A direction in change is considered correct if \(\text{sign}(\Delta \hat{P}) = \text{sign}(\Delta P)\). Second, \(R^2\) is used to evaluate how well the predicted change correlates with the target change. The LR model uses the counts of 6-mers between and after the two splice donors as input features. For the CNN, which takes two fixed-size 35bp regions as input, the first input window is centered around the point-mutation in the between-donors region, while the second input window is placed directly after the downstream splice donor.

### 3.7 Differentially spliced isoforms across cell lines

An interesting extension of modeling cell line-specific alternative splicing is to evaluate how well the models can predict differential expression (the difference in ratio) of splicing isoforms between cell lines. To this end, the following problem is considered: For cell lines (\(C_1, \ldots, C_K\)), define the set \(T = \{0,\ldots,0\},\ldots,(1,\ldots,1)\) of all \(K\)-length binary sequences. For each target logic \(\vec{t} \in T\), find sequences \(S\) which minimize the Manhattan distance between the predicted ratios and the target logic:

\[
d(S, \vec{t}) = \sum_{k=1}^{K} |\text{Predict}(S, \mathcal{M}^{C_k}) - t_k|
\]

where \(\text{Predict}(S, \mathcal{M}^{C_k})\) is a function which predicts the isoform ratio \(\hat{P}_{C_k}\) for sequence \(S\). The problem is approached by implementing a Greedy search strategy which, from a suitable candidate list of start sequences, generates neighbor sequences and enqueues them in a priority queue according to \(d(S, \vec{t})\). A neighbor sequence of \(S\) is generated by performing \(M\) base-pair mutations in the randomized regions of \(S\). To increase search speed and allow the procedure to sample over a large part of the state space, \(M\) is set to 10.

To generate start sequences for each target logic \(\vec{t}\), the DNA library is fed through the prediction models \(\mathcal{M}^{C_k}\) for each cell line \(C_k\) and the resulting prediction vectors (\(\hat{P}_{C_1}, \ldots, \hat{P}_{C_k}\)) are sorted according to \(d(S, \vec{t})\). The top 100 sequences with lowest \(d\) are chosen as start candidates. The problem is investigated for the Alternative 5’-splicing library. Only the binary case of finding differential expression of splice donor 1 (\(\hat{P}_{C_1}\) versus 1 – \(\hat{P}_{C_1}\)) is considered.

During initial testing, it was concluded that CNNs which are separately trained on cell line-specific data do not perform well at predicting differential expression, most likely due to the architecture allowing a too high degree of freedom. Instead, a joint CNN model is used for this task, which is trained by minimizing the sum of loglosses across all cell lines in order to simultaneously predict a vector \(\hat{P} = (\hat{P}_{\text{HEK}}, \hat{P}_{\text{HELA}}, \hat{P}_{\text{MCF7}}, \hat{P}_{\text{CHO}})\) of SD\(_1\) usage ratios for an input DNA sequence.
Evaluation of differential prediction

Differential splicing prediction is evaluated on the Alt. 5’-splicing test set by measuring the correlation between the predicted difference in SD usage ratio, \( \Delta \hat{P} = \hat{P}_{C_i} - \hat{P}_{C_j} \), and the target ratio difference, \( \Delta P_T = P_{C_i} - P_{C_j} \), between each pair of cell lines \( (C_i, C_j) \).

4 Results

This section presents the results of the model evaluation. First, the Alternative 5’-splicing models are tested on both synthetic and non-synthetic data. Next, a similar analysis is conducted on the Skipped exon models. Finally, the Alt. 5’-splicing CNN is evaluated and compared to Logistic regression when considering the task of modeling differential splicing across cell lines.

4.1 Alternative 5’-splicing analysis

First, the Multinomial Alternative 5’-splicing CNN (which jointly predicts SD1-, SD2- and SD3 splicing) is evaluated when applied to the synthetic 5’-splicing library. Figure 15 shows scatter plots of the predicted splice donor usage ratio versus target ratio on the HEK library test set. The correlation is strong for predictions on all splice donors; \( R^2 = 0.80 \) for predicting SD1 ratio, \( R^2 = 0.80 \) for SD2 and \( R^2 = 0.63 \) for SD3 predictions. Table 3 lists the Mean absolute error \( (E_{MAE}) \) and \( R^2 \) when using the CNN model to predict each splice donor usage ratio for all cell lines.

Table 3: Error statistics \( (R^2 \text{ and } E_{MAE}) \) of the splice donor usage ratio predictions when using the Multinomial CNN. The error statistics are shown for each cell line instance of the Alternative 5’-splicing library test set.

<table>
<thead>
<tr>
<th></th>
<th>CNN test ( R^2 )</th>
<th>CNN test ( E_{MAE} )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SD1</td>
<td>SD2</td>
</tr>
<tr>
<td>HEK</td>
<td>0.80</td>
<td>0.80</td>
</tr>
<tr>
<td>HELA</td>
<td>0.77</td>
<td>0.78</td>
</tr>
<tr>
<td>MCF7</td>
<td>0.78</td>
<td>0.79</td>
</tr>
<tr>
<td>CHO</td>
<td>0.79</td>
<td>0.79</td>
</tr>
</tbody>
</table>

Figure 15: Predicted SDi usage \( \hat{P}_i \) versus target usage \( P^T_i \). Predicted on the HEK instance of the Alt. 5’-splicing library test set using the CNN.
Next, we consider the binary task of predicting $SD_1$ usage ratio, $\hat{P}_1$, versus the ratio of splicing at any other position, $1 - \hat{P}_1$. For this analysis, the Logistic regression model and CNN are evaluated in parallel, starting on the training set of the Alt. 5’-splicing library. Figure 16 shows the correlation between predicted $SD_1$ usage ratio versus target ratio on the HEK training set. The left graph plots the scatter for the LR model while the right one plots the scatter for the CNN. The LR model exhibits significantly lower correlation ($R^2 = 0.63$) compared to the CNN ($R^2 = 0.81$). This trend can be observed for all cell lines, which is shown in Table 4. On average, the CNN has 0.17 higher $R^2$ than the LR model and less than half of the LR model’s $E_{MAE}$.

<table>
<thead>
<tr>
<th></th>
<th>$SD_1$ train $R^2$</th>
<th>$SD_1$ train $E_{MAE}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LR</td>
<td>CNN</td>
</tr>
<tr>
<td>HEK</td>
<td>0.63</td>
<td>0.81</td>
</tr>
<tr>
<td>HELA</td>
<td>0.62</td>
<td>0.78</td>
</tr>
<tr>
<td>MCF7</td>
<td>0.61</td>
<td>0.79</td>
</tr>
<tr>
<td>CHO</td>
<td>0.64</td>
<td>0.80</td>
</tr>
</tbody>
</table>

Table 4: Error statistics ($R^2$ and $E_{MAE}$) of the $SD_1$ ratio predictions on the training set of the Alternative 5'-splicing library for the CNN and LR model.

An identical evaluation is performed on the Alt. 5'-splicing test sets. Figure 17 shows the scatter plots on the HEK test set for the LR model (left plot) and CNN (right plot) when predicting $SD_1$ usage ratio. Table 5 shows the $R^2$ and $E_{MAE}$ for each model and cell line. The $R^2$ is on average 0.15 higher for the CNN compared to the LR, and the $E_{MAE}$ is on average 45.3% smaller. Scatter plots for the LR model and the CNN on the remaining cell line test sets can be found in Appendix A.

Figure 16: Predicted splice donor usage $\hat{P}_1$ versus target usage $P_T^1 (SD_1)$ for splice donor $SD_1$ on the HEK training set of the Alt. 5'-splicing library. Scatter plots are shown for (a) the LR model and (b) the CNN model.
(a) LR: \( \hat{P}_1 \) vs. \( P^T_1 (SD_1) \), \( R^2 = 0.63 \).

(b) CNN: \( \hat{P}_1 \) vs. \( P^T_1 (SD_1) \), \( R^2 = 0.79 \).

Figure 17: Predicted splice donor usage \( \hat{P}_1 \) versus target usage \( P^T_1 \) for splice donor \( SD_1 \) on the HEK test set of the Alt. 5’-splicing library. Scatter plots are shown for (a) the LR model and (b) the CNN model.

Table 5: Error statistics (\( R^2 \) and \( E_{MAE} \)) of the \( SD_1 \) ratio predictions on the test set of the Alternative 5’-splicing library for the CNN and LR model.

<table>
<thead>
<tr>
<th></th>
<th>( SD_1 ) test ( R^2 )</th>
<th>( SD_1 ) test ( E_{MAE} )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LR</td>
<td>CNN</td>
</tr>
<tr>
<td>HEK</td>
<td>0.63</td>
<td>0.79</td>
</tr>
<tr>
<td>HELA</td>
<td>0.62</td>
<td>0.76</td>
</tr>
<tr>
<td>MCF7</td>
<td>0.61</td>
<td>0.77</td>
</tr>
<tr>
<td>CHO</td>
<td>0.64</td>
<td>0.78</td>
</tr>
</tbody>
</table>

Next, the evaluation tests how well the binary CNN model (which regresses \( SD_1 \) usage) generalizes across cell lines of the synthetic library. To this end, a separate CNN model is trained on each cell line’s training set and then cross-tested against each cell line’s test set. The \( R^2 \) for each model evaluated on each cell line test set is shown in Table 6 below. The average \( R^2 \) for out-of-domain evaluations (where the model is tested on a cell line it was not trained for) is 0.73, which is close to the average within-domain performance (\( R^2 = 0.77 \)).

Table 6: \( R^2 \) values for predicting \( SD_1 \) usage ratio across all cell line test sets for CNNs trained separately on each of the cell lines’ Alt. 5’-splicing training sets. Rows denote test sets and columns denote model instances.

<table>
<thead>
<tr>
<th></th>
<th>CNN ( SD_1 ) cell line cross-testing, ( R^2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>HEK</td>
</tr>
<tr>
<td>HEK data</td>
<td>0.79</td>
</tr>
<tr>
<td>HELA data</td>
<td>0.78</td>
</tr>
<tr>
<td>MCF7 data</td>
<td>0.76</td>
</tr>
<tr>
<td>CHO data</td>
<td>0.74</td>
</tr>
</tbody>
</table>

Finally, the models are applied to non-synthetic data, with a test strategy as described in section 3.6. The models are trained on the synthetic HELA library
and then tasked with predicting the change in splice donor usage ratio, $\Delta \hat{P}$, between heterozygous SNP isoforms of human genes gathered from the 1000 Genomes project. The evaluation considers three different levels of restriction on what SNP genes are tested. The first case considers the entire data set of SNPs where mutations can occur anywhere in the intron between two splice donors (132 genes). Figure 18 shows the scatter plot of the predicted isoform ratio change versus target change for the CNN. The table lists the $R^2$, $E_{MAE}$ and $\text{Dir} \ %$ for both the CNN and the LR model. The $\text{Dir} \ %$ metric computes the percentage of correctly predicted direction of change (explained in section 3.6). The LR model has a higher correlation with the target change ($R^2 = 0.17$) than the CNN ($R^2 = -0.21$).

![Figure 18: (a) Scatter plot of predicted isoform ratio change $\Delta \hat{P}$ due to SNP mutations versus target change (for the CNN) and (b) error statistics for isoform ratio change predictions when using the LR model and the CNN.](image)

In the next case, SNPs are restricted to only occur within $+5$ to $+40$ bps downstream of the first splice donor, resulting in 56 remaining SNP genes. This is equivalent to where the first randomized region in the synthetic Alt. 5’-splicing library was positioned. Figure 19 shows the scatter plot of the predicted isoform ratio change versus target change for the CNN and the table compares the performance of the CNN to the LR model. The correlation is still higher for the LR model ($R^2 = 0.48$) compared to the CNN ($R^2 = 0.25$).

![Figure 19: (a) Scatter plot of predicted isoform ratio change $\Delta \hat{P}$ due to (restricted) SNP mutations versus target change (for the CNN) and (b) error statistics for isoform ratio change predictions when using the LR model and the CNN.](image)
For the final level of restriction, only SNPs where the target change in isoform ratio is larger than 0.04 is considered (along with the restriction from the previous case), resulting in a total of 32 SNP events. Figure 20 shows the scatter plot of predicted isoform ratio change versus target change for the CNN and the table compares the performance of the CNN to the LR model.

<table>
<thead>
<tr>
<th></th>
<th>CNN</th>
<th>LR</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E_{MAE}$</td>
<td>0.043</td>
<td>0.043</td>
</tr>
<tr>
<td>Dir %</td>
<td>94%</td>
<td>90%</td>
</tr>
<tr>
<td>$R^2$</td>
<td>0.58</td>
<td>0.53</td>
</tr>
</tbody>
</table>

(a) CNN: $\hat{P} \text{ vs. } \hat{P}_T$, $R^2 = 0.58$

Figure 20: (a) Scatter plot of predicted isoform ratio change $\Delta \hat{P}$ due to (restricted) SNP mutations versus target change (for the CNN) and (b) error statistics for isoform ratio change predictions when using the LR model and the CNN.

### 4.2 Alternatively skipped exon analysis

The evaluation results of the Alternatively skipped exon models are presented below. Similar to the evaluation procedure carried out for Alternative 5'-splicing, the models are tested on a withheld test set from the synthetic DNA library. Figure 21 shows the scatter plots of predicting the Exon inclusion ratio on the HEK test set using each of the two models. The correlation between predicted and target isoform ratio is moderate for both the LR model and CNN ($R^2 = 0.34$ and $R^2 = 0.42$ respectively).

![Scatter plot of predicted exon inclusion ratio vs. target ratio on the HEK test set of the Alt. skipped exon library.](image)

(a) LR: $\hat{P}$ vs. $P_T$, $R^2 = 0.34$

(b) CNN: $\hat{P}$ vs. $P_T$, $R^2 = 0.42$

Figure 21: Predicted exon inclusion ratio versus target ratio on the HEK test set of the Alt. skipped exon library. Scatter plots are shown for (a) the LR model and (b) the CNN model.

The same evaluation is carried out on the MCF7 cell line. Figure 22 shows the
scatter plots of ratio prediction for both of the models. Here, the correlation is even weaker ($R^2 = 0.02$ for the LR model and $R^2 = 0.07$ for the CNN) compared to the evaluation on HEK.

![LR - Skipped exon test set (MCF7).](image1)

(a) LR: $\hat{P}$ vs. $P_T$, $R^2 = 0.02$

![CNN - Skipped exon test set (MCF7).](image2)

(b) CNN: $\hat{P}$ vs. $P_T$, $R^2 = 0.07$

Figure 22: Predicted exon inclusion ratio versus target ratio on the MCF7 test set of the Alt. skipped exon library. Scatter plots are shown for (a) the LR model and (b) the CNN model.

Finally, a separate CNN model is trained on each cell line’s Skipped exon library instance and cross-tested on each cell line’s test set in order to evaluate generalization across cell types. Table 7 below shows the $R^2$ for each cell line-specific model and test set. The correlation is weak for the out-of-domain tests, with some of the $R^2$ coefficients being negative.

| CNN cell line cross-testing, $R^2$ |
|-------------------------------|-------------|-------------|-------------|
| Model | HEK | HELA | MCF7 |
| HEK data | 0.42 | 0.17 | 0.17 |
| HELA data | 0.09 | 0.39 | 0.04 |
| MCF7 data | -0.13 | -0.78 | 0.07 |

Table 7: $R^2$ values for predicting exon inclusion ratio across all cell line test sets for CNNs trained separately on each of the cell line training sets of the Skipped exon library. Rows denote test sets and columns denote model instances.

### 4.3 Differential splicing prediction

The final part of the evaluation considers the problem of predicting gene variants from the Alternative 5’-splicing library which are differentially spliced (i.e. have different splicing isoform ratios) between the cell lines HEK, HELA, MCF7 and CHO. The evaluation compares the LR model and the joint CNN (as described in section 3.7) on the basis of how well their predicted difference in splice donor usage ratio between cell lines correlate with the target difference. The evaluation is carried out on the test set by predicting $SD_1$ usage between pairs of cell lines.

Figure 23 plots for each of the two models the correlation between predicted difference in ratio of cell lines MCF7 and CHO, $\Delta \hat{P} = \hat{P}_{MCF7} - \hat{P}_{CHO}$, versus target difference in ratio, $\Delta P_T = P_{MCF7}^{T} - P_{CHO}^{T}$ as a colored scatter. The x-axis
plots the target MCF7 ratio, the y-axis plots target CHO ratio and the color chart plots \( \text{BLUE} = -1.0 \leq \Delta \hat{P} \leq 1.0 = \text{RED} \) (ideally, the upper left region should be blue and the lower right should be red). The correlation is similar for both models, with \( R^2 = 0.26 \) for the LR model and \( R^2 = 0.33 \) for the CNN. Figure 24 displays the colored scatter plot for the pair (HELA, MCF7), where \( R^2 = 0.16 \) for LR and \( R^2 = 0.23 \) for the CNN. The same analysis is carried out for all other pairs of cell lines and Table 8 summarizes the evaluations by listing the \( R^2 \) coefficients for both models. As can be seen, the correlation between predicted difference in \( SD_1 \) usage ratio and target ratio is moderate, with a slightly higher \( R^2 \) for the CNN compared to the LR model (0.04 higher \( R^2 \) on average). Scatter plots for the remaining pairs of cell lines can be found in Appendix B. The appendix also lists a selection of the most differentially spliced genes found by the Greedy search procedure developed in section 3.7.

![Colored scatter plot](image)

(a) LR: \( \Delta \hat{P} \) vs. \( \Delta P_T \), \( R^2 = 0.26 \)

(b) CNN: \( \Delta \hat{P} \) vs. \( \Delta P_T \), \( R^2 = 0.33 \)

Figure 23: Colored scatter plot of \( \Delta \hat{P} = \hat{P}_{\text{MCF7}} - \hat{P}_{\text{CHO}} \) versus \( \Delta P_T = P^T_{\text{MCF7}} - P^T_{\text{CHO}} \), on the Alt. 5’-splicing test set. The x-axis plots \( P^T_{\text{MCF7}} \), the y-axis plots \( P^T_{\text{CHO}} \) and the color chart plots \( \Delta \hat{P} \).

![Colored scatter plot](image)

(a) LR: \( \Delta \hat{P} \) vs. \( \Delta P_T \), \( R^2 = 0.16 \)

(b) CNN: \( \Delta \hat{P} \) vs. \( \Delta P_T \), \( R^2 = 0.23 \)

Figure 24: Colored scatter plot of \( \Delta \hat{P} = \hat{P}_{\text{HELA}} - \hat{P}_{\text{MCF7}} \) versus \( \Delta P_T = P^T_{\text{HELA}} - P^T_{\text{MCF7}} \), on the Alt. 5’-splicing test set. The x-axis plots \( P^T_{\text{HELA}} \), the y-axis plots \( P^T_{\text{MCF7}} \) and the color chart plots \( \Delta \hat{P} \).
### Analysis

Here, the evaluation results from the previous section are discussed and analyzed. This section will try to present reasonable explanations or causes for the obtained results as well as discuss the immediate consequences of them, beginning with the Alternative 5'-splicing analysis. Table 4 and 5 compared the \( R^2 \) and Mean absolute error of the LR model to the CNN when predicting the usage ratio of splice donor 1. With an average increase in \( R^2 \) of 0.15 and almost half the average prediction error on the test set, the CNN is clearly outperforming the LR model. The increased accuracy can easily be seen in the scatter plots of Figure 16 and 17, where the predicted versus target ratio correlate into an almost straight line for the CNN. When comparing the error statistics between training- and test set, neither model appears to be overfitting. The CNN only has a slightly higher prediction error on the test set, while the LR model has an almost identical error between training and test data. If the LR is only able to predict as good on new data as it could on training data, it must be that the model lacks in capacity to learn as much about the splicing library as the CNN.

Of course, the difference in performance may not be surprising, considering the internals of the networks; the LR model linearly combines the weight of each 6-nt sequence before doing the logistic regression. The CNN on the other hand has 3 convolutional layers and a fully connected hidden layer where sequence information may be combined in a large number of ways before doing the actual regression. We know from section 2.1.3 that Intronic Enhancers and Silencers do cooperate to a large extent in order to regulate alternative splicing, which is something only the CNN has the ability to capture. As an example, CELF-family binding proteins ETR3 require the binding of CUGBP1 in order to coregulate the splicing of TNTT2 exon 5[2]. The linear LR model has no way of conditioning the effect of ETR3 proteins on the presence or absence of CUGBP1 binding sites, while the CNN can represent such dependencies within its hidden layers. The observant reader may then argue that one simply has to use a suitable polynomial basis function to generate combinatorial features of the 6-mer motifs, use these non-linear features as input to the LR model, and it will be able to perform as well as the CNN. Perhaps, but consider what happens to the dimensionality: Using a polynomial kernel of degree 2 together with counts of 6-mer motifs (with 4096 unique 6-mer motifs), the dimensionality \( D \) becomes \( D = \frac{4096 \cdot (4096 - 1)}{2} + 4096 = 8,390,656 \). The dimensionality becomes much larger than the number of available training examples (265,000 for the Alt.

<table>
<thead>
<tr>
<th>Cell line pairs</th>
<th>( R^2 ) LR</th>
<th>( R^2 ) CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>HEK vs. HELA</td>
<td>0.04</td>
<td>0.05</td>
</tr>
<tr>
<td>HEK vs. MCF7</td>
<td>0.12</td>
<td>0.17</td>
</tr>
<tr>
<td>HEK vs. CHO</td>
<td>0.15</td>
<td>0.16</td>
</tr>
<tr>
<td>HELA vs. MCF7</td>
<td>0.16</td>
<td>0.23</td>
</tr>
<tr>
<td>HELA vs. CHO</td>
<td>0.05</td>
<td>0.05</td>
</tr>
<tr>
<td>MCF7 vs. CHO</td>
<td>0.26</td>
<td>0.33</td>
</tr>
</tbody>
</table>

Table 8: \( R^2 \) coefficients for LR and the joint CNN when predicting difference in \( SD_1 \) usage ratio between pairs of cell lines.
5'-splicing library) and overfitting will become a major concern. Furthermore, the model will likely only see a fraction of all possibly 6-mer motif combinations, meaning a large portion of the 8,390,656 weights will not be learned. The CNN overcomes this problem entirely by modeling non-linearity via multiple layers instead of a large number of features. According to the calculations made in section 3.3.2, the current CNN architecture has 51,333 adjustable weights, which is a lot less than 8,390,656. As explained in section 2.3.3, this is the major benefit of deep learning; a deep network can potentially model the same relationships as a shallow network using only a fraction of activational units.

Moving on to the task of predicting change in isoform ratio due to SNP mutations in non-synthetic human genes, the caveats of the CNN model become apparent. By inspecting the error statistics in Figure 18 - 20, it is evident that the LR model performs better at predicting SNP change than the CNN. In Figure 20, the CNN model has marginally better $R^2$ ($R^2 = 0.58$ versus $R^2 = 0.53$), but for the earlier parts of the evaluation where SNP mutations can occur anywhere on the intron, the LR model clearly is superior. Specifically, the performance of the CNN deteriorates when predicting SNPs where the mutation occurs outside of a +5 to +40bp window downstream of the first splice donor. Not surprisingly, this is the region where the CNN was trained in the synthetic library (the first randomized region is located +5bp downstream of $SD_1$). While the LR model has been trained on the exact same library, it is able to generalize quite well across the entire intron. This is most likely due to the low-variance design of the LR. Or oppositely, the CNN’s capacity to learn very complex relationships in the synthetic library allows it to predict accurately within that domain, but this specificity does not carry to other aspects of a gene. In particular, the results in Figure 18 - 20 reveal a large positioning bias for the CNN. However, the CNN does generalize in other aspects. For one, the CNN predicts Alternative 5'-splicing across cell lines very well, as can be seen in Table 6 where CNNs are separately trained on each cell line and then used to predict $SD_1$ usage for test sets of other cell lines. On out-of-domain tests, the $R^2$ is on average only 0.04 lower compared to within-domain tests.

To summarize the analysis up to this point, the deep learning properties of the Convolutional Neural Network comes with both advantages and disadvantages compared to the Logistic regression model. It is capable of learning and predicting alternative splicing on the synthetic DNA library to a higher degree than the LR model. However, Logistic regression generalizes much better to other genes and, in particular, to other intronic regions.

Let’s briefly discuss the Alternatively Skipped exon models. Figure 21 - 22 display a rather weak correlation between predicted exon inclusion ratio and target ratio. This is the case for both the CNN and LR model, with an $R^2$ of 0.42 and 0.34 on the HEK library respectively. For the MCF7 library, results are even worse ($R^2 = 0.07$ for the CNN and $R^2 = 0.02$ for Logistic regression). The most probable explanation for the poor performance is the low RNA sequencing read count per library member (2.0 on average per member and cell line) compared to the Alt. 5'-splicing library (54.4 on average per member and cell line), which as explained in section 3.4 directly translates to target ratio noise. A supporting argument is the fact that the MCF7 library, which the models perform worst on, has lowest read count of all of the libraries, while they perform best on the HEK library which has the highest read count. In fact, the MCF7 library only has reads for approximately 300,000 of the library members (effectively having only
300,000 members) while the HEK library has reads for approximately 900,000 members. A more interesting question is if the models actually learn about exon skipping or not. Because the test sets also have relatively low read counts, one cannot be sure if the models perform poorly due to an inability to learn from the training data, or due to testing against noisy target ratios. Further analysis of the library is required to draw any conclusions.

The final part of the evaluation conducted in section 4 regarded the modeling of differentially spliced genes of the Alt. 5’-splicing library across cell lines. Table 8 shows that both the LR model and CNN are capable of modeling difference in splice donor usage ratio between cell lines. The seemingly low $R^2$’s should be taken together with the fact that many of the sequences with target differences in ratio close to 1.0 are most likely due to noise (as a result of low RNA read counts). This is supported in the data by computing the average RNA read count for members whose $SD_1$ usage ratio differs between cell lines by more than 0.9, which for the Alt. 5’-splicing library is 4.3. This is significantly lower than the overall average read count of 54.4. Alternative splicing is an extremely conserved process across eukaryotic cells and the differential expression of spliced isoforms due to cell-specific properties is very moderate[2]. Considering that the models are not explicitly trained to regress the difference in usage ratio, it is quite remarkable that the average $R^2$ is as high as the results indicate (0.17 for the CNN). The reason why the CNN has stronger correlation than the LR model (0.04 higher $R^2$ on average) is probably due to its highly increased prediction accuracy on each individual cell line ($R^2 = 0.79$ on average) as previously discussed. Furthermore, the correlation observed on the test set gives confidence to the differential predictions found by the Greedy first search (of which a selection is shown in Table 11 of Appendix B) and, while the top search results are somewhat moderate, some sequences indicate a predicted difference in ratio of as much as 0.5. One would expect the predictions found by the search procedure to be at least as correlated with the true splicing isoform ratios as indicated on the test set. Most likely, though, the correlation could be even stronger; the CNN has been trained on hundreds of thousands of examples and it seems reasonable it would be able to predict at a higher accuracy than estimations made off a relatively low number of RNA reads.

Finally, let’s compare the interpretability of the CNN and the LR model. Can the splicing-regulatory elements be identified and understood from the models? For the Logistic regression, the 8192 weights correspond to effect sizes of specific 6-mer sequence motifs in either the upstream or downstream randomized intronic region. Negative weights are Intronic Silencers, while positive weights are Intronic Enhancers. Sorting the weights on magnitude will rank the effectors on their influence. Table 9 lists a selection of the 20 largest enhancers found by the HEK-based Alt. 5’ LR model. The G-rich sequences are hnRNPH binding sites and GAA-sequences are known to recruit SR proteins such as TRA2β[2]. Similarly, Table 10 lists the most influential silencers found by the model. Most notably, many of the silencers are in fact new splice sites which are inserted into the randomized regions, effectively repressing $SD_1$ usage. Other silencer elements recruit SR proteins such as TRA2β or ASF/SF2[2]. These RBPs are normally regarded as enhancers, but most likely these binding sites enhance the usage of $SD_2$ or $SD_3$ and consequently silences $SD_1$ usage.
Selection of the largest positive LR weights

<table>
<thead>
<tr>
<th>Region</th>
<th>6-mer</th>
<th>Weight</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GGGGGG</td>
<td>0.88</td>
<td>hnRNP H</td>
</tr>
<tr>
<td>1</td>
<td>AGGGGG</td>
<td>0.49</td>
<td>hnRNP H</td>
</tr>
<tr>
<td>1</td>
<td>GGGGGA</td>
<td>0.46</td>
<td>hnRNP H</td>
</tr>
<tr>
<td>1</td>
<td>TGGGGG</td>
<td>0.45</td>
<td>hnRNP H</td>
</tr>
<tr>
<td>1</td>
<td>GGGGAA</td>
<td>0.34</td>
<td>hnRNP H or TRA2β</td>
</tr>
<tr>
<td>1</td>
<td>GAAGGG</td>
<td>0.29</td>
<td>TRA2β or Srp40</td>
</tr>
</tbody>
</table>

Table 9: Selection of the top 20 Intronic Enhancers with largest magnitude as found by the LR model. Their effects are mapped by cross-referencing [2].

Selection of the largest negative LR weights

<table>
<thead>
<tr>
<th>Region</th>
<th>6-mer</th>
<th>Weight</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AGGTAA</td>
<td>-0.74</td>
<td>Splice site</td>
</tr>
<tr>
<td>1</td>
<td>GTAAGT</td>
<td>-0.39</td>
<td>Splice site</td>
</tr>
<tr>
<td>1</td>
<td>GAAGAA</td>
<td>-0.28</td>
<td>TRA2β or ASF/SF2</td>
</tr>
<tr>
<td>1</td>
<td>GAAGAC</td>
<td>-0.22</td>
<td>TRA2β or 9G8</td>
</tr>
<tr>
<td>2</td>
<td>AGGTAA</td>
<td>-0.18</td>
<td>Splice site</td>
</tr>
<tr>
<td>1</td>
<td>ACGAAG</td>
<td>-0.16</td>
<td>TRA2β</td>
</tr>
</tbody>
</table>

Table 10: Selection of the top 20 Intronic Silencers with largest magnitude as found by the LR model. Their effects are mapped by cross-referencing [2].

The same type of model interpretation is not possible for the CNN. While the convolutional kernels in the first layer do respond to certain 6-mer motifs they have been trained to be sensitive towards, the convolutions are immediately fed through a ReLU ($g(z) = \max(z, 0)$), meaning only positive convolutions are kept as non-zero. Consequently, important regulatory elements are all transformed into positive values so that the next layer can respond to them, regardless of whether the elements are silencers or enhancers. Furthermore, the sign of these convolutions may in successive layers be flipped numerous times. To see what 6-mer motifs the first convolutional layer is sensitive towards, it is fed with every possible 6-mer motif as input and the top 50 convolutions of largest positive magnitude are kept for each kernel. Then, the top list of each kernel is cross-referenced with the top list of the LR weights. Doing this, we find 44 of the 50 largest LR enhancers for at least one of the kernel top lists. We also find 32 of the 50 largest LR silencers in at least one of the kernel top lists. Going beyond this interpretation is hard, as the non-linear interaction between sequence information is deeply encoded within the layers of the CNN.

To conclude the analysis, a few words on the future of Convolutional Neural Nets in sequence analysis and how the model developed here may be improved or built upon. What is remarkable with this type of model is its ability to predict splicing very accurately solely on the basis of the DNA sequence information. There are no hand-crafted features, e.g. RBP binding scores or exon length, as there are in the state of the art Neural nets described in section 2.2.2. Instead, the model infers splicing-regulatory effects simply by finding statisti-
cal correlations in the intronic sequences. But to further improve differential modeling between cell lines for the current CNN build, incorporating features such as Transcription Factor abundance levels may be required. These metrics could easily be connected as ad-hoc inputs to the fully-connected layers in order to differentiate between specific cellular contexts. However, obtaining data sets of these features will involve a much more complex process compared to how the libraries used in this paper were generated. Retrieving massive sets of RNA-Seq data is relatively easy with today’s Next-gen sequencing techniques and the very large amount of training data is probably the fundamental reason for the high performance. Another challenge with the current CNN model is overcoming the positioning bias of where in the intron the CNN was trained (as previously discussed). Mixing in completely location-invariant features such as 6-mer counts could possibly alleviate this issue. Sequence convolutions could be restricted to small regions overlapping the splice sites while summative features such as motif counts could be used for the bulk of the intronic and exonic regions. The architecture and hence the learning capabilities of the CNN is primarily only restricted by the structure and variability of the DNA library which it is trained on, meaning the CNN could potentially model other aspects of splicing, or completely different transcriptional events, if provided the data.

6 Conclusion

In this report, a deep learning methodology based on Convolutional Neural Networks is applied to the problem of predicting alternative splicing RNA isoform ratio using only the original DNA sequence as input. The results of the analysis showed that CNNs have the capacity to learn more of the complex regulatory interactions involved in alternative splicing compared to linear Logistic regression. For the synthetic Alternative 5' splicing library analyzed in this paper, the CNN was able to achieve on average a 0.15 higher $R^2$ and almost half as low Mean absolute error than the LR model when predicting splice donor usage ratio on the test set. The CNN was shown to predict isoform ratio very accurately with a mean test set $R^2$ of 0.77 and, by training a joint model over multiple cell lines, differentially expressed splicing events could be modeled efficiently. The analysis further showed that the CNN generalized predictions very well across cell lines, with a mean $R^2$ of 0.73 when predicting isoform ratio on cell line-specific libraries different from its training library. For the synthetic Skipped exon library, both LR and the CNN was only able to achieve moderate correlation with the target isoform ratios, with a CNN $R^2$ of 0.42 on the HEK library. The reason for the poor performance is probably due to target ratio noise as a result of low RNA read count per library member.

While prediction accuracy is notably raised when predicting Alternative 5' splicing on sequences from the synthetic data set using the CNN, the performance quickly deteriorates when predicting splicing behavior on non-synthetic genes where the intronic regions used as input are displaced from where the CNN was trained on. The accuracy of the Logistic regression model does not deteriorate to the same extent and it is concluded that the high-bias, low-variance characteristics of the regularized LR model allows for better generalization across sequence position compared to the CNN. Additionally, Logistic regression is far easier to interpret in terms of identifying sequence-regulatory elements.
In conclusion, there is no single best choice of model for predicting alternative splicing. Both Logistic regression and Convolutional Neural Nets have their advantages and disadvantages and the model choice depends on the application. CNNs allow for highly accurate modeling within specific contexts which could be advantageous in a Synthetic biology setting, for example when designing differentially spliced gene constructs as shown in this paper. However, the CNN does come with a bias towards the contextual environment it was trained within, be it gene type or other structural relationships, which does not carry well into new contexts. The LR model is more suitable for learning about splicing events in general and for identifying regulatory elements, a task which requires more contextual generalization and easier interpretability than the CNN provides.

7 Acknowledgments

The paper is based on previous work conducted by Seelig Lab at the Dept. of Electrical Engineering and Computer Science & Engineering at the University of Washington. In particular, the Logistic regression model was originally developed by Alexander Rosenberg at Seelig Lab. Rosenberg also created both of the synthetic DNA libraries used in this report. The author wishes to thank Prof. Georg Seelig for supervising the thesis work and to thank Alexander Rosenberg, Nick Bogard and Randolph Lopez for collaborating on the project.
Bibliography


[15] Xiong, Hui Yuan, Joseph Barash, and Brendan J. Frey. "Bayesian prediction of tissue-regulated splicing using RNA sequence and cellular context." Bioinformatics


Appendices

Appendix A - Alternative 5’-analysis supplemental material

This appendix lists the scatter plots of predicted $SD_1$ usage ratio versus target ratio on the Alt. 5’-splicing test set for the cell lines which were not presented in section 4. Figures 25-27 show the scatter plots for the LR model and CNN model respectively.

Figure 25: Predicted splice donor usage $\hat{P}_1$ versus target usage $P_T^1 (SD_1)$ for splice donor $SD_1$ on the HELA test set of the Alt. 5’-splicing library. Scatter plots are shown for (a) the LR model and (b) the CNN model.

Figure 26: Predicted splice donor usage $\hat{P}_1$ versus target usage $P_T^1 (SD_1)$ for splice donor $SD_1$ on the MCF7 test set of the Alt. 5’-splicing library. Scatter plots are shown for (a) the LR model and (b) the CNN model.
Figure 27: Predicted splice donor usage $\hat{P}_1$ versus target usage $P^T_1$ for splice donor $SD_1$ on the CHO test set of the Alt. 5'-splicing library. Scatter plots are shown for (a) the LR model and (b) the CNN model.
Appendix B - Differential splicing supplemental material

This appendix lists the colored scatter plots of predicted difference in $SD_1$ usage ratio versus target ratio difference on the Alt. 5'-splicing test set for the cell line pairs which were not presented in section 4. Figures 28-31 show the scatter plots for the LR model and CNN model respectively.

Figure 28: Colored scatter plot of $\Delta \hat{P} = \hat{P}_{\text{HEK}} - \hat{P}_{\text{HELA}}$ versus $\Delta P_T = P_T^{\text{HEK}} - P_T^{\text{HELA}}$, on the Alt. 5'-splicing test set. The x-axis plots $P_T^{\text{HEK}}$, the y-axis plots $P_T^{\text{HELA}}$ and the color chart plots $\Delta \hat{P}$.

Figure 29: Colored scatter plot of $\Delta \hat{P} = \hat{P}_{\text{HEK}} - \hat{P}_{MCF7}$ versus $\Delta P_T = P_T^{\text{HEK}} - P_T^{MCF7}$, on the Alt. 5'-splicing test set. The x-axis plots $P_T^{\text{HEK}}$, the y-axis plots $P_T^{MCF7}$ and the color chart plots $\Delta \hat{P}$.
Regarding the Greedy best first search method developed in section 3.7 for finding gene variants which are maximally close to specific target differential expressions $\mathbf{t} = (T_{C_1},...,T_{C_K}) \in \{(0,...,0),..., (1,...,1)\}$, Table 11 below lists a selection of the most differentially spliced gene variants found for various cell line combinations and target logic expressions. The table shows for each search result the 5’- and 3’- randomized regions together with the predicted ratios.
### Differential search results

<table>
<thead>
<tr>
<th>5' region</th>
<th>3' region</th>
</tr>
</thead>
<tbody>
<tr>
<td>(HEK, CHO) = (0, 1), $P = (0.33, 0.69)$</td>
<td>acggccggacgaagaacaaggtttc</td>
</tr>
<tr>
<td>(HEK, MCF7) = (0, 1), $P = (0.38, 0.70)$</td>
<td>ggggggggggggggggggtgggg</td>
</tr>
<tr>
<td>(HEK, MCF7) = (1, 0), $P = (0.64, 0.28)$</td>
<td>caggtaggtaggttagttccattcc</td>
</tr>
<tr>
<td>(HELA, MCF7, CHO) = (1, 0, 1), $P = (0.77, 0.45, 0.89)$</td>
<td>ggggagctggagtggagtgggt</td>
</tr>
<tr>
<td>(HEK, MCF7, CHO) = (1, 0, 1), $P = (0.77, 0.53, 0.94)$</td>
<td>gaggatggaatgggagatagctg</td>
</tr>
<tr>
<td>(HEK, MCF7, CHO) = (0, 0, 1), $P = (0.20, 0.10, 0.49)$</td>
<td>gaggagatcggggaggggatgggt</td>
</tr>
<tr>
<td>(HEK, HELA, MCF7, CHO) = (0, 1, 0, 1), $P = (0.44, 0.58, 0.22, 0.74)$</td>
<td>gaggatggaagcaacgggatggagtgg</td>
</tr>
<tr>
<td>(HEK, HELA, MCF7, CHO) = (0, 1, 0, 1), $P = (0.44, 0.58, 0.22, 0.74)$</td>
<td>gaggagatcggggaggggatgggt</td>
</tr>
</tbody>
</table>

**Table 11:** Top differential search results for a selection of cell lines and target logic expressions. 5' region and 3' region shows the randomized regions.