Inference of Computational Models of Alternative Polyadenylation and Splicing

by

Michael Ka Kit Leung

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy

Department of Electrical and Computer Engineering
University of Toronto

Inference of Computational Models of Alternative Polyadenylation and Splicing

Michael Ka Kit Leung

Doctor of Philosophy

Department of Electrical and Computer Engineering
University of Toronto

2018

Abstract

Instructions from the genome are first copied to make messenger RNAs, which are then translated to make proteins. To expand the repertoire of these instructions, cells can modify the messenger RNAs in different ways. Two such modifications are alternative polyadenylation and alternative splicing. Using RNA-Seq data and deep learning, we trained computational models that can be applied to sequences in the genome to predict tissue-specific polyadenylation and splicing patterns. Presented with multiple alternative polyadenylation sites, the polyadenylation model can predict the probability each site would be selected for cleavage and polyadenylation. Similarly, given alternative exons, the splicing model can predict which exon would more likely be included. The performance of these models in predicting polyadenylation and splicing patterns for genomic regions not observed during training is evaluated, and an analysis of what the models have learned reveals sequence elements that are known to influence these cellular processes. Importantly, these computational models are trained on genome-wide patterns based on the reference genome but can generalize to individual variations. Each model can thus be viewed as a simulator, where the genotype of an individual can be fed in as an input, and the output describes how the individual's mutations affect the mechanisms of polyadenylation and splicing in different tissue types. The relevance of these models for problems in genomic medicine is described, and proof-of-concept applications are demonstrated.

Acknowledgments

I would like to express my gratitude to my supervisor, Professor Brendan Frey, for the opportunity to perform research in his lab. Even though I had no previous background in machine learning and computational biology, he took a chance in recruiting me into his group. I thank his mentorship and for providing a stimulating environment that fostered ownership and mastery of one's work. Within the Probabilistic and Statistical Inference Group, I was able to interact with many talented individuals, whose ability to perform high-quality research propelled me to aim for the same. I would like to thank the members of the lab, Andrew Delong, Babak Alipanahi, Hui Xiong, Leo Lee, Jimmy Ba, Alireza Makhzani, Alice Gao, Hannes Bretschneider, Oren Kraus, and Christopher Srinivasa for the insights gained during our interactions. Overall, graduate work has been an extremely rewarding experience, and I would have no reservations in doing it again.

I would also like to thank my parents and my sister for their patience in supporting me through my studies. It has been a long journey and your encouragement over the years is greatly appreciated.

Lastly, I would like to acknowledge the Natural Sciences and Engineering Research Council of Canada for providing the funding that assisted me through graduate work.

Table of Contents

List of Tables

List of Figures

Chapter 1
Machine Learning Models of Biological Systems for Genomic Medicine

The concept of precision medicine is not entirely new. Physicians have been using blood type to tailor blood transfusions for over a century (Collins and Varmus, 2015). What is different today is the rapid growth in genomic data that can be quickly and cheaply collected from the patient and the wider community, and the potential for insights from analyzing and sharing that data. Specifically, the theme of this thesis is on machine learning models of transcriptional modification and their relevance in genomic medicine, namely to assist the discovery of targeted therapies and to identify disease risks for potential preventative measures. This introductory chapter is based on the publication (Leung *et al.*, 2016):

M. Leung, A. Delong, B. Alipanahi, and B. Frey. (2016) "Machine Learning in Genomic Medicine: A Review of Computational Problems and Data Sets". Proceedings of the IEEE, 104(1), 179-197.

1 Introduction

1.1 The Genome

A genome can be viewed as an instruction book for building an organism. It has long been understood that DNA molecules are the physical medium of genetic information storage (Watson and Crick, 1953), and by 2001 the Human Genome Project had drafted the content of a typical human genome (Lander *et al.*, 2001; Lander, 2011). However, the bigger challenge was to interpret the structure, function, and meaning of the genetic information itself. Biologist Eric Lander summarized the situation in seven words (Lander, 2012): "Genome: Bought the book; Hard to read." Still, much is known about how genetic information is organized into distinct genes. Each gene is a like a chapter in the instruction book, describing how to build a particular family of molecules. Protein-coding genes describe how to build large molecules made from amino-acid chains (proteins), whereas non-protein-coding genes describe how to build small molecules made from ribonucleic acid (RNA) chains (Strachan and Read, 2010; Alberts *et al.*, 2002). Roughly, the human genome contains 20,000 protein-coding genes (de Klerk and 't Hoen, 2015), and 25,000 non-coding genes (Harrow *et al.*, 2012). Some genes are crucial for life, some are crucial for health, and some can be deleted in their entirety without apparent harm.

Currently, protein-coding exons are the most understood regions in the genome. The universal genetic code for proteins was experimentally confirmed over 50 years ago (Crick *et al.*, 1961), and knowing how a coding mutation changes the corresponding amino acid sequence is a standard feature in genome diagnostic pipelines. For example, if a mutation introduces a 'stop codon' into the sequence (called a 'nonsense' mutation) then it is known that the protein will be truncated as a general rule. However, predicting whether a mutation will disrupt the stability or structure of the final protein molecule is a long-standing open problem (Moult *et al.*, 2014). Furthermore, coding regions make up only ~1.5% of the human genome, even though there is evidence that at least 5% of positions undergo purifying selection – this means that ~3.5% of the genome may consist of non-coding elements with functional roles (Lindblad-Toh *et al.*, 2011). Disease-causing mutations are increasingly being found outside of protein-coding regions (Hindorff *et al.*, 2009), indicating that analysis tools for coding regions are not enough to study genetic diseases. Many of the functional non-coding positions are regulatory sequences, meaning they instruct the cell how to regulate important processes such as gene expression and the reliable identification of exons. This underscores the importance of developing computational models that can automatically identify and understand regulatory instructions in the genome. These regulatory elements contribute significantly to the complexity of cell biology, which cannot be accounted for only by the sheer number of genes (King and Jukes, 1969) (e.g. balsam poplar trees have twice as many genes as humans (Tuskan *et al.*, 2006)) or the coding regions themselves (e.g. less than 1% of human genes have protein-coding regions that are distinct from those of mice and dogs (Clamp *et al.*, 2007)).

Genetic diseases can arise from mutations outside the protein-coding region of the genome. One example is spinal muscular atrophy (SMA) which is the leading genetic cause of infant mortality in North America (Cartegni and Krainer, 2002). It results if a baby's genome is missing the *SMN1* gene, or contains a damaged version of it, resulting in the deficient production of the survival motor neuron (SMN) protein. Another version of the gene, called *SMN2*, can compensate for the production of the SMN protein. Figure 1-1 shows the nucleotide sequence from the seventh exon of the protein-coding gene *SMN2*. Due to differences in nucleotides at the four positions shown, the cell's machinery usually fails to include the exon, resulting in a protein that does not function properly, and thereby is unable to compensate for the production of the SMN protein. Researchers are evaluating therapies that restore the function of exon 7 in *SMN2* (Hua *et al.*, 2011;

Naryshkin *et al.*, 2014). SMA is well-studied and can be diagnosed by outward symptoms, but genetic testing is crucial for confirmation and therapeutic development. In other genetic diseases, the mechanisms are more complex. Cancer is a prime example of a heterogeneous disease, that is, a disease with multiple causal pathways all leading to similar symptoms but requiring different treatments (Hanahan and Weinberg, 2011). For cancer, genomic data is becoming essential for providing more detailed diagnoses and targeted treatments (Rubin, 2015).

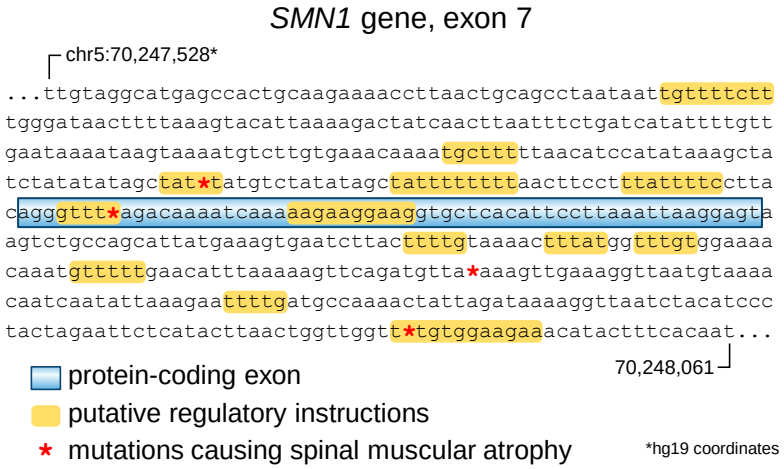


Figure 1-1. An exon and the regulatory instructions identified using machine learning. If an infant is homozygous in a version of the survival motor neuron gene, *SMN2*, the result is spinal muscular atrophy, a leading cause of infant mortality. Three of the nucleotides lie within genomic instructions that a neural network identified as being important for including this exon when building the protein (Xiong *et al.*, 2015).

How can we learn to read the genome to help diagnosis and treat genetic diseases? Unlike familiar cognitive tasks such as visual object detection and speech recognition, humans are not naturally equipped to perceive and interpret genomic sequences nor to understand all the mechanisms, pathways, and interactions that go on inside a living cell. To overcome this, one strategy is a data-driven approach wherein the inputs and outputs of a biological system are measured, and then a predictive model is built. If the model can generalize to unseen input, for example, a previously uncharacterized region of the genome, it likely has uncovered features that are predictive of the output. The model can subsequently be probed to reveal insights related to the regulation of the output phenomenon. This task of mapping an input to an output is a common problem in machine learning, specifically supervised learning.

1.2 Computational Models of Biology

Predicting phenotypes (e.g., traits and disease risks) from the genome is in principle, a supervised machine learning problem. The inputs are a stretch of DNA sequence (genotype) relevant to the underlying biology, and the outputs are the phenotypes. In this thesis, the mapping from the sequence elements and cell state (such as a tissue type) to the prediction of a regulatory pattern is referred to as a 'regulatory code'. The task is akin to 'traditional' machine learning problems like computer vision or natural language processing. However, there are important differences in prediction tasks that are found in computational models of biology. In traditional machine learning problems, a human 'oracle' is able to provide reasoning and truths as to why a model (e.g. a classifier) ought to make a particular prediction given some input. However, in biology (and many other physical sciences for that matter), such information is not generally available. Another difference is the 'completeness' of information in the inputs. For example, in computer vision, all the information needed for a prediction is more or less completely given from the pixels of an image. In biology, one can only get a glimpse of what's happening inside a cell or a population of cells, often in constrained regions, from available choices of experimental techniques. These limited measurements mean that the computational model must infer many of the hidden factors that contribute to regulation which are not readily available via current measurement technologies, from data. Arguably, this makes computational biology more difficult than traditional machine learning problems.

1.2.1 Assessing Disease Risks via Molecular Phenotype Prediction

One of the goals of genomic medicine is to predict phenotypes, such as disease risks, from a genotype. However, there are two reasons in which directly mapping genomic sequences to complex phenotypes and diseases may not be ideal. First is the complexity of the relationship between a full genotype and its phenotype. Even within a single cell, the genome directs the state of the cell through many layers of intricate and interconnected biophysical processes and control mechanisms that have been shaped ad-hoc by evolution. Attempting to infer the outcomes of these complex regulatory processes by observing only genomes and phenotypes is rather like trying to learn how computer chess playing programs work by examining binary code and wins and losses, while ignoring which moves were taken. Second, even if one could infer such models (those that are predictive of disease risks), it is likely that the parameters of these models would not correspond to biological mechanisms that can be acted upon. Insight into disease mechanisms is

important for the purpose of developing targeted therapies, but can also provide complementary information for phenotypic screens, which traditionally identifies chemicals with desired biological effects without knowledge of the precise targets (Moffat *et al.*, 2014).

An alternative approach is to train computational models to predict measurable variables of the cell, also known as molecular phenotypes, and then these variables can be linked to phenotype. For example, in the case of spinal muscular atrophy described above, the cell variable could be the frequency with which the exon is included when the gene is being copied to make a protein. Other examples of cell variables include the locations where a protein binds to a strand of DNA containing a gene, the number of copies of a gene (transcripts) in a cell, the distribution of proteins along the transcript, and concentration of proteins. This approach addresses the two aforementioned problems. Since these cell variables are more closely related to and more easily determined from genomic sequences than phenotypes like diseases, learning models that map from DNA to cell variables can be more straightforward. High-throughput assay technologies are generating massive amounts of data profiling these cell variables under diverse conditions, and these datasets can be used to train larger and more accurate models. Also, since the cell variables correspond to intermediate biochemically active quantities, such as the concentration of a gene transcript, they are good targets for therapies. If high disease risk is associated with a change in a cell variable compared to a healthy individual, an effective therapy may consist of restoring that cell variable to its normal state. In the above example of spinal muscular atrophy, therapies that modify the genomic instructions to increase the frequency with which the exon is included in the protein are currently being tested in clinical trials (Hua *et al.*, 2011).

1.3 Role of Machine Learning to Model Biological Systems

In recent years, machine learning researchers have focused their most high-profile efforts on speech recognition (Graves *et al.*, 2013) and computer vision (Krizhevsky *et al.*, 2012). Computer vision in particular has a long history in machine learning due to its intuitive, accessible nature. Human beings are exceedingly good at computer vision tasks, and so when our learning algorithms do not satisfy our own expectations we often find new insights. In fact, the hand-written digit recognition dataset known as MNIST (Y. LeCun, Cortes, *et al.*, 1998) has been called 'the Drosophila of machine learning'—a reference to the fruit fly model organism in biology—owing to MNIST's widespread use as a testbed for new learning algorithms.

The situation in biology is fundamentally different from the situation in computer vision. The visual world is directly accessible to us, and researchers exploit our knowledge of how images are generated through light, occlusion, and projection. The nano-scale world of a cell's machinery is not directly accessible and, despite decades of painstaking effort, our knowledge of the mechanisms at play is woefully incomplete (Ritchie *et al.*, 2015; Albert and Kruglyak, 2015). This is true even for single-cell organisms like yeast (Costanzo *et al.*, 2010; Lehner, 2013). Determining how genotype affects phenotype is arguably orders of magnitude more complex than the pixels-to-labels relationship in high-profile vision challenges such as ImageNet (Deng *et al.*, 2009). The details of many interactions, quantities, and processes in the cell are 'hidden' from us because we do not have the technology to systematically measure them. In other words, the few cell variables that we can observe are the outcome of many layers of interacting cell variables that we cannot.

To build computational models of molecular phenotypes, assays to measure the corresponding biological quantities must exist, and training data must be collected under many conditions. Well into the 1990s, biological assays typically required several manual steps and generated small amounts of data. Such techniques are useful for developing and testing hypotheses, but do not provide sufficient data to infer accurate predictive models of complex outcomes. With the commoditization of high-throughput assay technologies, it is now commonplace to acquire hundreds of thousands of measurements for a cell variable in a single low-cost experiment. For example, microarray technology has been used to peer into living cells for decades (Schena *et al.*, 1995), but new assays and new chemistry are still being developed around this fundamental approach, such as universal protein binding microarrays (Berger *et al.*, 2006), ChIP-chip (Ren *et al.*, 2000) and RNAcompete (Ray *et al.*, 2009). High-throughput sequencing technologies are likewise being used for a wide range of tasks (Metzker, 2010), for example, identifying protein binding sites, sequencing the genomes of different organisms in evolutionary studies, and profiling the genomes of individuals in medical studies for the purpose of discovering variations, either in regions of interest or across the entire genome.

In addition to measuring genotypes on a large scale, high-throughput technologies can be used to measure molecular phenotypes, such as the abundances of different transcripts (Mortazavi *et al.*, 2008). Although somatic mutations, which are alterations in the DNA after conception, can occur in cancers and some neurological diseases (Watson *et al.*, 2013; Poduri *et al.*, 2013), the genome of an individual is relatively stable. The 'transcriptome' on the other hand, varies from

cell to cell, and is affected by the cell's surrounding environment, for example, the tissue type it represents. Previously, microarrays were used to measure transcripts on a large scale, but now high-throughput sequencing is the method of choice. Another application of high-throughput sequencing is to profile how proteins interact with specific regions of DNA (Johnson *et al.*, 2007). Binding of proteins can influence how the instructions in the genome are utilized, offering a layer of complexity that can be exploited for regulation of cell biology. Data such as these, which measure particular cell variables of interest, allow us to peer into the underlying workings of the cell, at the most fundamental level of the instructions that define an organism. High-throughput assay technologies have made it feasible to measure cell variables of interest covering vast portions of the genome at various cell states, including disease conditions, and a wealth of data is now publicly available. This presents an exceptional opportunity for data scientists to infer predictive models of cell variables using machine learning techniques.

The inputs to the computational model include sequence characteristics from a stretch of DNA, such as the frequency of particular nucleotides or presence of certain motifs, some of which can be learned from the sequence themselves (Alipanahi *et al.*, 2015). To account for instructions encoded in the DNA that impact cell variables through biochemical processes and structures, additional features can be derived, for example, the binding of proteins to DNA and RNA, nucleosome positioning and occupancy profiles (van der Heijden *et al.*, 2012), and RNA secondary structures (Lorenz *et al.*, 2011). Generally, it is beneficial that the model's inputs be fully extractable from DNA sequences. For a computational model to be practical in making predictions in the context of genomic medicine, it is desirable for the inputs to be easily obtainable. Given that the cost of whole genome sequencing continues to rapidly decrease, a growing number of genomes will be available for training purposes, and within the context of genomic medicine, it will likely become standard for a patient's genome to be available.

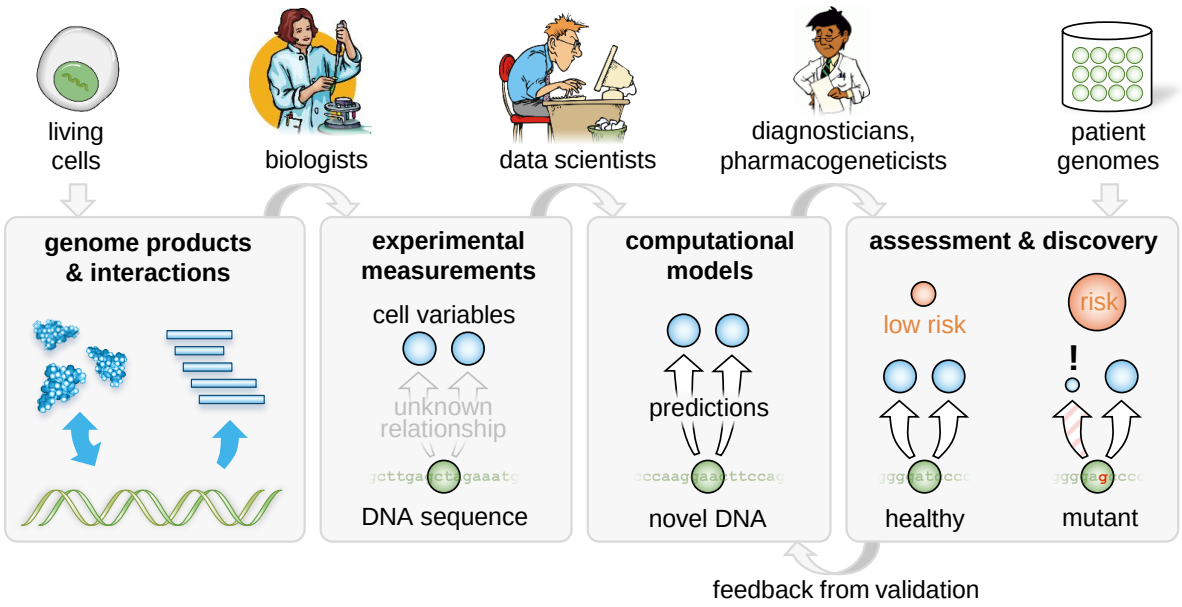


Figure 1-2. A simplified view of how biologists, data scientists, and medical researchers can work towards genomic medicine (Leung *et al.*, 2016). Machine learning plays a central role by turning high-throughput measurements into specialized or general-purpose predictive models for cell variables—quantities that are relevant to cell function. By knowing how mutations affect disease via cell variables, diagnosticians and pharmacogeneticists can more easily find direct correlates with disease, develop treatments, and plan targeted therapies for individual patients.

Figure 1-2 illustrates a workflow through which different actors participate towards the goal of genomic medicine. An important aspect of the approach is the use of machine learning to infer models that are capable of generalizing to new genetic contexts. For example, one may infer a model using the publicly available reference genome and data profiling transcripts in healthy tissues, but then apply it to the genome of a diseased cell and ascertain how the distribution of transcripts changes in the diseased cell. This notion of generalization is a crucial aspect of the models that need to be inferred. From a modeling perspective, a greater ability to generalize to new genetic context is expected for those cell states that were observed during training. Consequently, an important aspect of model development is validation using DNA sequences that the model has never seen before and using data for cell states that are different from those used during training. One cannot expect the models to be accurate for any DNA sequences and cell states that are extremely different from those used during training, so the validation procedure should also attempt to characterize the inputs for which the model is reliable. If a model is good at generalization, it can analyze mutated DNA sequences that lead to changes in cell variables that may be indicative of disease state, without needing experimental measurements from diseased

cells. In practice, this kind of 'zero-shot' learning has been successfully used to identify mutations that cause a variety of diseases, using a model that was trained using the reference genome and healthy tissues (Xiong *et al.*, 2015).

While the model that predicts cell variables does not directly take into account information pertaining to disease, if the model accurately reflects how the instructions in the genome are processed, then it should be able to detect diseases that are caused by mutations that change cell variables. This approach has been shown to work very well for a large number of mutations and disease (Xiong *et al.*, 2015), but of course it makes errors. If mutations are scored by how much they cause a change in the cell variables, then false positives will arise when a mutation causes a large change in cell variables that have no impact on disease. For example, a mutation can change a cell variable which leads to a change in hair color. False negatives will arise for mutations that act through cell variables that are not being modeled. Both kinds of error will also arise due to inaccuracies in the computational models. When investigating specific diseases, scores for mutations can be combined with disease-specific data, such as population data. In this way, sets of candidate mutations can be filtered to identify the ones that are most likely to have a causal effect on a cell variable. More generally, these scores can be used as input features for models that are specific to certain diseases, where the models may utilize many such scores across multiple regions in the genome.

Computer systems that can read the text of the genome can be used in a variety of ways to support genomic medicine. For example, a recent breakthrough in 'gene editing' is allowing scientists to alter the genomes of already-living cells, with an efficacy no one thought possible just a few years ago. Gene therapies can now include targeted modifications, such as removing deleterious mutations or even inserting new sequences at pre-determined locations in a genome. Genome editing technology (Cong *et al.*, 2013; Mali *et al.*, 2013) opens a door to unprecedented opportunities in genomic medicine, making it more important than ever that we can predict the effects of these edits *in silico*. In other words, knowing *how* to write is not the same as knowing *what* to write.

1.4 Other Approaches to Model the Genetic Basis of Disease Risks

In this section, some approaches to associate genetic variants with disease risks are described. These methods directly model the genotype-to-phenotype relationship and are in common use. The goal is to provide a more holistic view of other methods that tackle this problem, which do not necessarily require the use of machine learning, and their challenges.

1.4.1 Genome-Wide Association Studies

The goal of genome-wide association studies (GWAS) is to detect how traits within a population can be related to variants in particular genomic locations, or loci. Early GWAS experiments used microarrays that were designed to find common variants in the human population, called single-nucleotide polymorphisms (SNPs), which are variations that are relatively frequent across humans (frequency greater than 1%). Modern GWAS analysis use more comprehensive sets of variants and even whole genome sequencing data, which is not restricted to a subset of variants. Here, we focus on some of the challenges of GWAS (Visscher *et al.*, 2012).

From a data analysis point of view, one of the main difficulties with GWAS is establishing statistical significance between a potentially causal variant with a change in risk for particular diseases between the affected group of individuals compared to a control. The main problem in GWAS and any association-based technique (e.g., expression quantitative trait loci or eQTLs) is that they indicate correlation, not causation. Due to confounding hidden variables, such as correlations between nearby variants caused by crossing over (linkage disequilibrium) or differences in subpopulations caused by factors such as migration, two or more genomic loci might be correlated, and a SNP could be picked up by GWAS, simply because some other genomic locus is causal (Pritchard and Przeworski, 2001). The causal variant is often not even observed in the GWAS study. GWAS furthermore provides a huge number of putative causal mutations, and researchers may be biased towards candidates that have greater 'narrative potential' (Goldstein *et al.*, 2013).

Some of the bigger GWAS studies involve tens of millions of SNPs that are conducted on thousands of individuals. Assessing the statistical significance of an immense number of SNPs is challenging and requires careful multiple-hypothesis correction or false discovery rate analysis (Johnson *et al.*, 2010). The problem is compounded by the fact that many common variants have

weak effects, and those that have strong effects tend to be rare (Gibson, 2012). To improve significance, some studies limit the profiling of SNPs to the coding region of the genome (Bamshad *et al.*, 2011), with the assumption that mutations in these regions are more likely to impact risks as they can alter the function of the proteins (Ng *et al.*, 2009). Another way to address this problem is by increasing the sample size. Significant resources have been channeled into large population study initiatives such as The Cancer Genome Atlas (TCGA) and The International Haplotype Map (HapMap) Project, which have raised debates within the research community on the cost-benefit ratio of these projects (The International HapMap Consortium, 2005; Altshuler *et al.*, 2010; Ledford, 2015). Another major hurdle is population structure and its stratification. A recent paper (Skafidas *et al.*, 2014) that used a genetic classifier based on SNPs for Autism Spectrum Disorder detection, raised some controversy (Robinson *et al.*, 2014) and it was alleged that most of the observed signal was due to 'potential population stratification' (genetic differences due to ancestry).

One approach to make better use of GWAS data beyond statistical associations is to use computational models that take as input the SNP profiles of individuals to predict disease risks. These SNP profiles tend to be high-dimensional, and typically have a large proportion of SNPs that are not relevant to the disease at hand (and therefore noisy). Several tools are available for prioritizing causal variants (e.g. PolyPhen (Adzhubei *et al.*, 2010), SIFT (Kumar *et al.*, 2009), SPANR (Xiong *et al.*, 2015)), and machine learning algorithms have been used for learning predictive models of disease risks (Kooperberg *et al.*, 2010; Kruppa *et al.*, 2012).

1.4.2 Evolutionary Conservation

Comparative genomics is a powerful way to identify genomic sequences that have function. The most well-known resource that comparative genomics provides is sequence conservation. The rationale behind sequence conservation is as follows. First consider evolution as being driven by two forces: the slow accumulation of random mutations, and selective pressures against mutations that damage reproductive fitness within a population (Ureta-Vidal *et al.*, 2003). Now consider the genomes of several species that diverged from a common ancestor long ago; long enough that random mutations have had plenty of time to occur. When we compare the genomes, we find many long distinct sequences that are nearly identical, or 'conserved', across species. When a sequence is conserved, it is strong evidence that evolution is exerting selective pressure on the

positions within those sequences. Studies estimate at least 5-6% of the human genome is conserved with mammals (Dermitzakis *et al.*, 2003; Lindblad-Toh *et al.*, 2011).

Detection of conserved sequences has been instrumental in annotating functional elements in the human genome (Lindblad-Toh *et al.*, 2011), such as exons. Conservation scores are available for multiple organisms from the software tool phastCons (Siepel *et al.*, 2005). For each position in a genome, phastCons provides a number between 0 and 1, where 0 indicates no discernible conservation, and 1 indicates 100% conservation across all species considered. Other methods for quantifying conservation include GERP (Cooper *et al.*, 2005) and phyloP (Pollard *et al.*, 2010). Conservation scores for each position in the human genome can be viewed online as a 'track' within the UCSC Genome Browser (James Kent *et al.*, 2002).

A mutation that lowers reproductive fitness is called deleterious, whereas a mutation that causes a disease is called pathogenic (Goldstein *et al.*, 2013). Many mutations are of course both deleterious and pathogenic, such as mutations causing Tay Sachs disease, but it is important to understand that conservation only suggests regions of the genome that are 'constrained' via purifying selection (Cooper *et al.*, 2010). Even so, conservation-based techniques have been an extremely useful input feature for predictive models of disease. One recent example is the combined annotation dependent depletion (CADD) method (Cooper *et al.*, 2010). Kircher et al. first developed a 'mutation simulator' that generates realistic synthetic mutations without regard to selective pressure. They then trained an ensemble of ten linear support vector machines to discriminate between synthetic mutations (assumed deleterious) and the ~16 million actual human mutations that have survived selective pressure (non-deleterious) since the human-chimpanzee common ancestor.

1.5 Machine Learning Models of Molecular Phenotypes

The underlying theme of this thesis is to apply machine learning to model molecular phenotypes that can be assayed via next-generation sequencing. Specifically, the molecular phenotypes to be modeled are splicing and polyadenylation, and the input to the model are genomic sequences. These models are viewed in the context of genomic medicine, where we provide proof-of-concept applications. The concept of applying machine learning to model molecular phenotypes is by no means novel. However, this approach has gained significant momentum by two recent developments.

The first of these is the arrival of next-generation sequencing technologies that made it possible to profile many organisms at scales larger than ever before at manageable costs (Metzker, 2010). This introduced petabytes of data in the public domain, and is growing exponentially each year (Marx, 2013). Starting out as a means to profile differences in the genome of species and individuals, it has extended into the cataloging of the transcriptome of individual cell types and tissues. For example, technologies such as RNA-Seq has been responsible for generating transcriptomic data volumes that are exceeding that of genomic profiling (Wang *et al.*, 2009), and has largely displaced technologies like microarray. The availability of larger than ever before volumes of data means that one can potentially afford to train larger computational models for certain biological processes.

The second of these developments is a field of machine learning called 'deep learning', which generally refer to methods that map data through multiple levels of abstraction, where higher levels represent more complex entities (Bengio, 2009). One of the promises of deep learning is they can learn complex functions that map inputs to outputs, without using hand-crafted features or rules. Given enough data, these models have large capacities for representing complex relationships efficiently, and their highly-parallel nature means they can be trained rapidly with many processing cores hardware like graphics processing units (GPU). In recent years, deep learning methods have surpassed the state-of-the-art performance for many machine learning tasks in computer vision and natural language processing (Bengio *et al.*, 2013). There are three ingredients for their successes (Deng and Yu, 2014). The first is the availability of lower-cost hardware with increased processing capabilities, like those offered by GPUs. The second is due to new advances in deep learning methods that can effectively train these complex models. The final ingredient is the availability of large datasets that these models can learn from. The first two ingredients are not domain-specific. With the increasingly rapid growth in the volume of 'omic' (e.g. genomics, transcriptomics, and proteomics) data, one may now have the final ingredient necessary for the successful application of deep learning to problems in biology. Due to their layered structure, deep learning has the potential to produce meaningful and hierarchical representations that can efficiently be used to describe complex biological phenomena. For example, deep neural networks may be useful for modeling multiple stages of a regulatory mechanism at the sequence level followed by higher levels of abstraction. A large body of work

now exists which leverages deep learning in the life sciences (Ching *et al.*, 2018), which was not the case in as little as five years ago at the beginning of this thesis.

In the following chapters, the application of deep learning to model next generation sequencing assays of alternative splicing and alternative polyadenylation is described.

Chapter 2
Inference and Analysis of a Mouse Splicing Code

Transcriptional modifications refer to the processes by which cells modify the primary transcript, the single-stranded ribonucleic acid (RNA) transcribed from the DNA, that contains the instructions to maintain the identity of a cell and direct its function. By modifying the transcript, cells can have different functions even though they share the same genome. Modification of these messages is a regulated process that depends on the complex interactions between sequences on the RNA and the presence of other proteins. The major processes in transcriptional modifications are 5'-end capping, splicing, and 3'-end cleavage and polyadenylation (Bentley, 2014). This chapter describes the development of a computational model to infer alternative splicing patterns from mouse RNA-Seq data, building upon the work from (Barash *et al.*, 2010). The content here is based on the publication (Leung *et al.*, 2014):

M. Leung, H. Xiong, L. Lee, and B. Frey. (2014) "Deep learning of the tissue-regulated splicing code". Bioinformatics, 30(12), i121-i129.

2 The Mouse Splicing Code

2.1 Introduction

Alternative splicing (AS) is a process whereby the exons of a primary transcript may be connected in different ways during pre-mRNA splicing. This enables the same gene to give rise to splicing isoforms containing different combinations of exons, and as a result different protein products, contributing to the cellular diversity of an organism (Wang and Burge, 2008). Furthermore, AS is regulated during development and is often tissue-dependent, so a single gene can have multiple tissue-specific functions. The significance of AS lies in the evidence that at least 95% of human multi-exon genes are alternatively spliced, and that the frequency of AS increases with species complexity (Pan *et al.*, 2008; Barbosa-Morais *et al.*, 2012).

One mechanism of splicing regulation occurs at the level of the sequences of the transcript. The presence or absence of certain regulatory elements can influence which exons are kept, while others are removed, before a primary transcript is translated into proteins. Computational models that take into account the combinatorial effects of these regulatory elements have been successful in predicting the outcome of splicing (Barash *et al.*, 2010).

Previously, a 'splicing code' that utilizes a Bayesian neural network (BNN) was developed to infer a model that can predict the outcome of alternative splicing from sequence information in different cellular contexts (Xiong *et al.*, 2011). One advantage of Bayesian methods is that they are more robust against overfitting by integrating over models. Rather than using a single set of parameters (typical of methods based on maximum likelihood or maximum a posterior) which defines, for instance a neural network, when making predictions, the Bayesian approach uses a distribution of the parameters of the model. This means that instead of relying on a specific configuration of the parameters (which may fit the training data well, but not the test data), the Bayesian approach makes predictions based on different configurations of the parameters of the model from a posterior distribution. When the training data is sparse, as is the case for many datasets in the life sciences, the Bayesian approach can be beneficial. It was shown that the BNN outperforms several common machine learning algorithms, such as multinomial logistic regression and support vector machines, for AS prediction in mouse trained using microarray data.

There are several practical considerations when using BNNs. They often rely on methods like Markov Chain Monte Carlo (MCMC) to sample models from a posterior distribution, which can be difficult to speed up and scale up to a large number of hidden variables and a large volume of training data. Furthermore, computation-wise, it is relatively expensive to get predictions from a BNN, which requires computing the average predictions of many models.

Recently, deep learning methods have surpassed the state-of-the-art performance for many tasks (Bengio *et al.*, 2013). Deep learning generally refers to methods that map data through multiple levels of abstraction, where higher levels represent more abstract entities. The goal is for an algorithm to automatically learn complex functions that map inputs to outputs, without using hand-crafted features or rules (Bengio, 2009). One implementation of deep learning comes in the form of feedforward neural networks (LeCun *et al.*, 2015), where levels of abstraction are modeled by multiple non-linear hidden layers.

With the increasingly rapid growth in the volume of 'omic' data (e.g. genomics, transcriptomics, proteomics), deep learning has the potential to produce meaningful and hierarchical representations that can efficiently be used to describe complex biological phenomena. For example, deep networks may be useful for modeling multiple stages of a regulatory network at the sequence level and at higher levels of abstraction.

Ensemble methods are a class of algorithms that are popular due to their generally good performance (Caruana and Niculescu-Mizil, 2006), and are often used in the life sciences (Touw *et al.*, 2013). The strength of ensemble methods comes from combining the predictions of many models. Random forests is an example, as is the Bayesian model averaging method previously used to model the regulation of splicing. Recently, training neural network has been improved using a technique called dropout, which makes neural networks behave like an ensemble method (Hinton *et al.*, 2012). Dropout works by randomly removing hidden neurons during the presentation of each training example. The outcome is that instead of training a single model with N hidden variables, it approximates the training of 2^N different networks, each on a different subset of the training data. It is described as an 'extreme form of bagging', and is a very computationally efficient way of doing model averaging (Hinton *et al.*, 2012).

With large datasets, learning with MCMC methods can be slow, and can be outperformed by stochastic optimization methods in practice (Ahn *et al.*, 2012). These algorithms process small subsets (minibatches) of data at each iteration, and update model parameters by taking small steps in the direction of the gradient to optimize the cost function (usually the log likelihood). It is common to use stochastic gradient descent to train feedforward neural networks (Y. A. LeCun *et al.*, 1998). The learning algorithm (backpropagation) is also conceptually simple. Briefly, training a feedforward neural network involves:

1. Given a set of B training inputs (a minibatch) $\mathbf{x} = x_1, x_2, \ldots x_B$, for all hidden units in a layer, scale $\mathbf{x}$ by the weights that connects $\mathbf{x}$ to the hidden unit, perform a summation, and compute the activation of the hidden unit. Repeat this step for any additional layers, where the activations of the previous layer serve as input for the following layer. This is the forward propagation step.
2. Compute the gradient of the cost function. This is a generally a function of the output of the neural network after the feedforward step $\mathbf{h} = h_1, h_2, \ldots h_B$ and the training target $\mathbf{y} = y_1, y_2, \ldots y_B$ (Bishop, 2006). The magnitude of the gradient is large when the difference between $\mathbf{y}$ and $\mathbf{t}$ is large.
3. Using the chain rule, compute the gradient of the cost function with respect to all weights and biases (the parameters) of the neural network. The gradients are computed in reverse topological order of the forward propagation step. This is the back propagation step.

4. Each parameter of the neural network is updated with the average of the *B* gradients from the minibatch, scaled by a learning rate.

These computations are highly parallel and consist of, for the most part, matrix operations, which makes them suitable for speed up using graphics processing units (GPU).

In this chapter, we show that the use of large (many hidden variables) and deep (multiple hidden layers) neural networks can improve the predictive performances of the splicing code compared to previous work. We also provide an evaluation method for researchers to improve and extend computational models for predicting AS. Another goal is to describe the procedure for training and optimizing a deep neural network (DNN) on a sparse and unbalanced biological dataset. Furthermore, we show how such a DNN can be analyzed in terms of its inputs.

We show results supporting that DNN with dropout can be a competitive algorithm for doing learning and prediction on biological datasets, with the advantage that they can be trained quickly, have enough capacity to model complex relationships, and scale well with the number of hidden variables and volume of data, making them potentially highly suitable for 'omic' datasets.

Different from the previous BNN, which used 30 hidden units, our architecture has thousands of hidden units with multiple non-linear layers and millions of model parameters. We also explored a different connection architecture compared to previous work. Before, each tissue type was considered as a different output of the neural network. Here, tissues are treated as an input, requiring that the complexity of the splicing machinery in response to the cellular environment be represented by a set of hidden variables that jointly represent both the genomic features and tissue context.

Besides a different model architecture, we also extended the code's prediction capability. In previous work, the splicing code infers the direction of change of the percentage of transcripts with an exon spliced in (PSI), relative to all other tissues (Katz *et al.*, 2010). Here, we perform absolute PSI prediction for each tissue individually without the need for a baseline averaged across tissues. We also predict the difference in PSI (ΔPSI) between pairs of tissues to evaluate the model's tissue-specificity. We show how these two prediction tasks can be trained simultaneously, where the learned hidden variables are useful for both tasks.

We compare the splicing code's performance trained with the DNN with the previous BNN and additionally optimized a multinomial logistic regression (MLR) classifier on the same task for a baseline comparison. A GPU was used to accelerate training of the DNN, which made it feasible to perform hyperparameter search to optimize prediction performance with cross-validation.

2.2 Methods

2.2.1 Dataset

The dataset consists of 11,019 mouse alternative exons profiled from RNA-Seq data prepared by (Brawand *et al.*, 2011). The exons are based on a set of cassette exons (an exon that can be spliced out or retained, flanked by two constitutive exons) derived from EST/cDNA sequences (Barash *et al.*, 2010, 2013). Five tissue types are available, including whole brain, heart, kidney, liver, and testis. To estimate the splicing level for each exon and tissue, we mapped the 75 nucleotide reads to splice junctions, requiring a minimum overhang of 8 nucleotides on each side of the junction, giving 60 mappable positions. A bootstrap method that takes into account position-dependent read biases in RNA-Seq was then used to obtain an estimate of PSI that reflects its uncertainty (Kakaradov *et al.*, 2012). This generates a distribution of PSI for each exon and tissue, which represent the probability that the given exon within a tissue type has PSI value ranging from these intervals. Three real-values are then calculated by summing this probability mass over equally split intervals of 0 to 0.33 (*low*), 0.33 to 0.66 (*medium*), and 0.66 to 1 (*high*), representing targets for classification. Table 2-1 shows the distribution of exons in each category in this dataset, counted by selecting the label with the largest value. It can be seen that most of the exons in the dataset are included in the transcript (high PSI), followed by exons that are excluded (low PSI). Exons that are sometimes included or excluded (medium PSI) are the minority.

Table 2-1. The number of exons classified as *low*, *medium*, and *high* for each mouse tissue. Exons with large tissue variability (TV) are displayed in a separate column. The proportion of *medium* category exons that have large tissue variability is higher than the other two categories.

	Brain		**Heart**		**Kidney**		**Liver**		**Testis**	
	All	**TV**	**All**	**TV**	**All**	**TV**	**All**	**TV**	**All**	**TV**
Low	1782	579	1191	460	1287	528	1001	413	1216	452
Medium	669	456	384	330	345	294	254	220	346	270
High	5229	1068	4060	919	4357	941	3606	757	4161	887
Total	7680	2103	5635	1709	5989	1763	4861	1390	5723	1609

To obtain the distribution denoting the difference in PSI, or ΔPSI, the difference between bootstrap samples was calculated for all pairs of tissues to generate a distribution in the same manner. The possible values range from 0 to 1 for the PSI distribution, and -1 to 1 in the ΔPSI distribution. To test whether the model generalizes to a different dataset, RNA-Seq data from (Barbosa-Morais *et al.*, 2012) was processed in the same manner for brain and heart, which was used only for testing.

For each exon, a set of intronic, exonic, and structural features was derived from sequences in the alternative exon (A), flanking constitutive exons (C1 and C2), and introns between C1 and A (I1) and A and C2 (I2), forming a feature vector of length 1393. These features include those originally described in (Barash *et al.*, 2010) and the extended feature set from (Barash *et al.*, 2013). Features related to the premature termination codon have been removed since they rely on knowing the splicing pattern a priori and cannot be computed by just the local genomic sequences. Instead, four binary 'translatability' features are introduced, which describe whether exons can be translated without a stop codon in one of three possible reading frames. The features are summarized in Appendix A.

2.2.2 Model

We formulate splicing prediction as a classification problem with multiple classes. Figure 2-1 shows the architecture of the DNN. The nodes of the neural network are fully connected, where each connection is parameterized by a real-valued weight θ. The DNN has multiple layers of non-linearity consisting of hidden units. The output activation a of each hidden unit v in layer l processes a sum of weighted outputs from the previous layer, using a non-linear function f:

$$a_v^l = f(\sum_m^{M^{l-1}} \theta_{v,m}^l a_m^{l-1})$$

where M^l represents the number of hidden units in layer l, and a^0 and M^0 are the input into the model and its dimensionality, respectively. We used two different activation functions for the hidden units, namely the *TANH* function, and the rectified linear unit (*RELU*), which is defined as (Glorot *et al.*, 2011):

$$f_{RELU}(z) = \max(0, z)$$

The *RELU* unit was used for all hidden units except for the first hidden layer, which utilized *TANH* units, based on empirical performance on validation data during hyperparameter optimization (see Section 2.2.3).

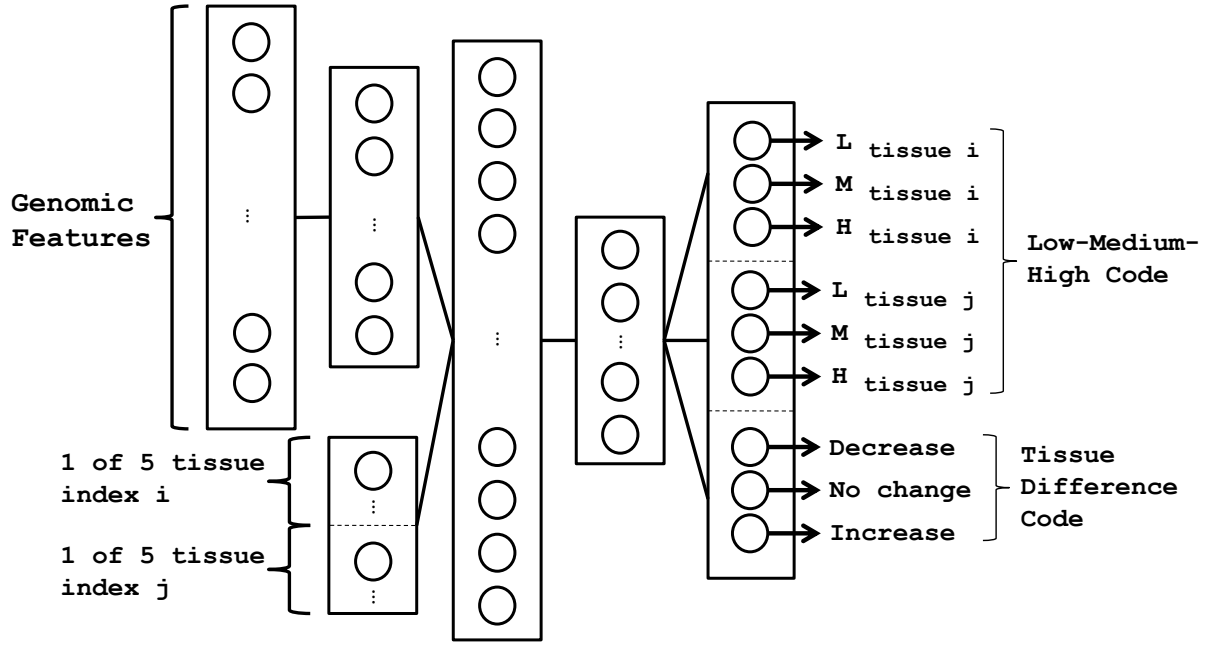


Figure 2-1. Architecture of the deep neural network used to predict alternative splicing patterns. It contains three hidden layers, with hidden variables that jointly represent genomic features and cellular context (tissue types).

Inputs into the first hidden layer consist of N = 1393 genomic features $x_{n=1...N}$ describing an exon, neighboring introns, and adjacent exons. To improve learning, the features were normalized by the maximum of the absolute value across all exons. The purpose of this hidden layer is to reduce the dimensionality of the input and learn a better representation of the feature space.

The identity of two tissues, which consists of two 1-of-T binary variables (one-hot encoding) $t_{i=1...T}$ and $t_{j=1...T}$, is then appended to the vector of outputs of the first hidden layer, together forming the input into the second hidden layer. For this work, T = 5 for the 5 tissues available in the RNA-Seq data. We added a third hidden layer as we found it improved the model's performance. The weighted outputs from the last hidden layer are used as input into a softmax function for classification in the prediction $h_j(x,t,\theta)$, which represents the probability of each splicing pattern j (e.g. low, medium, or high) for $j = 1, \ldots, K$:

$$h_j = \frac{\exp\left(\sum_m \theta_{j,m}^{last}\ a_m^{last}\ \right)}{\sum_k^K \exp(\sum_m \theta_{k,m}^{last}\ a_m^{last})}$$

To learn a set of model parameters θ, we used the cross-entropy cost function E on predictions $h(x,t,\theta)$ given targets $y(x,t)$, which is minimized during training:

$$E = -\sum_b \sum_{k=1}^{K} y_{b,k}\ log(h_{b,k})$$

where b denotes the training examples, and k indexes K classes.

We are interested in two types of predictions. The first task is to predict the PSI value given a particular tissue type and a set of genomic features. To generate the targets for training, we created $K = 3$ classes which we label as *low*, *medium*, and *high* categories. Each class contains a real-valued variable obtained by summing the probability mass of the PSI distribution over equally split intervals of 0-0.33, 0.33-0.66, and 0.66-1. They represent the probability that a given exon and tissue type has a PSI value ranging from these corresponding intervals, hence are soft class labels. We will refer to this as the '*low*, *medium*, *high*' (LMH) code, with targets $y_k^{LMH}(x, t_i)$.

The second task describes the difference in PSI between two tissues for a particular exon. We again generate 3 classes, and call them *decreased inclusion*, *no change*, and *increased inclusion*, which are similarly generated, but from the ΔPSI distributions. We chose an interval that more finely differentiates tissue-specific AS for this task, where a difference of greater than 0.15 would be labeled as a change in PSI levels. We summed the probability mass over the intervals of -1 to -0.15 for *decreased inclusion*, -0.15 to 0.15 for *no change*, and 0.15 to 1 for *increased inclusion*. The purpose of this target is to learn a model that is independent of the chosen PSI class intervals in the LMH code. For example, the expected PSI of two tissues t_i and t_j for an exon could be 0.40 and 0.60. The LMH code would be trained to predict medium for both tissues, whereas this tissue difference code would predict that t_j has increased inclusion relative to t_i. We will refer to this as the '*decrease*, *no change*, *increase*' (DNI) code, with targets $y_k^{DNI}(x, t_i, t_j)$.

Both the LMH and DNI codes are trained jointly, reusing the same hidden representations learned by the model. For the LMH code, two softmax classification outputs predict the PSI for

each of the two tissues that are given as input into the DNN. A third softmax classification function predicts the difference in PSI, or ΔPSI, for the two tissues. We note that two PSI predictions are included in the model's output so we have a complete set of predictions that utilize the full input features. The cost of the model used during optimization is the sum of the cross-entropy functions for both prediction tasks.

The BNN architecture used for comparison is the same as previously described (Xiong *et al.*, 2011), but trained on RNA-Seq data with the expanded feature set and LMH as targets. This is shown in Figure 2-2. Although hidden variables were shared across tissues in both the BNN and DNN, a different set of weights were used following the single hidden layer to predict the splicing pattern for each tissue separately in the BNN.

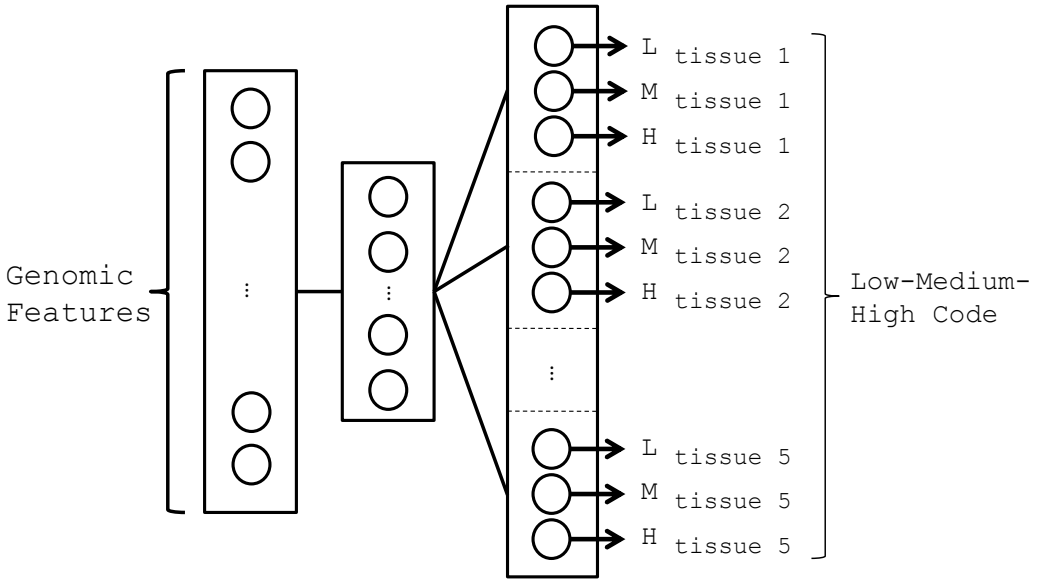


Figure 2-2. Architecture of the Bayesian neural network (Xiong *et al.*, 2011) used for comparison, where *low-medium-high* predictions are made separately for each tissue.

In the current DNN, the tissue identities are inputs and are jointly represented by hidden variables together with genomic features. In order for the BNN to make tissue difference predictions in the same manner as the DNI code, we fitted a MLR on the predicted LMH outputs for each tissue pair. This is illustrated in Figure 2-3.

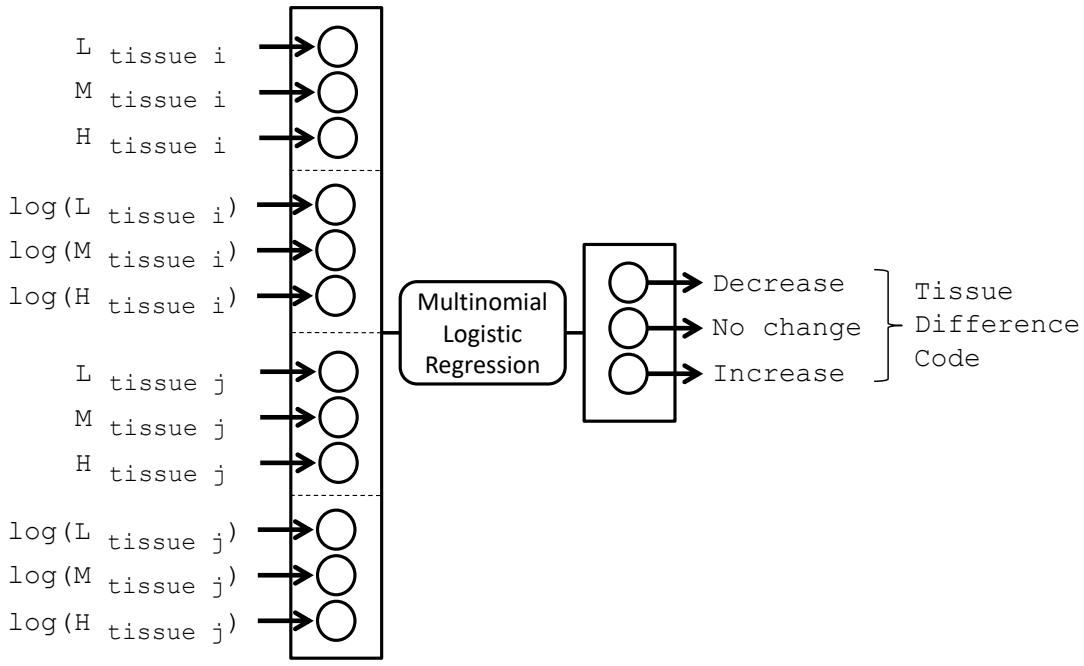


Figure 2-3. Input and output configuration for training a multinomial logistic regression classifier to utilize the outputs of the *low-medium-high* code to make tissue difference predictions.

2.2.3 Training

The first hidden layer was trained as an autoencoder to reduce the dimensionality of the feature space in an unsupervised manner. An autoencoder is trained by supplying the input through a non-linear hidden layer, and reconstructing the input, with tied weights going into and out of the hidden layer. This method of pretraining the network has been used in deep architectures to initialize learning near a good local minimum (Hinton and Salakhutdinov, 2006; Erhan *et al.*, 2010). We used an autoencoder instead of other dimensionality reduction techniques like principal component analysis because it naturally fits into the DNN architecture, and that a non-linear technique may discover a better and more compact representation of the features.

In the second stage of training, the weights from the input layer to the first hidden layer (learned from the autoencoder) are fixed, and ten additional inputs corresponding to tissues are appended. A one-hot encoding representation is used, such that specifying a tissue for a particular training example can take the form [0 1 0 0 0] to denote the second tissue out of 5 possible types. We have two such inputs totaling 10 variables that specify tissue types. The reduced feature set and tissue variables become input into the second hidden layer. The weights connected to this and the final hidden layer of the DNN are then trained together in a supervised manner, with targets

being PSI and ΔPSI. After training these final two layers, weights from all layers of the DNN were fine-tuned by backpropagation.

Each training example consists of 1393 genomic features and two tissue types as input. The targets consist of (1) PSI for each of the two tissues, and (2) ΔPSI between the two tissues. Given a particular exon and 5 possible tissue types, we constructed 5 x 5 = 25 training examples. This construction has redundancy in that we generate examples where both tissues are the same in the input to teach the model that it should predict *no change* for ΔPSI given identical tissue indices. Also, if the tissues are swapped in the input, a previously *increased* inclusion label should become *decreased* inclusion. The same rationale extends to the LMH code. Generating these additional examples is one method to incorporate this knowledge without explicitly specifying it in the model architecture.

We applied a threshold to exclude examples from training if the total number RNA-Seq junction reads is below 10. This removed 45.8% of the total number of training examples. We further define exons as having large tissue variability if ΔPSI >= +/-0.15 for at least one tissue pair profiled. These exons make up 28.6% of the total number of remaining exons that have more than 10 junction reads.

To promote the neural network to better discover the meaning of the inputs representing tissue types, we biased the distribution of training examples in the minibatches. We first selected all events which exhibit large tissue variability, and constructed minibatches based only on these events. At each training epoch, we further sampled (without replacement) training cases from the larger pool of events with low tissue variability, of size equal to one-fifth of the minibatch size. The purpose is to have a consistent backpropagation signal that updates the weights connected to the tissue inputs and bias learning towards the event with large tissue variability early on before overfitting occurs. As training progresses, the splicing pattern of the events with low tissue variability is also learned. This arrangement effectively gives the events with large tissue variability greater importance (i.e. more weight) during optimization. A side effect is that it also places more importance to the *medium* category of the LMH code during training, since they tend to be present more often in exons with tissue-specific splicing patterns.

Both the LMH and DNI codes are trained together. Since each of these two tasks might be learning at different rates, we allowed their learning rates to differ. This is to prevent one task

from overfitting too soon and negatively affecting the performance of another task before the complete model is fully trained (Silver and Mercer, 1996). This is implemented by having different learning rates for the weights between the connections of the last hidden layer and the softmax functions for each task.

The performance of the model was assessed using the area under the ROC curve (AUC) metric. To evaluate the PSI predictions for the LMH code, we used the 1 vs. all formulation. This produces three AUCs (AUC_{Low}, AUC_{Med}, AUC_{High}), one for each class. For ΔPSI predictions, since the *no change* class is much more abundant, we find that the multi-class 1 vs. all formulation tends to overestimate the tissue-specificity performance of the model due to class skew (Fawcett, 2006). Furthermore, the model can predict, based on the genomic features alone, that there is tissue-specific splicing for a given exon (which is biologically meaningful), but not necessarily how different tissues change the splicing pattern. We therefore provide two metrics to evaluate the DNI code. The first is to compute the AUC_{DvI} based on the *decrease* vs. *increase* class between two tissues. The second is to compute AUC_{Change} by comparing *no change* vs. the other two classes.

To train and test the DNN, data was split into approximately five equal folds at random for cross-validation. Each fold contains a unique set of exons that are not found in any of the other folds. Three of the folds were used for training, one used for validation, and one held out for testing. We trained for a fixed number of epochs (1500, see below) and selected the hyperparameters that give the optimal AUC performance on the validation data. The model was then re-trained using these selected hyperparameters with both the training and validation data. Five models were trained this way from the different folds of data. Predictions from all five models on their corresponding test set were used to evaluate the code's performance. To estimate the confidence intervals, the data was randomly partitioned five times, and the above training procedure was repeated.

The DNN weights were initialized with small random values sampled from a zero-mean Gaussian distribution and a variance selected via hyperparameter optimization (see Table 2-2). Learning was performed with stochastic gradient descent with momentum and dropout, where minibatches were constructed as described above. A small L1 weight penalty (see Table 2-2) was included in the cost function (Tibshirani, 1994). The model's weights were updated after each

minibatch. The learning rate ε was decreased with epochs e, and also included a momentum term μ that starts out at 0.5, increasing to 0.99, and then stays fixed. The momentum term accelerates learning and stabilizes learning near the end of training when the momentum is high by distributing gradient information over many updates. The weights of the model parameters θ were updated as:

$$\theta_e = \theta_{e-1} + \Delta\theta_e$$

$$\Delta\theta_e = \mu_e \Delta\theta_{e-1} - (1 - \mu_e)\varepsilon_e \nabla E(\theta_e)$$

where $\nabla E(\theta)$ is the gradient of the cost function with respect to the model parameters.

We used a dropout rate of 50% for all layers except for the input layer (the autoencoder), where we did not use dropout, as it empirically decreased the model's predictive performance. Training was carried out for 1500 epochs for both the pretraining with the autoencoder and supervised learning.

The performance of a DNN depends on a good set of hyperparameters. Instead of doing a grid search over the hyperparameter space, we used a Bayesian framework called *spearmint* to automatically select the model's hyperparameters that optimize the model's performance on validation data (Snoek *et al.*, 2012). The method uses a Gaussian Process to search for a joint setting of hyperparameters that optimizes an algorithm's performance on validation data. It uses the performance measures from previous experiments to decide which hyperparameters to try next, taking into account the trade-off between exploration and exploitation. This method eliminates many of the human judgments involved with hyperparameter optimization and reduces the time required to find such hyperparameters. The algorithm requires only the search range of hyperparameter values to be specified, as well as how long to run the optimization for. We used the expected improvement criterion in the optimization, as it does not require its own tuning parameter, unlike other methods in the framework. We score each experiment by the sum of the AUCs from both the LMH and DNI codes, requiring the set of hyperparameters to perform well on both tasks.

Data were split into 5 approximately equal folds at random for cross-validation. Each fold contained a unique set of exons that are not found in any of the other folds. Three of the folds were used for training, one used for validation, whose performance was used for hyperparameter

selection, and one for testing. For each fold of the data partition, a separate hyperparameter optimization procedure was performed to ensure a set of test data is always held out from the optimization. The model performance of each selection of hyperparameter was scored by the sum of the AUCs from both the LMH and DNI codes on validation data, and therefore required the setting to perform well on both tasks. The optimal set of hyperparameters were then used to re-train the model using both training and validation data. Five models were trained this way from the different folds of data. Predictions made for the corresponding test data from all models were then evaluated and reported.

The hyperparameters that were optimized and their search ranges are: (1) the learning rate for each of the two tasks (0.1 to 2.0), (2) the number of hidden units in each layer (30 to 9000), (3) the L1 penalty (0.0 to 0.25), (4) the standard deviation of the normal distribution used to initialize the weights (0.001 to 0.200), (5) the momentum schedule defined as the number of epochs to linearly increase the momentum from 0.50 to 0.99 (50 to 1500), and (6) the minibatch size (500 to 8500). The number of training epoch was fixed to 1500. In our experience, a good set of hyperparameters were generally found in approximately 2 days, where experiments were run on a single GPU (Nvidia GTX Titan). The selected set of hyperparameters are shown in Table 2-2. There is a large range of acceptable values for the number of hidden units in the second layer.

Table 2-2. The hyperparameters selected to train the deep neural network. Some are listed in ranges to reflect the variations from the different folds as well as hyperparameters from the top performing runs within a given fold.

	Range Selected
Hidden Units (layer 1)	450 - 650
Hidden Units (layer 2)	4500 - 6000
Hidden Units (layer 3)	400 - 600
L1 Regularization	0 - 0.05
Learning Rate (LMH code)	1.40 - 1.50
Learning Rate (DNI code)	1.80 - 2.00
Momentum Rate	1250
Minibatch Size	1500
Weight Initialization	0.05 - 0.09

The DNN was implemented in Python, making use of *Gnumpy* for GPU-accelerated computation (Tieleman, 2010). The GPU used was a Nvidia GTX Titan. For the configuration with the optimal hyperparameters, the GPU provided ~15-fold speedup over our original CPU implementation. This was crucial as otherwise hyperparameter optimization would not have been practical.

We compared the splicing code's performance trained with the DNN with the BNN, as well as a MLR classifier as a baseline. The MLR was trained by removing the hidden layer while keeping the training methodology identical to the neural networks. Since the predictions of the BNN consist only of the PSI prediction for each tissue separately at the output (Xiong *et al.*, 2011), in order for the BNN to make tissue difference predictions in the same manner as the DNI code, we used a MLR on the predicted outputs for each tissue pair. For a fair comparison, we similarly trained a MLR on the LMH outputs of the DNN to make DNI predictions, and report that result separately. In both cases, the inputs to the MLR are the *low*, *medium*, *high* predictions for two tissues as well as their logarithm. Schematic of the BNN and MLR architecture is shown in Figure 2-2 and 2-3.

2.3 Results and Discussion

We present three sets of results that compare the test performance of the BNN, DNN, and MLR for splicing pattern prediction. The first is the PSI prediction from the LMH code tested on all exons. The second is the PSI prediction evaluated only on targets where there are large variations across tissues for a given exon. These are events where ΔPSI >= +/- 0.15 for at least one pair of tissues, to evaluate the tissue-specificity of the model. The third result shows how well the code can classify ΔPSI between the five tissue types. Hyperparameter tuning was used in all methods. The averaged predictions from all partitions and folds are used to evaluate the model's performance on their corresponding test dataset. Similar to training, we tested on exons and tissues that have at least 10 junction reads.

For the LMH code, since the same prediction target can be generated by different input configurations, and there are two LMH outputs, we compute the predictions for all input combinations containing the particular tissue, and average them into a single prediction for testing. To assess the stability of the LMH predictions, we calculated the percentage of instances in which there is a prediction from one tissue input configuration that doesn't agree with another tissue input configuration in terms of class membership, for all exons and tissues. Of all predictions, 91.0% agreed with each other, 4.2% have predictions that are in adjacent classes (i.e. *low* and *medium*, or *medium* and *high*), and 4.8% otherwise. Of those predictions that agreed with each other, 85.9% correspond to the correct class label on test data, 51.2% for the predictions with adjacent classes, and 53.8% for the remaining predictions. This information can be used to assess the confidence

of the predicted class labels. Note that predictions spanning adjacent classes may be indicative that the PSI value is somewhere between the two classes, and the above analysis using hard class labels can underestimate the confidence of the model.

2.3.1 Performance Comparison

Table 2-3(a) reports AUC_{LMH_All} for PSI predictions from the LMH code on all tissues and exons. The performance of the DNN in the *low* and *high* categories are comparable to the BNN, but excels at the *medium* category, with especially large gains in brain, heart, and kidney. Since a large portion of the exons exhibit low tissue variability, evaluating the performance of the model on all exons may mask the performance gain of the DNN. This assumes that exons with high tissue variability are more difficult to predict, where a computational model must learn how AS interprets genomic features differently in different cellular environments. To more carefully see the tissue-specificity of the different methods, Table 2-3(b) reports AUC_{LMH_TV} evaluated on the subset of events that exhibit large tissue variability. Here, the DNN significantly outperforms the BNN in all categories and tissues. The improvement in tissue-specificity is evident from the large gains in the *medium* category, where exons are more likely to have large tissue variability. In both comparisons, the MLR performed poorly compared to both the BNN and DNN.

Table 2-3. Comparison of the LMH code's AUC performance on different methods. (a) Tested on all exons (AUC_{LMH_All}). (b) Tested on the subset of exons with large tissue variability (AUC_{LMH_TV}).

(a)

		Low	Medium	High
Brain	MLR	0.813±0.001	0.724±0.003	0.815±0.001
	BNN	**0.892±0.004**	0.752±0.003	**0.880±0.004**
	DNN	**0.893±0.005**	**0.794±0.009**	**0.883±0.006**
Heart	MLR	0.846±0.001	0.731±0.003	0.836±0.001
	BNN	**0.911±0.003**	0.747±0.003	**0.895±0.002**
	DNN	**0.907±0.006**	**0.797±0.012**	**0.894±0.011**
Kidney	MLR	0.867±0.001	0.756±0.002	0.863±0.001
	BNN	**0.925±0.004**	0.783±0.004	**0.916±0.004**
	DNN	**0.919±0.006**	**0.826±0.011**	**0.912±0.009**
Liver	MLR	0.865±0.002	0.756±0.002	0.865±0.001
	BNN	**0.927±0.003**	0.779±0.006	**0.923±0.005**
	DNN	**0.922±0.005**	**0.805±0.010**	**0.911±0.008**
Testis	MLR	0.856±0.001	0.723±0.004	0.852±0.001
	BNN	**0.911±0.003**	**0.755±0.006**	**0.904±0.003**
	DNN	**0.907±0.006**	**0.766±0.007**	**0.897±0.007**

(b)

		Low	Medium	High
Brain	MLR	0.711±0.002	0.588±0.002	0.708±0.001
	BNN	0.779±0.005	0.611±0.005	0.765±0.007
	DNN	**0.828±0.010**	**0.695±0.011**	**0.81.1±0.004**
Heart	MLR	0.739±0.003	0.586±0.004	0.727±0.001
	BNN	0.781±0.003	0.589±0.003	0.757±0.003
	DNN	**0.820±0.011**	**0.674±0.013**	**0.797±0.012**
Kidney	MLR	0.797±0.003	0.643±0.002	0.794±0.002
	BNN	0.839±0.005	0.664±0.005	0.833±0.006
	DNN	**0.862±0.006**	**0.732±0.013**	**0.853±0.012**
Liver	MLR	0.801±0.005	0.637±0.003	0.794±0.003
	BNN	0.849±0.007	0.654±0.007	0.844±0.007
	DNN	**0.877±0.006**	**0.694±0.012**	**0.848±0.008**
Testis	MLR	0.773±0.002	0.608±0.003	0.770±0.001
	BNN	0.811±0.005	0.639±0.009	0.810±0.005
	DNN	**0.846±0.011**	**0.678±0.009**	**0.835±0.009**

Next, we look at how well the different methods can predict ΔPSI between two tissues, where it must determine the direction of change. As described above, ΔPSI predictions for the BNN was made by training a MLR classifier on the LMH outputs (BNN-MLR). To make the

comparison fair, we included the performance of the DNN in making ΔPSI predictions by also using a MLR classifier (DNN-MLR) on the LMH outputs. Finally, we evaluated the ΔPSI predictions directly from the DNI code, as well as the MLR baseline method, where the inputs include the tissue types.

Table 2-4(a) shows the AUC_{DvI} for classifying *decrease* vs. *increase* inclusion for all pairs of tissue. Both the DNN-MLR and DNN outperform the BNN-MLR by a good margin. Comparing the DNN with DNN-MLR, the DNN shows some gain in differentiating brain and heart AS patterns from other tissues. The performance of differentiating the remaining tissues (kidney, liver, and testis) with each other is similar between the DNN and DNN-MLR. We note that the similarity between the DNN and DNN-MLR in terms of performance can be due to the use of soft labels for training. Using MLR directly on the genomic features and tissue types performs rather poorly, where predictions are no better than random.

Table 2-4. Comparison of the DNI code's performance on different methods. (a) AUC for decrease vs. increase (AUC_{DvI}) (b) AUC for change vs. no change (AUC_{Change})

(a)

	Brain vs. Heart	Brain vs. Kidney	Brain vs. Liver	Brain vs. Testis	Heart vs. Kidney	Heart vs. Liver	Heart vs. Testis	Kidney vs. Liver	Kidney vs. Testis	Liver vs. Testis
MLR	0.503 ±0.002	0.488 ±0.008	0.483 ±0.011	0.512 ±0.005	0.500 ±0.015	0.478 ±0.017	0.511 ±0.005	0.494 ±0.008	0.519 ±0.005	0.513 ±0.006
BNN-MLR	0.653 ±0.003	0.737 ±0.002	0.691 ±0.004	0.729 ±0.005	0.726 ±0.003	0.667 ±0.004	0.683 ±0.007	0.547 ±0.006	0.650 ±0.008	0.650 ±0.009
DNN-MLR	0.779 ±0.001	**0.830 ±0.001**	0.816 ±0.001	**0.823 ±0.002**	0.824 ±0.001	0.813 ±0.001	0.824 ±0.001	**0.768 ±0.005**	0.799 ±0.002	0.791 ±0.001
DNN	**0.794 ±0.007**	**0.833 ±0.008**	**0.825 ±0.006**	**0.829 ±0.007**	**0.861 ±0.010**	**0.851 ±0.011**	**0.848 ±0.008**	**0.762 ±0.010**	**0.825 ±0.010**	**0.818 ±0.013**

(b)

	Change vs. No Change
MLR	0.747 ±0.001
BNN-MLR	0.766 ±0.008
DNN-MLR	0.799 ±0.008
DNN	**0.865 ±0.010**

The models are further evaluated on predicting whether or not there is a difference in splicing patterns for all tissues, without specifying the direction. AUC_{Change} is computed on all exons and tissue pairs. This is shown in Table 2-4(b). The results indicate that this is a less demanding task since the models can potentially utilize just the genomic features to determine

whether an exon will have tissue variability. The difference in performance between all methods is less compared to AUC_{DvI}. However, since the evaluation is over all pairs of tissues, the DNN, which has access to the tissue types in the input, does significantly better. Although this is also true for the MLR, it still performed worst overall. This suggests that in the proposed architecture where tissues types are given as an input, the MLR lacks the capacity to learn a representation that can jointly utilize tissue types and genomic features to make predictions that are tissue-specific. Both results from Table 2-4 show that there is an advantage to learning a DNI code rather than just learning the LMH code.

To test whether the predictions generalize to RNA-Seq data from a different experiment, we selected data for two mouse tissues, namely the brain and the heart, from (Barbosa-Morais *et al.*, 2012), and analyzed how our model, which is trained with data from (Brawand *et al.*, 2011), performs. Table 2-5 shows the set of evaluations on the DNN identical to that of Tables 2-3 and 2-4, tested on this RNA-Seq data. For the brain, there is an ~1-4% decrease in AUC_{LMH_All} and ~4-5% decrease for AUC_{LMH_TV}. For the heart, the model's performance on both datasets is equivalent to within one standard deviation for both AUC_{LMH_All} and AUC_{LMH_TV}. A decrease in performance of ~7% is observed in AUC_{DvI} for brain vs. heart. There is an increase in AUC_{Change} but that is due to only two tissues being evaluated as opposed to five, where the AUC would be pulled down by the other tissues with lower performances if they were present.

Table 2-5. Performance of the DNN evaluated on a different RNA-Seq experiment. (a) AUC_{LMH_All} (b) AUC_{LMH_TV} (c) AUC_{DvI} (d) AUC_{Change}

(a)

	Low	Medium	High
Brain	0.881±0.005	0.761±0.010	0.870±0.006
Heart	0.907±0.005	0.784±0.013	0.890±0.010

(b)

	Low	Medium	High
Brain	0.791±0.009	0.661±0.010	0.776±0.008
Heart	0.826±0.010	0.653±0.012	0.788±0.011

(c)

	Brain vs. Heart
DNN-MLR	0.729±0.001
DNN	0.742±0.015

(d)

	Change vs. No Change
DNN-MLR	0.817±0.010
DNN	0.919±0.007

Overall, the decrease in performance is not unexpected, due to differences in PSI estimates from variations in the experimental setup. To see how PSI differed, we computed the expected PSI values for brain and heart in all exons from both sets of experiments and evaluated their

Pearson correlation. For the brain, the correlation is 0.945, and for the heart, it is 0.974. This can explain why there is a larger decrease in performance for the brain, which is a particularly heterogeneous tissue, and hence can vary more between experiments depending on how the samples were prepared. We note that the performance of the DNN on this dataset is still better than the BNN's predictions on the original dataset. Viewed as a whole, the results indicate that our model can indeed be useful for splicing pattern predictions for PSI estimates computed from other datasets. It also shows that our RNA-Seq processing pipeline is consistent.

We believe there are several reasons why the proposed DNN has improved predictive performance in terms of tissue-specificity compared to the previous BNN splicing code. One of the main novelties is the use of tissue types as an input feature, which stringently required the model's hidden representations be in a form that can be well-modulated by information specifying the different tissue types for splicing pattern prediction. This requirement would be relaxed if each tissue was trained separately. Furthermore, this hidden representation is described by thousands of hidden units and multiple layers of non-linearity. In contrast, the BNN only has 30 hidden units to represent the variables that can be used by the model to modulate splicing based on the cellular condition, which may not be sufficient. Another difference is that for the DNI code, a training objective was specifically designed to learn to predict ΔPSI, which is absent from the BNN. However, the performance gain of the DNN-MLR over BNN-MLR shows that this is only part of the improvement.

In addition, we performed hyperparameter search to optimize the DNN, where we gained considerable improvements over our original hand-tuned models, at ~4.5% for the DNI code and ~3.5% for the LMH code. Interestingly, the final set of hyperparameters found opts for a much larger (~4x) number of hidden units than our initial attempt (with matching hyperparameters). Manually trying to find these hyperparameters would have been difficult, where a user may settle for a sub-optimal set of hyperparameters due to the substantial effort and time required for training a model with millions of parameters.

Another performance boost came from the use of dropout, which contributed ~1 to 6% improvement in the LMH code for different tissues, and ~2 to 7% in the DNI code, compared to without. The performance difference would likely be larger if hyperparameter optimization was not performed on the model that did not use dropout. We note also that even with dropout, a small

L1 weight penalty was found to be beneficial, which may explain our model's tolerance for a large number of hidden units with very sparse weights.

One additional difference compared with previous work is that training was biased towards the tissue-specific events (by construction of the minibatches), thereby promoting the model to learn a good representation about cellular context. We were able to get some small performance gains (within 2 standard deviations) of ~1 to 2% in AUC_{LMH_TV} and AUC_{DVI} using this methodology compared to training with all events treated equally. More importantly, biasing the training examples encourages the model to learn about the tissues as input, which has a significantly different meaning compared to the genomic features and make up only a small number of the input dimension. We find that without this learning bias, the model more frequently settles to a bad local minimum, or doesn't learn to utilize the tissues as input at all. Together, all these changes allowed us to train a model that significantly improves upon previous work.

With regards to training the two tasks jointly, we found that with hyperparameter tuning, the performance of the model when each task was trained separately compared to being trained together was not statistically different. This is likely because both tasks are too similar for any transfer learning to take place, as evident by the similarity in performance in the DNI code between the DNN and DNN-MLR models. Nevertheless, we find that training both codes together stabilizes learning, specifically, training becomes more tolerant to a larger range of hyperparameters leading to reduced variance between models.

2.3.2 Model and Feature Analysis

A major contribution to the success of splicing pattern predictions that generalize well comes from the richness of our feature set. For example, we observed a significant decrease in the performance of the splicing code if the reduced feature vector dimension is too small by either principal component analysis or an autoencoder with a small number of hidden units. We found that the performance of both the LMH code and the DNI code drops by up to 4% when the reduced dimension is at 150 (down from 1393). This suggests a sufficiently large number of hidden variables denoting genomic features are required to interact with tissue inputs to achieve good performance.

It can be useful to see how the genomic features are used by the DNN to perform splicing pattern predictions. We analyzed our model in two different ways.

In the first method, in order to see which features types are important to the model, we substituted genomic features to their median across all exons and looked at how the predictive performance changed. We divided the full feature set into 55 groups based on what they represent. The grouping, along with additional descriptions, can be found in Appendix A. Here, the performance measure is defined as the sum of the three classes from AUC_{LMH_All}. The decrease in test performance (as a fraction of that obtained with the full feature set) when each group of features is substituted by their median is shown in Figure 2-4.

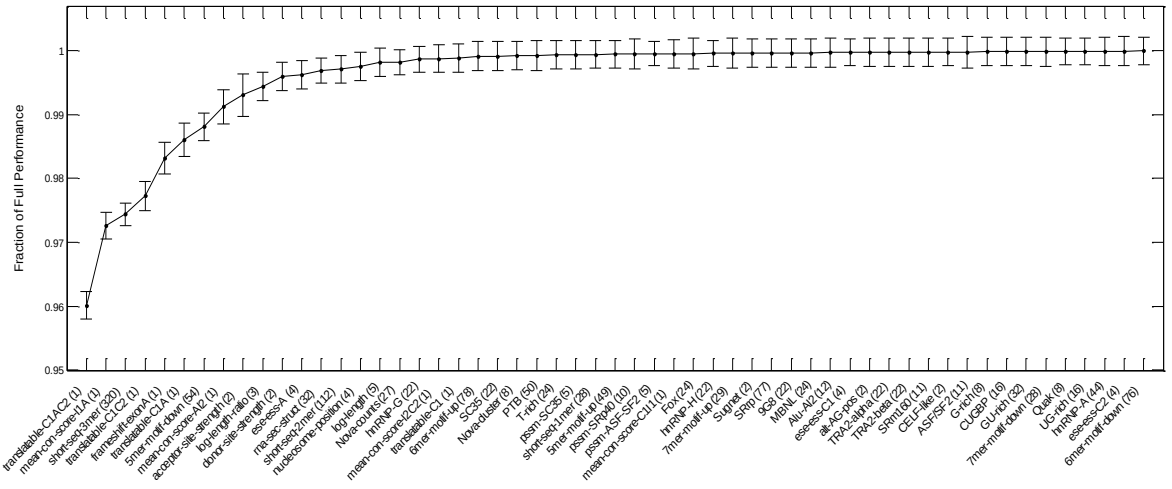


Figure 2-4. Plot of the change in AUC_{LMH_All} performance by substituting the values in each feature groups by their median. Feature groups that are more informative to the predictive performance of the model have lower values. The groups are sorted by the mean over multiple partitions and folds, with the standard deviations shown. The number of features for each feature group is indicated in brackets.

Feature groups that cause a large decrease in performance are presumably more informative for splicing pattern predictions for the current mouse alternative exon dataset. The standard deviation is computed from the five trained models with random partitions of the data as described above. The order of the feature group towards the right of the plot should not be used to determine their order of importance due to the small difference they make to the model relative to their standard deviations. It is interesting to see how small the decrease in AUC is when each feature group is effectively removed. Many features contain redundant information, and therefore can compensate for missing features from other groups. For example, some of the motifs for

splicing factors are represented in features representing *n*-mer counts. The most influential features describe the translatability of the exon, conservation scores, and whether the alternative exon introduces a frameshift. The feature groups corresponding to counts of 3-mers and 5-mers are also important.

To examine how each individual feature affects the DNN's predictions, we adapted the method from (Simonyan *et al.*, 2014). Briefly, examples from the dataset are given as input to the trained model and forward propagated through the neural network. At the output, the target is modified to a different value, for example, in classification, by changing the class label. The error signal is then backpropagated to the inputs. The resulting signal describes how much each input feature needs to change in order to make the modified prediction, as well as the direction. The computation is extremely quick, as it only requires a single forward and backward pass through the DNN, and all examples can be calculated in parallel. We used this procedure on exons with low tissue variability, and modified the *low* PSI targets to *high*, and the *high* PSI targets to *low*. Table 2-6 lists the top 25 features with the largest backpropagated signal magnitude (which indicate that these features need to change the least to affect the prediction the most, and are hence important; note also that all of our features are normalized). The table also indicates general trends in the direction of change for each feature over the dataset. If more than 5% of the examples do not follow the general direction of change, it is indicated by both an up and down arrow. Some of the splicing rules inferred by the model can be seen. For example, the presence of splicing silencers inhibits the splicing of the alternative exon leading to higher inclusion, a shorter alternative exon is more likely to be spliced out, and the strength and position of acceptor and donor sites can lead to different splicing patterns.

Next, we wanted to see how features are used in a tissue-specific manner. Using the set of exons with high tissue variability, we computed the backpropagation signal to the inputs with the output targets changed in the same manner as above, for each tissue separately. Figure 2-5 shows the sum of the magnitudes of the gradient, normalized by the number of examples in each tissue for the top 50 features. We can observe that the sensitivity of each feature to the model's predictions differs between tissues. The profile for kidney and liver tend to be more similar to each other than others, which associates well with the model's weaker performance in differentiating these two tissues. This figure also provides a view of how genomic features are differentially utilized by the DNN, modulated by the input tissue types. In both Table 2-6 and

Figure 2-5, the backpropagation signals were computed on examples from the test set, for all five partitions and folds.

Table 2-6. The top 25 features (unordered) of the splicing code that describes low and high percent inclusion. The direction in which each feature can increase (↑) or decrease (↓) the PSI predictions is shown. Features whose effect on PSI is context-dependent (i.e. they depend on other features) is indicated with (↑↓). Feature details can be found in Appendix A.

Feature Description*	Direction
strength of the I1 acceptor site	↑
strength of the I2 donor site	↑
strength of the I1 donor site	↓
mean conservation score of first 100 bases in 3' end of I1	↑↓
mean conservation score of first 100 bases in 5' end of I2	↑↓
counts of Burge's exonic splicing silencer in A	↑
counts of Chasin's exonic splicing silencer in A	↑
log base 10 length of exon A	↑
log base 10 length ratio between A and I2	↑
whether exon A introduces frameshift	↑↓
predicted nucleosome positioning in 3' end of A	↑↓
frequency of AGG in exon A	↓
frequency of CAA in exon A	↑
frequency of CGA in exon A	↑
frequency of TAG in exon A	↓
frequency of TCG in exon A	↑
frequency of TTA in exon A	↓
translatability of C1-A	↑
translatability of C1-A-C2	↑
translatability of C1-C2	↓
counts of Yeo's 'GTAAC' motif cluster in 5' end of I2	↑
counts of Yeo's 'TGAGT' motif cluster in 5' end of I2	↑
counts of Yeo's 'GTAGG' motif cluster in 5' end of I2	↑
counts of Yeo's 'GTGAG' motif cluster in 5' end of I2	↑
counts of Yeo's 'GTAAG' motif cluster in 5' end of I2	↑

**C1* and *C2* denote the flanking constitutive exons; *A* denotes the alternative exon; *I1* denotes the intron between *C1* and *A*; *I2* denotes the intron between *A* and *C2*

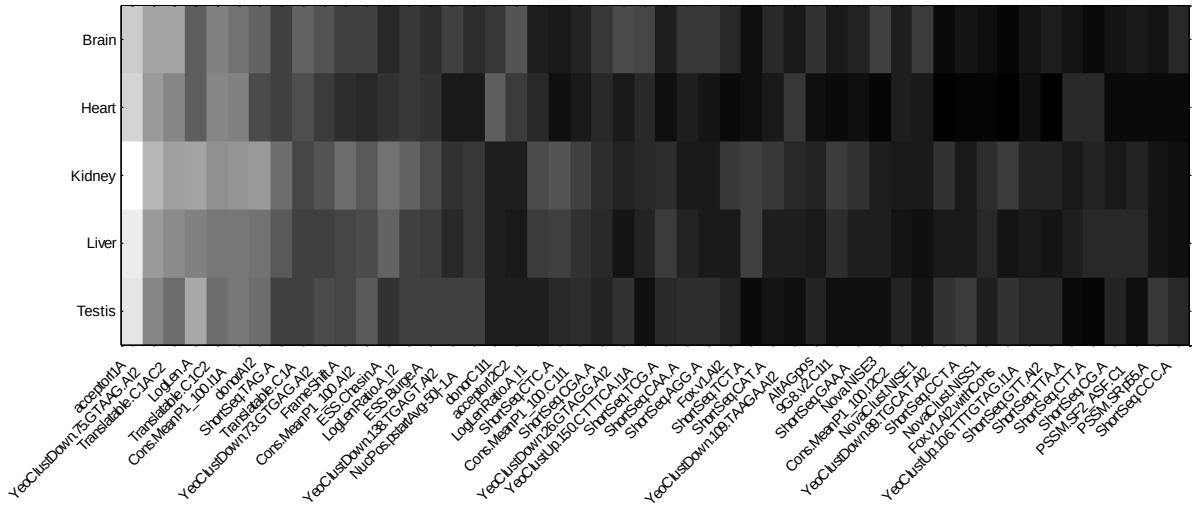


Figure 2-5. Magnitude of the backpropagated signal to the input of the top 50 features computed when the targets are changed from low to high, and high to low. White indicates that the magnitude of the signal is large, meaning that small perturbations to this input can cause large changes to the model's predictions. The features are sorted left to right by the magnitude across all tissue types.

2.4 Conclusion

In this chapter, we introduced a computational model that extends the previous splicing code with new prediction targets and improved tissue-specificity, using a learning algorithm that scales well with the volume of data and the number of hidden variables. The approach is based on deep neural networks, which can be trained rapidly with the aid of graphics processing units, thereby allowing the models to have a large set of parameters and deal with complex relationships present in the data. We demonstrate that deep architectures can be beneficial even with a sparse biological dataset. We further described how the input features can be analyzed in terms of the predictions of the model to gain some insights into the inferred tissue-regulated splicing code.

Our architecture can easily be extended to the case of more data from different sources. For example, utilizing the same architecture, we may be able to learn a hidden representation that spans additional tissue types as well as multiple species. Through transfer learning, training such a model with multiple related targets might be beneficial particularly if the number of training examples in certain species or tissues is small.

Chapter 3
Inference and Analysis of a Human Polyadenylation Code

Processing of transcripts at the 3'-end involves cleavage at a polyadenylation site followed by the addition of a poly(A)-tail. By selecting which site is cleaved, the process of alternative polyadenylation enables genes to produce transcript isoforms with different 3'-ends. To facilitate the identification and treatment of disease-causing mutations that affect polyadenylation and to understand the sequence determinants underlying this regulatory process, a predictor that can accurately infer polyadenylation patterns from genomic features is desirable. In this chapter, the development of a computational model to infer alternative polyadenylation patterns from human RNA-Seq data is described. The model is trained to predict which polyadenylation site is more likely to be selected in genes with multiple sites. Using the same model, we show that the predictor can also be used to scan the 3' untranslated region to find candidate polyadenylation sites. To illustrate its potential for genomic medicine, we furthermore demonstrate how it can be used to classify the pathogenicity of variants near annotated polyadenylation sites in ClinVar, and to anticipate the effect of an antisense oligonucleotide experiment to redirect polyadenylation. The content of this chapter is based on the publication (Leung *et al.*, 2018):

M. Leung, A. Delong, and B. Frey. (2018) "Inference of the Human Polyadenylation Code". Bioinformatics, 34(17), 2889-2898.

3 The Human Polyadenylation Code

3.1 Introduction

Polyadenylation is a pervasive mechanism responsible for regulating mRNA function, stability, localization, and translation efficiency. As much as 70% of human genes are subject to alternative polyadenylation (APA) and wide-spread mechanisms have been found which influence its regulation (Elkon *et al.*, 2013). By selecting which polyadenylation site (PAS) is cleaved, different transcript isoforms that vary either in their coding sequences or in their 3' untranslated region (3'-UTR) can be produced. Transcripts differentially cleaved can influence how they are regulated. For example, longer variants can harbor additional destabilization elements that alter a transcript's stability (Shaw and Kamen, 1986), and shortened variants can escape regulation from microRNAs, which have been observed in various cancers (Lin *et al.*, 2012; Di Giammartino *et al.*, 2011).

Furthermore, APA can be tissue-dependent, so a single gene can generate different transcripts, for instance, based on the tissue in which it is expressed (Tian and Manley, 2016). One mechanism of APA regulation occurs at the level of the sequences of the transcript. The presence or absence of certain regulatory elements can influence which PAS is selected. PAS selection is also influenced by a site's position relative to other sites. A computational model that can accurately predict how polyadenylation is affected by genomic features as well as cellular context is highly desirable to understand this widespread phenomenon. Moreover, several inherited diseases have been linked to errors in 3'-end processing (Danckwardt *et al.*, 2008). Such model would enable the exploration of the effects of genetic variations on polyadenylation and their implications for disease.

In this chapter, we present the polyadenylation code, a computational model that can predict alternative polyadenylation patterns from transcript sequences. While there have been previous works in classifying whether a stretch of sequence contains a PAS (Cheng *et al.*, 2006; Akhtar *et al.*, 2010; Chang *et al.*, 2011; Kalkatawi *et al.*, 2012; Xie *et al.*, 2013; Ho *et al.*, 2013), or characterizing whether a PAS is tissue-specific (Hafez *et al.*, 2013; Weng *et al.*, 2016), many of them are aimed at improving gene annotations and understanding which features are involved in APA regulation, and do not address the question of predicting how APA sites are variably selected. Here, we tackle this question by developing a model that can predict a score, which we refer to as PAS strength (Shi, 2012), that describes the efficiency in which a PAS is recognized by 3'-end processing machinery for cleavage and polyadenylation. The ability to predict PAS strength enables this model to generalize to multiple prediction tasks, even though it is not explicitly trained for them. For example, the model can be applied to a gene with multiple PAS to determine the relative transcript isoforms that would be produced, in a tissue-specific manner. The model can predict the consequence of nucleotide substitutions on PAS strength, which can be used to prioritize genetic variants that affect polyadenylation. It can be used to assess the effects of antisense oligonucleotides to alter transcript abundance. It can also scan the 3'-UTR of the human genome to find potential PAS. We demonstrate examples of these applications and provide analysis on how different features affect the predictions of the model.

3.2 Methods

3.2.1 Inferring the Strength of a Polyadenylation Site

The goal of this work is to infer a score that describes the strength of a PAS, or the efficiency in which it is recognized by the 3'-end processing machinery. The problem would be straightforward if this target variable is directly measurable. However, current sequencing protocols only provide a measurement of the relative transcript abundance from APA. Various approaches exist in the literature which attempt to quantify the strength of a PAS. For example, normalized read counts are often used, but quantification can be affected by factors such as sequencing biases, transcript length, and RNA decay (Oshlack and Wakefield, 2009; Gallego Romero *et al.*, 2014). Some studies classify PAS strength based on whether a canonical polyadenylation signal or other known sequence elements are present near the PAS (Akhtar *et al.*, 2010). We believe a more principled approach to predict a quantitative description of the strength of a PAS is to model it as a hidden variable, and infer it from data. Moreover, the position of a PAS relative to neighboring sites affects its selection. Some biological processes and tissues tend to favor PAS at the distal end, whereas cells under disease states tend to utilize PAS that are more proximal (Elkon *et al.*, 2013). Therefore, the model should include a variable that accounts for the distance between neighboring sites during training. Even though the position of a PAS is modeled, a desirable characteristic of the predictor is that during inference, positional information should be optional. This can be useful in regions of the genome where there are insufficient annotation sources to ascertain the distance to a nearby PAS. This would also enable one to apply this model to any DNA sequence associated with a site, optionally modify the bases within, and see the predicted effect on polyadenylation regulation. To determine which PAS in a gene with multiple sites is more likely to be selected, the model can be applied to each PAS separately to compare their relative strengths. Optionally, their positions can be factored into the model's prediction if annotation sources are available in order to get a better estimate.

3.2.2 The Polyadenylation Code

The polyadenylation code is a model that can infer tissue-specific PAS strength scores from sequence, and optionally account for the influence of position if it is provided. It takes as input a sequence of length 200 bases centered on a PAS. We benchmark two models which operate on the sequence differently.

The first model is built on hand-crafted features derived from the literature of sequence elements associated with polyadenylation (Appendix B). The genomic sequence is processed by a feature extraction pipeline, which divides the sequence into 4 regions relative to the PAS (Hu *et al.*, 2005). Some features are limited to specific regions, namely the polyadenylation signals in the 5'-5' and 5'-3' regions, and hexamers defined in (Hu *et al.*, 2005). Other features are computed in all regions, including counts of RNA-binding protein (RBP) motifs that may be involved in polyadenylation, all possible 1 to 4 n-mers counts, and nucleosome positioning features from (van der Heijden *et al.*, 2012). The feature vector is mapped to a fully-connected neural network. We will refer to this model as the Feature-Net.

The second model directly learns from the genomic sequence, using a convolutional neural network (Conv-Net) architecture (Y. LeCun, Bottou, *et al.*, 1998), which can efficiently discover sequence patterns without prior knowledge even when the location of the patterns is unknown. The Conv-Net comprises of tunable motif filters which are free to adapt to the input sequence to optimize the predictive performance of the model. It also contains pooling operations that enable the model to focus on select locations in the input sequence whose composition maximally activate the motif filters. The use of convolutional neural networks to learn from raw genomic sequences have been successfully applied in other areas of biology (Alipanahi *et al.*, 2015; Zhou and Troyanskaya, 2015; Kelley *et al.*, 2016; Angermueller *et al.*, 2017).

To account for the positional preference of PAS, the log distance between sites is also an input feature for both models. Given two sites, the proximal (5') site has a position feature of 0, whereas the distal (3') site has a position feature that is equal to the logarithm of the distance between the distal and proximal site.

Figure 3-1 shows the schematic of both models. After the sequences are transformed by the Feature-Net and Conv-Net into a hidden representation, it is processed by separate fully-connected hidden layers to make tissue-specific predictions. The architecture therefore factors predictions into two components: a score that describes the tissue-specific PAS strength, followed by predictions that represent the relative abundance of transcripts from RNA-Seq experiments between two competing PAS. The parameters of the fully-connected layers model the cell state of tissues, which describes the steady-state environment of the cell, such as the protein concentrations in the cytosol, that can affect transcriptional modifications. We do not explicitly define what these

cell state parameters consist of or how they factor in the predictions, but rather simply model them as hidden variables and learn them from data. A similar approach has been described in the splicing regulatory model by Xiong et al. (Xiong *et al.*, 2015).

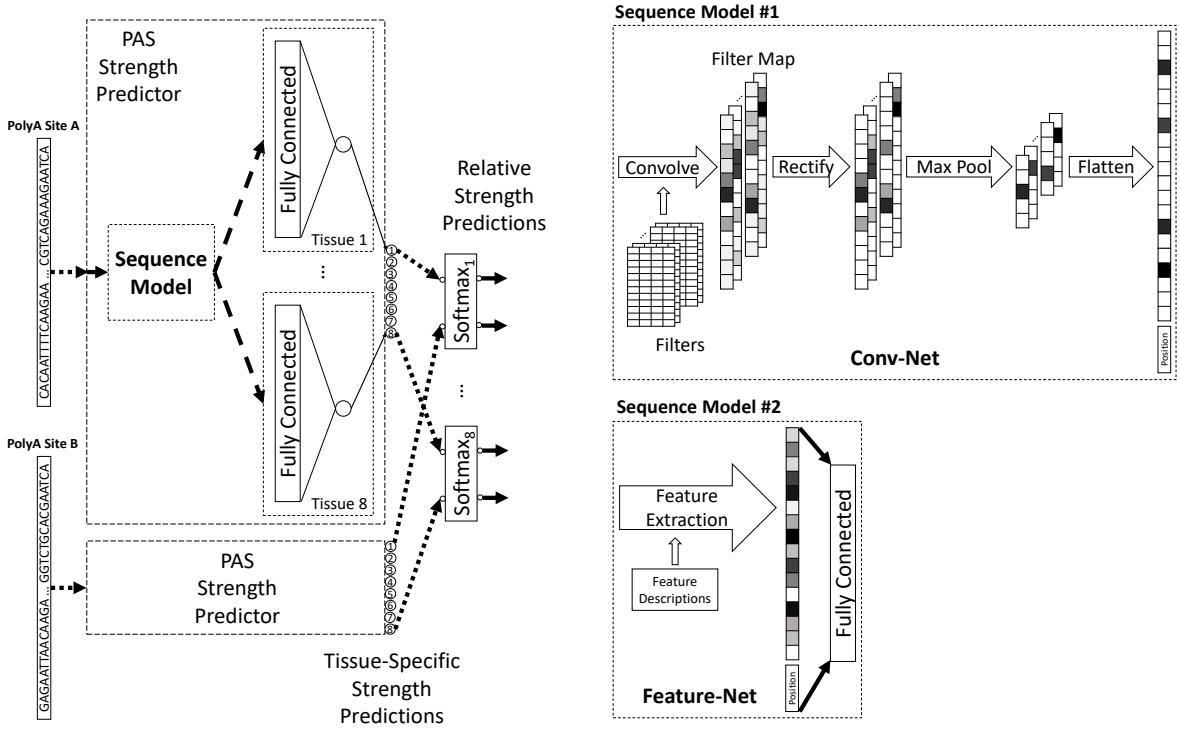


Figure 3-1. (left) A schematic of the components of the neural network that represent the polyadenylation model. The genomic sequence surrounding a polyadenylation site is an input to the strength predictor, which outputs eight tissue-specific scores describing the efficiency of the site for cleavage and polyadenylation. The model is trained from the relative strength between pairs of competing sites. (right) Two architectures are compared for the sequence model, a convolutional neural network that operates directly on sequences and a fully-connected neural network that takes in a feature vector processed by a feature extraction pipeline.

Seven distinct tissue types are available in the dataset used to train the models. Since there are two sets of sequencing reads for the naïve B-cells obtained from different donors (Lianoglou *et al.*, 2013), we treat them as separate tissues, and so our models have eight polyadenylation strength prediction outputs. We choose not to rely on evolutionary conservation to force the models to learn patterns from the genome itself (Leung *et al.*, 2016). We also do not want to make use of additional data sources such as conservation tracks or expression data as input. For our model to be widely applicable to multiple tasks, it is beneficial for the input to be easily obtainable,

such as sequences. Requiring anything beyond sequences makes a model more difficult to apply across diverse problem domains.

A training example consists of two PAS from the same gene, and requires the model to predict their relative strengths, which can be interpreted as the probability that each site would be selected for cleavage and polyadenylation. The relative strength is measured by the count of reads from RNA-Seq that have been mapped to each site. As shown in Figure 3-1, a softmax function is used to squash the real-valued predictions from the PAS strength predictor into a normalized score that can be interpreted as the probability that one PAS is chosen over the other. The predictions are penalized against training targets of the relative abundances of transcripts for these PAS, which is measured from the sequencing experiment. Most of the results presented in this work are based on the predictions from the PAS strength predictor (i.e. the logits) instead of the relative strength predictions that follow the softmax.

In this work, we apply the predictive model to multiple tasks, even though it is trained only to the task of modeling competing site selection. All the predictions for these other tasks are evaluated without any additional task-specific training or data augmentation to demonstrate the general applicability of this model.

3.2.3 Assembling the Polyadenylation Atlas

Analysis of human polyadenylation events is confined to the 3'-UTR, where PAS are most frequently located. To identify the 3'-UTR regions of the human genome, 3'-UTR annotations from UCSC (Kent *et al.*, 2002), GENCODE (Harrow *et al.*, 2012), RefSeq (Pruitt *et al.*, 2005), and Ensembl (Yates *et al.*, 2016) are combined, where overlapping regions are merged, and each 3'-UTR segment is further extended by 500 bases to capture potential uncharacterized regions. Then, to generate a comprehensive atlas of PAS, multiple polyadenylation annotations and reads from different 3'-end sequencing experiments are mapped to the 3'-UTR to generate an atlas of human PAS. The polyadenylation annotations used include PolyA_DB 2 (Lee *et al.*, 2007), GENCODE (Harrow *et al.*, 2012), and APADB (Müller *et al.*, 2014). Mapped reads that lie in the 3'-UTR from PolyA-Seq (Derti *et al.*, 2012) and 3'-Seq (Lianoglou *et al.*, 2013) are also used to expand the repertoire of PAS, where the genomic positions of reads from these sequencing experiments are used to mark the locations of PAS in the genome. PAS from different sources largely overlap, but some sites can be unique to one study due to differences in cell lines or tissue

types as well as sequencing protocol. Due to the inexact nature of 3'-end processing (Proudfoot, 2011), PAS that are within 50 bases of each other are clustered, and the resulting peak marked as the location of the PAS. The final PAS atlas contains 19,320 3'-UTR regions with two or more PAS from genes in the hg19 assembly for a total of 92,218 sites.

3.2.4 Quantifying Relative Polyadenylation Site Usage

The model is trained from the relative abundance of transcripts from a 3'-end sequencing experiment of seven distinct human tissues, including the brain, breast, embryonic stem (ES) cells, ovary, skeletal muscle, testis, and two samples of naïve B cells (Lianoglou *et al.*, 2013). Other cell lines are also available in the dataset, but they are not used. The version of aligned reads which have been processed through the study's computational pipeline is used, which include removal of internally primed and antisense reads, as well as the application of minimum expression requirements to reduce sequencing noise. These reads are assigned to our PAS atlas, resulting in read counts associated with each PAS.

To quantify the relative PAS usage for each gene which acts as the target to train the model, we adopted the Beta model derived from Bayesian inference described in (Xiong *et al.*, 2016), treating the percent read counts of one site relative to another site as the parameter of a Bernoulli distribution. With this model, the relative PAS usage of one site relative to another, referred to as Φ, is $p(\Phi) = Beta(1+N_{site1}, 1+N_{site2})$, where N_{site1} and N_{site2} are the number of reads from two different sites. We use the mean of this distribution as the target to train the model, that is, the PAS usage of site 1 relative to site 2 is $(1 + N_{site1}) / (2 + N_{site1} + N_{site2})$. For 3'-UTR regions with more than 2 PAS, different combinations of pairs of sites are generated as training targets and quantified as above. The assumption is that the relative strength of neighboring PAS can be described by the relative read counts at those sites, even if there are other sites present in the same gene. This assumption simplifies the architecture of the computational model and quantification of relative strength between sites.

3.2.5 Training the Neural Networks

The model is constructed and trained in Python using the TensorFlow library (Abadi *et al.*, 2015; Rampasek and Goldenberg, 2016). All hidden units of the neural network consist of rectified linear activation units (Glorot *et al.*, 2011). For the Feature-Net, the feature vectors are normalized with mean zero and standard deviation of one. For the Conv-Net, the input uses a one-hot encoding

representation for each of the 4 nucleotides. For a sequence of length *n*, the dimension of the input would be 4 x *n*. Padding is inserted at both ends of the input so that the motif filters can be applied to each position of the sequence from beginning to end. For a motif filter of length *m*, the additional padding on each side of the sequence would be 4 x (*m* - 1), where these additional padding would be filled with the value 0.25, equivalent to an N nucleotide in IUPAC notation. This is similar to what is done in (Alipanahi *et al.*, 2015).

Each training example consists of a pair of PAS from a gene, where the input is the two sites' genomic sequences, and the target is their relative read counts computed as described in Section 3.2.4. For genes with more than 2 PAS, different combinations of pairs of sites are generated as examples. Only examples with more than 10 reads are kept. This resulted in a dataset of 64,572 examples, which is split for training and testing.

The parameters of the neural network are initialized according to (Glorot and Bengio, 2010), and trained with stochastic gradient descent with momentum and dropout (Hinton *et al.*, 2012). Predictions from each softmax output are penalized by the cross-entropy function, and its sum across all tissue types is backpropagated to update the parameters of the neural network. Training and testing of the model are performed in a similar fashion as described in (Leung *et al.*, 2014). Briefly, data is split into approximately five equal folds at random for cross-validation. Each fold contains a unique set of genes that are not found in any of the other folds. Three of the folds are used for training, one is used for validation, and one is held out for testing. By selecting which fold is held out for testing, five models are trained. The predictions of these five models on their corresponding test set are used for performance assessment, as well as to estimate variances, for all the tasks analyzed in this work.

The validation set is used for hyperparameters selection. The selected hyperparameters for our models are shown in Table 3-1. A graphics processing unit is used to accelerate training and hyperparameter selection by randomly sampling the hyperparameter space.

Table 3-1. The following hyperparameters are determined by random sampling and selecting the set that provides the best validation performance. The range each hyperparameter is sampled from is indicated. The number of training epochs is fixed to 50.

Hyperparameter	LR	Feature-Net	Conv-Net
Mini-batch size [50 to 2500]	1777	1520	2042
Hidden units in the final fully connected layer per tissue [10 to 2000]	---	1384	119
Learning rate [0.0001 to 0.5]	0.10066	0.09537	0.35714
Initial momentum [0 to 0.99]	0.29108	0.21876	0.43301
L1 decay [1e-8 to 5e-3]	0.000087	0.000177	0.000181
Hidden units in the first hidden layer [50 to 2500]	---	1244	---
Number of filters [80 or 96]	---	---	80
Filter width [9 or 12]	---	---	12
Filter stride [fixed]	---	---	1
Pool width [fixed]	---	---	20
Pool stride [fixed]	---	---	10

3.3 Results

3.3.1 Polyadenylation Site Selection

The performance of the model to predict the likelihood that a PAS is selected for cleavage and polyadenylation against a competing site in the same gene is shown in Table 3-2. These are the tissue-specific relative strength predictions for pairs of PAS that's shown in Figure 3-1. Performance is assessed using the area under the receiver-operator characteristic (ROC) curve (AUC) metric on held-out test data. To compare the models' performance against a baseline, we also trained a logistic regression (LR) classifier, which is essentially the Feature-Net with hidden layers removed. Predictions from the model based on the Conv-Net architecture is consistently the best performer. There is sizable performance gain from using the neural network models compared to the logistic regression classifier.

For the more general task of predicting which PAS would be selected in a gene with multiple sites, the model is applied to all PAS in the 3'-UTR of each gene. A score for each site is computed from the logits (the output of the PAS strength predictor shown in Figure 3-1), where a larger value suggests that the site is more likely to be selected. The target is defined by the PAS in each gene which has the most measured reads in the 3'-Seq data. The metric we report here is the prediction accuracy, or the percentage of genes in which the model has correctly predicted the PAS that has the most reads. This is shown in Table 3-2 for genes with two to six sites, averaged

across all tissues. The number of genes used in this evaluation is 2270, 2043, 1745, 1364, and 1163, respectively, where a gene is included only if at least one of its sites has more than 10 reads.

Table 3-2. PAS selection performance between competing sites in different tissues.

Tissue Type	AUC		
	LR	Feature-Net	Conv-Net
Brain	0.826 ± 0.010	0.869 ± 0.007	0.895 ± 0.005
Breast	0.825 ± 0.006	0.862 ± 0.003	0.886 ± 0.004
ES Cells	0.849 ± 0.006	0.898 ± 0.002	0.911 ± 0.006
Ovary	0.830 ± 0.009	0.873 ± 0.006	0.895 ± 0.003
Skel. Muscle	0.828 ± 0.006	0.872 ± 0.005	0.893 ± 0.004
Testis	0.787 ± 0.007	0.828 ± 0.005	0.856 ± 0.007
B Cells 1	0.838 ± 0.005	0.880 ± 0.005	0.896 ± 0.004
B Cells 2	0.832 ± 0.004	0.880 ± 0.008	0.893 ± 0.007
All	0.824 ± 0.005	0.866 ± 0.004	0.889 ± 0.003

Table 3-3. PAS selection performance in genes with 2 to 6 sites.

Number of Sites	Accuracy (%)		
	LR	Feature-Net	Conv-Net
2	79.6	82.5	83.5
3	68.3	73.0	75.5
4	58.9	64.4	69.8
5	55.6	62.8	64.0
6	48.5	56.4	59.7

3.3.2 Pathogenicity Prediction of Polyadenylation Variants

An advantage of our model is that the PAS strength predictor can be used to characterize individual sites based only on the input sequence. We evaluate whether this model can be used for pathogenicity prediction. The basic approach involves applying the model to the 200 nucleotides sequence associated with a PAS from the reference genome to first generate a prediction of its strength, and then performing another prediction when one or more nucleotides in the sequence are altered. A difference is then computed between the reference and variant predictions. Since there are eight predictions, one for each tissue, we take the largest difference as the score to assess pathogenicity. A similar approach has been applied to splicing variants (Xiong *et al.*, 2015). The postulate is that if a variant causes a large change to the strength of a PAS, this can change the relative abundance of differentially 3'-UTR terminated transcripts that deviates from normal, potentially indicating disease associations.

To evaluate the efficacy of this approach, we extracted variants that overlap with our PAS atlas (within 100 bases on either side of an annotated PAS) from the ClinVar database (Landrum *et al.*, 2014). Some of these variants overlap with the terminal exon (e.g. missense mutations) and are removed. Table 3-4 shows the 12 variants that are labeled as pathogenic (CLNSIG=5) and 48

that are labeled as benign (CLNSIG=2) according to the classification guideline recommended by the American College of Medical Genetics and Genomics and the Association for Molecular Pathology (see Table 5 in the standard) (Richards *et al.*, 2015). The guideline refers to variants in these two categories as being sufficiently supported by empirical data such that healthcare providers can use the testing information associated with these variants for clinical decision making.

Table 3-4. Variants are given in notation *chromosome*:*position*:*reference*:*variant*, based on the hg19 assembly.

Pathogenic (CLNSIG=5):			
chr1:11082794:T:C,	chr8:22058957:T:C,	chr11:2181023:T:C,	chr11:5246715:T:C,
chr11:5246716:T:A,	chr11:5246716:T:C,	chr11:5246717:T:C,	chr11:5246718:A:G,
chr11:5246718:A:T,	chr11:46761055:G:A,	chr16:223691:A:G,	chr22:51063477:T:C
Benign (CLNSIG=2):			
chr1:156109644:G:A,	chr1:197053394:G:A,	chr2:71004492:T:C,	chr2:166847735:T:A,
chr2:166847735:T:C,	chr2:179326003:A:C,	chr2:207656535:T:C,	chr3:178952181:T:C,
chr4:141471538:C:T,	chr4:187131799:T:C,	chr5:112180071:A:G,	chr5:118877695:A:G,
chr6:7586120:T:A,	chr6:116953612:A:G,	chr6:158532382:T:C,	chr10:27035405:A:G,
chr11:74168280:G:A,	chr11:77811990:T:C,	chr12:64202890:C:G,	chr16:15797843:G:C,
chr18:48604848:C:T,	chr18:52895244:C:T,	chr19:1226654:C:T,	chr19:1395497:C:T,
chr19:1395500:C:A,	chr19:1395500:C:T,	chr19:1395503:C:T,	chr19:4090577:G:A,
chr19:4090588:G:A,	chr19:36494234:A:G,	chr19:36595935:G:A,	chr19:50364490:G:A,
chr22:29083867:G:A,	chr22:50964189:C:T,	chr22:50964196:G:A,	chr22:50964196:G:T,
chrX:135126891:A:T,	chrX:153287318:G:C,	chrX:153294581:A:G,	chrX:153294684:C:T,
chrX:153294987:C:G,	chrX:153295012:C:T,	chrX:153295725:C:T,	chrX:153295726:G:A,
chrX:153295763:G:C,	chrX:153295782:C:G,	chrX:153295809:C:T,	chrX:153295810:G:A

Figure 3-2 shows the receiver-operator characteristic (ROC) curve for this classification task. The model can predict pathogenic variants from benign ones with an area under the ROC (auROC) of 0.98 ± 0.02 and 0.97 ± 0.02, for the Conv-Net and Feature-Net respectively, both with a p-value of $< 1 \times 10^{-8}$. Even though the auROC's are effectively identical for both models, there is a clear advantage in the performance characteristic of the Conv-Net: it outperforms in the low false positive rate region where variant classification matters.

The ROC curve is constructed by plotting the true positive rate (TPR) and false positive rate (FPR) at different threshold settings, however, these quantities are unaffected by class imbalance. Sometimes, it is useful to look at the precision of a classifier, defined as the number of true positives out of all the predictions it classified as positive, which is a quantity that is affected by class imbalance. For variant classification, the number of negatives (benign) is generally much higher than the positives (pathogenic), and it is important to know out of all the variants which are classified as pathogenic, how many of them are truly pathogenic. Because of this, it is often preferable to use precision-recall curves (PRC) to compare models for imbalanced dataset (Saito and Rehmsmeier, 2015). Figure 3-2 shows the PRC curves comparing the Conv-Net and Feature-

Net, with an area under the PRC (auPRC) of 0.95 ± 0.05 and 0.91 ± 0.06 respectively. It can be seen that the Conv-Net achieves higher precision when high recall (true positive rate) is desired.

For these predictions, we used an input of zero for the position feature of the PAS strength model, since each variant is not analyzed with respect to neighboring sites. However, in general, it may be advantageous to incorporate this information. For example, a variant may cause a large change to a nearby PAS, but if there is a much stronger neighboring PAS in the same gene, the effects of the variant may be dwarfed by this neighbor, and therefore not have any significant mechanistic effects.

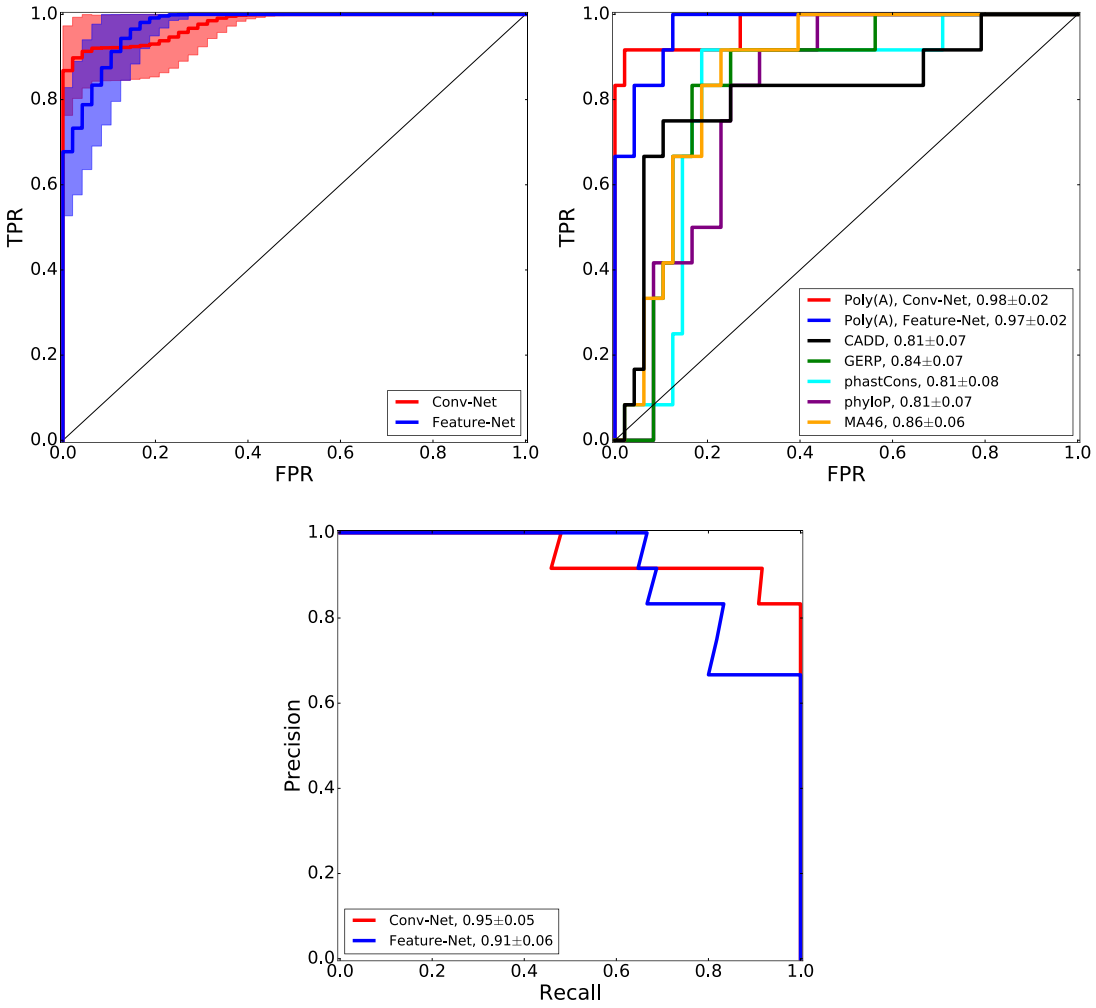


Figure 3-2. Classification performance of ClinVar variants near polyadenylation sites. (top-left) ROC curves comparing the variant classification performance of the Conv-Net and the Feature-Net. The shaded region shows the one standard deviation zone computed by bootstrapping. (top-right) ROC curves comparing our model's performance against other predictors. (bottom) PRC curves comparing the classification performance of the Conv-Net and Feature-Net. The area under the curve values are shown in the figure legend.

We further evaluate how the model compares with four phylogenetic conservation scoring methods: Genomic Evolutionary Rate Profiling (GERP) (Cooper *et al.*, 2005), phastCons (Siepel *et al.*, 2005), phyloP (Pollard *et al.*, 2010), and the 46 species multiple alignment track from the UCSC genome browser (Blanchette *et al.*, 2004). We also compare the predictions with Combined Annotation-Dependent Depletion (CADD), a tool which scores the deleteriousness of variants (Cooper *et al.*, 2010). Overall, as shown in Figure 3-2, the pathogenicity score from our model compares favorably, even though it has not been explicitly trained for this task. It is worth noting that although the model performed well for this ClinVar dataset, in general, a large difference in PAS strength does not necessarily imply pathogenicity, which is a phenotype that can be many steps downstream of 3'-end processing (Manning and Cooper, 2017).

The model can also be used to search for potential variants that would affect the regulation of polyadenylation. To visualize this approach, we applied the model and generated a mutation map (Alipanahi *et al.*, 2015) to a 100 nucleotide sequence in the human genome, where a ClinVar mutation that affects the polyadenylation signal is associated with β-thalassemia (Rund *et al.*, 1992). As shown in Figure 3-3, the polyadenylation signal is identified as an important region relative to other bases in the sequence.

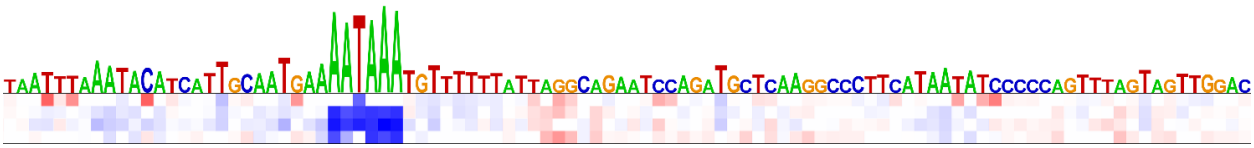


Figure 3-3. A mutation map of the genomic region chr11: 5,246,678-5,246,777. Each square represents a change in the model's score if the original base is substituted. The substituted base is represented in each row in the order 'ACGT'. Red/blue denotes a mutation that would increase/decrease the likelihood of the PAS for cleavage and polyadenylation.

3.3.3 Polyadenylation Site Discovery

The model is trained by centering the input sequence around a PAS at the cleavage site. If a PAS is off-center of the 200 nucleotides input sequence, or when no PAS is present, it stands to reason that the predicted PAS strength of the sequence would be small, due to the lack of sequence elements necessary for cleavage and polyadenylation. Alternatively, if the output of the PAS strength predictor is large, it would suggest that a PAS is present and is positioned near the center

of the input sequence. Naturally, we ask whether the model can be translated across the genome to find potential PAS. While there have been previous works on this task (Cheng *et al.*, 2006; Akhtar *et al.*, 2010; Chang *et al.*, 2011), our model is not explicitly trained for this.

To illustrate an example of a predicted PAS track, we selected a section of the human genome and applied the Conv-Net strength model to it in a base-by-base manner (Figure 3-4). The average strength prediction from all eight tissues, without application of any filtering or thresholding, is shown. For this example, we chose a region of the genome with multiple PAS, and where there are differences between annotation sources.

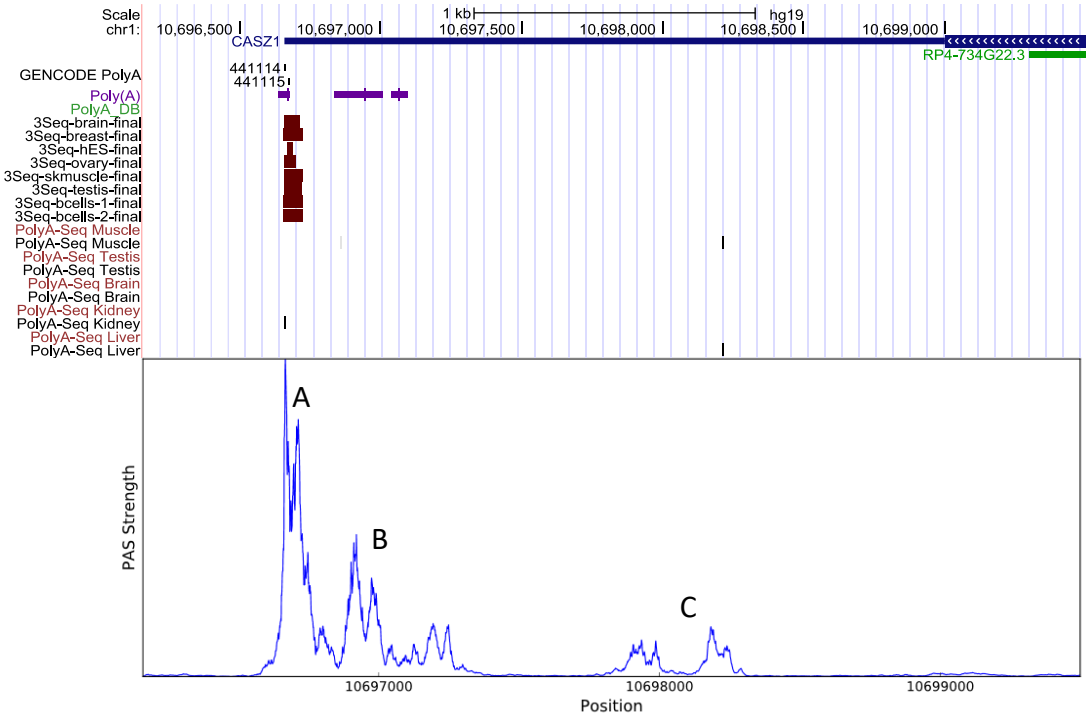


Figure 3-4. Example application of scanning the Conv-Net model across a section of the human genome to identify potential polyadenylation sites. (Top) Snapshot from the UCSC genome browser, showing tracks from top to bottom: GENCODE gene annotations, GENCODE Poly(A) track, predicted and reported PAS from polyA_DB (Cheng *et al.*, 2006; Zhang *et al.*, 2005), 3'-Seq (Lianoglou *et al.*, 2013), and PolyA-Seq (fwd. and rev. strands) (Derti *et al.*, 2012). (Bottom) Predictions from the model.

The predicted peaks labeled region A are present in all annotation sources. It is not a single sharp peak, indicating that various PAS are possible in this region. This agrees with the GENCODE Poly(A) track, which indicates that there are two peaks in this region, as well as 3'-Seq, which shows that there are RNA-Seq reads that map across a broad region for various tissues. As

mentioned earlier, the location of cleavage and polyadenylation is not exact. Region B is less well-defined, is weaker, and approximately aligns with the predicted positions from another PAS predictor (Cheng *et al.*, 2006), as well as the muscle track from PolyA-Seq (in light gray). Finally, a small peak is observed in Region C, predicted to be a very weak PAS, which is present in PolyA-Seq. Note that the model is trained only from 3'-Seq reads and has no knowledge of RNA-Seq information from other datasets or other annotation sources.

To assess the model's ability in discovering PAS, we created a dataset with positive and negative examples to assess its classification performance. There is no general consensus from previous works on what constitutes a proper criterion to construct negative sequences or a standardized dataset for this task (Ji *et al.*, 2015). We therefore defined the evaluation dataset based on our annotations and reads from 3'-Seq. Positive targets consist of annotated PAS in the 3'-UTR that has 10 or more reads. Since it is generally not appropriate to simply use random genomic sequences or locations for the negative set, we extracted the two immediately adjacent genomic regions near a PAS to ensure that both the negative and positive sequences have similar compositions (Figure 3-5). Each sequence is fed as input into the strength predictor, and the output from all tissues are averaged into a single value which is used for classification. The positional information of the sequence is not used (i.e. it has a position of zero). The AUC for classifying sequences with a PAS from negative sequences (without a PAS) for the LR, Feature-Net, and the Conv-Net are respectively 0.887± 0.003, 0.895± 0.004, and 0.907± 0.004. It is worth mentioning that of the negative sequences, 19% contain one of the two canonical polyadenylation signals (AAUAAA and AUUAAA), and 74% contain at least one of the known polyadenylation signals, meaning the model can distinguish real PAS from the background. It does not simply look for the presence of polyadenylation signals to detect PAS in the genome.

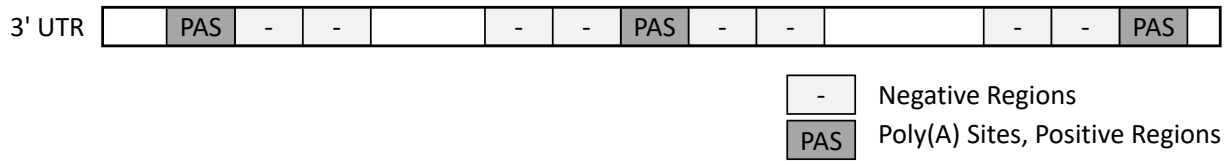


Figure 3-5. Two regions immediately adjacent to each polyadenylation site (PAS) are defined as negatives for classification. This ensures that the negatives have similar nucleotide composition compared to the positive sequences. Regions that are not between existing PAS are excluded to avoid including terminal exonic regions. If the spacing between adjacent PAS cannot fit four negative regions, they are also excluded from the negative set.

It is interesting to observe that there is a relatively smaller difference in the AUC's for all models, especially between the Conv-Net and the logistic regression model, compared to previous tasks, which differed more drastically in performance. Identification of PAS from the genome is a simpler problem, characterized by the presence of features that are generally well-documented in the literature (Tian and Manley, 2016). For this, a logistic regression classifier may be sufficient. On the other hand, predicting the strength of a PAS given its sequence is arguably more complex. Instead of a binary classification problem, a strength predictor must quantify a PAS by integrating its genomic signature, and predict how it compares with another site, which may also contain all the core polyadenylation signatures, but differ in other ways with respect to its sequence. This observation is supported by larger differences in the models' performance to the PAS selection problems in Table 3-2 and 3-3, which require strength quantification.

3.3.4 Predicting the Effect of Oligonucleotide Treatment

Antisense oligonucleotides therapies involve targeting RNAs via complementary base pairing, and can modulate RNA function by blocking the access of cellular machinery to the RNA (Kole et al., 2012). Application of this approach was demonstrated by Vickers et al. in the 3'-UTR, where oligonucleotides targeting polyadenylation signals and sites modulated the abundance of an mRNA (Vickers et al., 2001). Based on this, we show the utility of our model to provide an in-silico evaluation of oligonucleotides targeting regions near the PAS.

Three distinct forms of the transcript, Type 1, 2, and 3 are described in the study. A schematic of the E-selectin mRNA and the position of the polyadenylation signal, along with the targeted region of the oligonucleotides used are shown in Figure 3-6. All three forms harbor the canonical polyadenylation signal AAUAAA. A non-canonical polyadenylation signal AGUAAA is also present between the Type 1 and Type 2 cleavage sites, which is selected when the corresponding signals from Type 1 and Type 2 are blocked. Here, it is referred to as the Type 4 form of the transcript.

According to the study, Type 3 is by far the dominant form of the transcript, followed by Type 1 and Type 2 (no differentiation is reported between them). Type 4 is the least common. Using the model, the predicted strengths for the corresponding PAS for Type 1 to 4 transcripts are respectively: -0.242, -0.420, 0.020, -0.765. These values do not account for the position of the PAS. If the relative positions of the four PAS are provided to the model, then the strengths become:

-0.242, -0.170, 0.606, -0.584 (where Type 1 is assumed to be in position zero). These predictions match the observed abundances of the mRNA from the study.

The Vickers study performed a non-quantitative RT-PCR to assess the abundance of isoforms by administering different combinations of oligonucleotides targeting select regions of the transcript. To simulate this, we blocked the same regions of the input sequence complementary to the oligonucleotides by replacing the nucleotides with an N base and predicted the resulting strengths of each PAS. The results are depicted in Figure 3-6, where the predicted values are arranged in an image to match the gel from the original paper. Each column is scaled such that the sum of the intensities of each column is constant, but otherwise, no additional processing is performed. The original paper does not provide values from RT-PCR that would permit quantitative comparison with the output of our model, but qualitatively, the patterns of polyadenylation are generally captured. Note that the original paper mentions that Type 1 and 2 transcripts are shorter and therefore more efficiently amplified by PCR, and thus appear brighter than expected compared to Type 3. This experimental bias does not affect our simulated RT-PCR results in Figure 3-6.

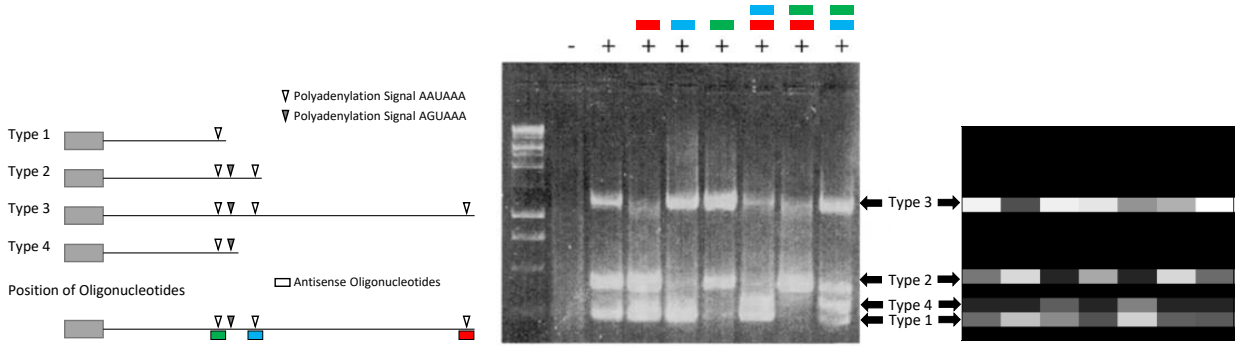


Figure 3-6. Predicting the effect of an antisense oligonucleotide experiment. (left) Schematic of human E-selectin 3′-UTR and the possible transcripts from polyadenylation site selection, reproduced from (Vickers *et al.*, 2001). The regions targeted by the oligonucleotides are shown. (right) Predicted PAS strength, simulating the effects of blocked nucleotides due to oligonucleotide treatment. (center) The figure from the original paper is reproduced here for ease of comparison. The oligonucleotides applied are shown on top of each column.

3.4 Discussions

3.4.1 Effect of Genomic Features on the Model's Predictions

To understand how different features contribute to performance, we train models using only individual feature groups. Table 3-5 shows each model's classification performance. Even though the polyadenylation signals are generally considered to be a main signature of PAS, they only partially account for the predictive performance for PAS selection compared to the full feature set. Overall, n-mers features are major contributors to the Feature-Net's performance, which is sufficiently rich to capture many motif patterns. It should be noted that each feature group has a different number of features (Appendix B), and therefore individual features in the larger feature groups may contribute only weakly, but as a whole affect predictions considerably. Position alone has very poor predictive capability, even though it was suggested to be a key feature in determining whether a PAS is used for tissue-specific regulation (Weng *et al.*, 2016). We also conducted an investigation on the uniqueness of each feature group, by training models with all features minus each feature group from Table 3-5. Removing the polyadenylation signals from the feature set reduces the performance from 0.866 ± 0.004 to 0.840 ± 0.008. All other groups, when removed, do not significantly reduce the performance of the model compared to the full feature set. This suggests that many features are redundant, and if removed, can be compensated by features in another group.

To see the contributions of individual features, we computed the gradient of the output with respect to the input feature vector of the neural network. This is referred to as the feature saliency of a prediction, and the gradients of features with large magnitudes can be interpreted as those that need to change the least to affect the prediction the most (Simonyan *et al.*, 2014). For this, we computed the feature saliency of all sites in our test set and selected the features that on average have the largest magnitude. Table 3-6 shows the top 15 features computed using this method and the direction in which the feature affects the strength of a PAS, where an up arrow indicates that the effect is positive.

Table 3-5. Comparison of Feature-Net PAS selection performance between competing sites using feature subsets.

Feature Group	AUC
All	0.866 ± 0.004
Poly(A) Signal	0.728 ± 0.004
Position	0.553 ± 0.004
Cis-Elements	0.608 ± 0.009
RBP Motifs	0.676 ± 0.009
Nucleosome Occupancy	0.656 ± 0.006
1-Mers	0.762 ± 0.004
2-Mers	0.794 ± 0.002
3-Mers	0.817 ± 0.004
4-Mers	0.833 ± 0.005

Table 3-6. Top 15 features of the Feature-Net, and the direction in which each feature can increase (↑) or decrease (↓) the strength of a polyadenylation site.

Rank	Region	Feature Name	Direction
1	5'-3'	PolyA Signal, AAUAAA	↑
2	---	Log distance between PAS	↑
3	5'-3'	PolyA Signal, AUUAAA	↑
4 to 15	5'-3'	1-mer, C	↓
	5'-3'	1-mer, U	↑
	5'-3'	2-mer, AG	↓
	3'-5'	2-mer, CA	↓
	3'-5'	3-mer, AAA	↑
	5'-3'	3-mer, UGU	↑
	5'-5'	3-mer, UGU	↑
	3'-5'	4-mer, AAAA	↑
	5'-5'	Cleavage Factor Im, UGUA	↑
	5'-3'	PolyA Signal, CAAUAA	↑
	5'-3'	PolyA Signal, AUAAAG	↑
	5'-5'	PolyA Signal, AGUAAA	↑

The top three features are consistent for all tissue types. Other features vary slightly between tissues and are grouped together unordered. As expected, the two most common canonical polyadenylation signals are the top features which increase the strength of a PAS. The log distance between PAS is also deemed to be important. Some features in this list are consistent with mechanisms of core elements known to be involved in cleavage and polyadenylation, including the upstream UGUA motif which the cleavage factor Im complex binds to, and a GU-rich sequence near the polyadenylation site (Tian and Graber, 2012). The genomic context upstream of the PAS appears to be more important, as most of the top features are in either the 5'-5' and 5'-3' region. Interestingly, three of the features reduce the strength of a PAS. They are the frequencies of C and AG nucleotides in the upstream region and the CA nucleotides downstream of the cleavage site, the latter of which is aligned with the knowledge that the C-terminal domain of RNA polymerase II interacts with CA-rich RNA sequences, and is known to play a role in inhibiting polyadenylation (Kaneko and Manley, 2005).

3.4.2 Determining Tissue-Specific Polyadenylation Features

Given that APA is used to achieve tissue-specific gene expression, we investigate whether our model can provide insights into this phenomenon. Previous computational approaches to address this problem are present in the literature. In Hafez et al., an A-rich motif was found to be enriched in brain-specific PAS (Hafez *et al.*, 2013). In Weng et al., the position of a PAS relative to another PAS and its position in the gene was found to be the strongest indicator of whether it is tissue-specific (Weng *et al.*, 2016). The computational models for both these works were trained to directly classify whether a PAS is tissue-specific. To be consistent with the methodology presented in this work, we will analyze our models without re-training them.

We use the set of tissue-specific and constitutive PAS defined in (Weng *et al.*, 2016) and apply the Feature-Net to generate predictions. To determine which feature is associated with tissue-specific PAS, we use the same gradient-based method as described in Section 3.4.1 to examine the top 200 most confident predictions for tissue-specific PAS, where our model predicts that at least one of the tissue outputs is considerably different than the rest, and for constitutive PAS, where our model predicts that all tissue outputs do not differ significantly. The magnitude of the gradients is then analyzed to see which features have a statistically greater effect on tissue-specific PAS compared to constitutive PAS. Statistical significance was determined by a permutation test by shuffling the predictions indicating whether a PAS is tissue-specific or constitutive. Applying a conservative p-value of 0.05/1506 (# of features) = 3×10^{-5}, 15 features were found to be associated with the model's ability to predict tissue-specific PAS. This is shown in Table 3-7. In the column indicating direction, an up arrow means the presence of the feature makes the site more likely to be tissue-specific, and vice versa.

Table 3-7. Features associated with the prediction of tissue-specific polyadenylation sites, and whether the presence of the feature makes a polyadenylation site more (↑) or less (↓) tissue-specific.

Region	Feature Name	P-value	Direction
5'-5'	4-mer, UUGU	8.0×10^{-11}	↓
3'-3'	3-mer, UUG	9.9×10^{-09}	↑
3'-3'	4-mer, CCCC	5.7×10^{-08}	↓
5'-5'	3-mer, UGU	6.8×10^{-08}	↓
3'-3'	4-mer, UCCC	1.1×10^{-07}	↓
5'-3'	4-mer, CGGC	1.0×10^{-06}	↓
5'-5'	Cis-element, UUUGUA	1.7×10^{-06}	↓
5'-5'	Cleavage Factor Im, UGUA	2.2×10^{-06}	↓
5'-5'	3-mer, UUG	3.4×10^{-06}	↓
5'-5'	3-mer, AUC	7.4×10^{-06}	↑
3'-3'	3-mer, UCC	1.2×10^{-05}	↓
5'-5'	2-mer, UC	1.7×10^{-05}	↑
5'-5'	4-mer, AUCC	1.9×10^{-05}	↑
5'-5'	2-mer, UU	2.0×10^{-05}	↓
3'-3'	3-mer, CCU	2.1×10^{-05}	↓

All but one of the entries in the table describe features that are in the 5'-5' and 3'-3' region, that is, most of them are located away from the cleavage site. Various G/U-rich features top the list, where its position upstream suggests the PAS is more likely to be constitutive but if downstream, tissue-specific. Polyadenylation signals are absent from the list. No hexamers other than UUUGUA was found, which was previously identified as a feature by statistical analysis from (Hu *et al.*, 2005). However, we found no association of this hexamer with tissue-specific polyadenylation from the literature. Given that the model only sees sequences from +/- 100 bases from the cleavage site, it may be possible that other more distal tissue-specific signatures may be present. Alternatively, sequence signatures may not be fully predictive since tissue-specific proteins can act by modulating core polyadenylation proteins instead of directly binding to the transcript (MacDonald and McMahon, 2010).

3.4.3 A Convolution Neural Network Model of Polyadenylation to Predict the Effect of Genomic Variations

We initially began this work with a feature-based model, and subsequently added a Conv-Net for comparison expecting it to approach the performance of the Feature-Net, not necessarily surpassing it. Given that the polyadenylation features were derived from many publications and multiple research groups, the prior work that went into obtaining the feature-based models, which include the logistic regression classifier used as a baseline in this work, should not be underestimated. The fact that the Conv-Net could learn a better model absent any insights or

hypotheses about mechanism, is an interesting result on its own. This is surprising at first, but perhaps not so if viewed in the context of other applications of machine learning like computer vision, where hand-crafted features have been largely superseded by models which learn directly from image pixels (LeCun *et al.*, 2015).

On top of this, the Conv-Net has additional advantages that are not available in feature-based models. For instance, it is completely free to discover novel sequence elements that may be relevant for polyadenylation regulation from data. An example set of filters from the Conv-Net model is shown in Figure 3-7. It also has the potential to be more computationally efficient. Feature extraction from sequences can be the most computationally intensive aspect of a model during inference. This is not required for models that directly operate on sequences. There are additional operations that are required in the Conv-Net, but these computations can be significantly sped up by graphics processing units, which can be important for application of the model to entire genomes.

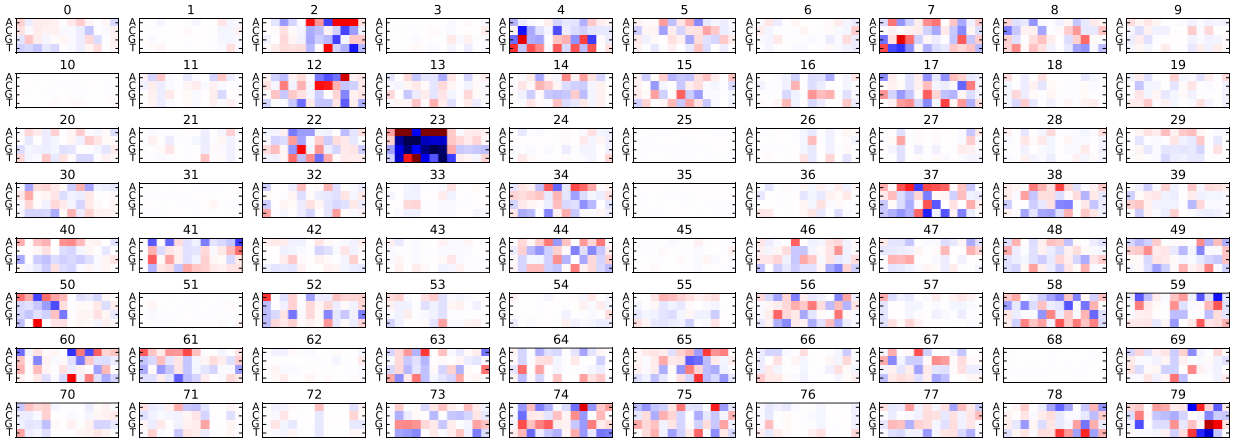


Figure 3-7. An example set of the 80 filters that are learned by the Conv-Net. All filters have been mean-subtracted and plotted with the same scale (i.e. the max and min for each filter plot is the same). Red and blue denote positive and negative values respectively. Various filters are blank, suggesting the number of filters in the Conv-Net model can be reduced. A filter that detects the two most common polyadenylation signal motifs, ATTAAA and AATAAA can be seen in filter #23, which is followed by a strong avoidance of a T nucleotide. Filters resembling GU-rich elements, such as filter #4 can also be found.

Since the Conv-Net operates directly on the genomic sequence, it also enables one to perform analysis at the single-base resolution more naturally. By analyzing the flow of gradients, the Conv-Net can determine how each base in the input sequence changes the output of the model.

If a model requires feature extraction, such as the Feature-Net, the output must be analyzed relative to each feature. Furthermore, in the Feature-Net, many features are derived in discrete sections of the genome (four in this case, see Appendix B) to reduce the dimensionality of the input. The Conv-Net on the other hand, is more efficient at sharing model parameters, thereby enabling the motif filters to be applied at much finer spatial steps across a genomic sequence (a stride of 1 is used, see Table 3-1), while still making overfitting manageable during training. By computing the gradients (Simonyan *et al.*, 2014), analysis regarding the magnitude and direction of the effect of each base on the model's output can be performed. This has the potential to offer a prescription to the design of oligonucleotides for antisense therapies. Figure 3-8 shows the saliency map of a region of the oligo-targeted mRNA examined in the Section 3.3.4, which spans the first three polyadenylation signals. This is different than the previous mutation map approach, which visualizes the change in the model's predictions between the reference genome and mutation at each base for the alternate nucleotides. Here, the gradient of each base relative to the model's prediction is shown, which includes the reference genomic sequence. It is also computed differently, involving a single backpropagation step in the Conv-Net. This operation is not readily available in the Feature-Net, where the genomic sequence is separated from the model by a feature extraction pipeline, and therefore dependent on the design of the pipeline. This saliency map can be generated for large stretches of the genome to look for potential sensitive regions to alter polyadenylation for therapeutic purposes.

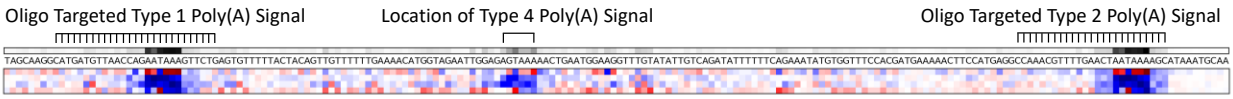


Figure 3-8. Saliency map from the Conv-Net of a section of the oligo-targeted mRNA from (Vickers *et al.*, 2001). The base is represented in each row in the order 'ACGT'. Red means the base increases the likelihood of the sequence for cleavage and polyadenylation. Blue is the reverse. The sum of the magnitude of the gradient is shown above the saliency map to suggest how sensitive the nucleotide is to the strength of the polyadenylation site. The position of the oligonucleotide used in the study is shown at the top. The Type 4 Poly(A) signal is labeled also but was not targeted in the original study.

3.5 Conclusion

Regulation of polyadenylation is a crucial step in gene expression, and mutations in DNA elements that control polyadenylation can lead to diseases. Accurate, predictive models of polyadenylation

will enable a deeper understanding of the sequence determinants of gene regulation and provide an important new approach to detecting and treating damaging genetic variations. In this chapter, we have presented the polyadenylation code, a versatile model that can predict alternative polyadenylation patterns from transcript sequences and can generalize to multiple tasks that it was not trained on. Beyond its original trained usage to predict PAS selection from competing sites, it can classify variants near PAS and can be used for PAS discovery. We provided an analysis of what sequences increase and decrease the strength of a PAS and identified features that are associated with tissue-specific and constitutive PAS. We also illustrate the potential of our model to infer, and design for, the effects of antisense oligonucleotide treatment in the 3'-UTR.

Chapter 4
Discussion

This chapter is a collection of various insights gained in the duration of this thesis work, as well as additional topics. It includes subjects such as interpretability of computational models as viewed by the wider community and the role of machine learning in genomic medicine. The section on interpretability includes partial content from the publication (Leung *et al.*, 2016):

M. Leung, A. Delong, B. Alipanahi, and B. Frey. (2016) "Machine Learning in Genomic Medicine: A Review of Computational Problems and Data Sets". Proceedings of the IEEE, 104(1), 179-197.

4 Discussion

4.1 Interpretability of Machine Learning Models

There is an inherent trade-off between the interpretability and the accuracy of a predictive model. Generally, linear models such as regression and models that depend on few variables are viewed to be more interpretable. They are regarded as so because one can more easily look at the parameters of these models to assess which features may be important. On the other hand, so-called 'black-box' models, which tend to have more predictive capability and often rely on fewer assumptions, are viewed to be harder to interpret. Neural networks for example, popularized by recent development in deep learning, can have many variables (hundreds of thousands to millions of parameters for typical vision and natural language processing problems), and many of these parameters are stacked in multiple layers such that they are not directly connected to the inputs and outputs of the model. This leads to these 'black-box' models to sometimes be disfavored in some industries. For example, the finance industry tends to have stricter regulatory and documentation requirements, which may require sacrificing predictive accuracy to instead use models with fewer variables such that a prediction can be 'explained'. It can be said that this sentiment is also true for the genome biology community, and likely will be for the drug development industries as well, which in the foreseeable future will more heavily rely on computational methods.

One reason to strive for interpretable models stems from the fact that as humans, we find satisfaction in being able to explain (simply) why something works. However, from a machine learning perspective, genome biology differs from domains such as image recognition, speech

recognition, and natural language processing in a very important way. Humans are very good at these latter tasks, which involve human perception (e.g. seeing images, hearing speech) and human action (e.g. grabbing an object, responding to words). In stark contrast, we can think of no reason why the genome should be interpretable by humans. Whereas there has been evolutionary pressure for humans to perceive, interpret, and respond to patterns of light, such as that produced by an advancing tiger, there has been no pressure for the genome to be interpretable by humans or for humans to develop the capacity to interpret the genome. Consequently, it is important to incorporate the latest biological knowledge and data into learning algorithms and to carefully and rapidly validate models in different ways, since the models cannot be 'checked by eye'.

In this section, we make a few remarks on the interpretability of machine learning models in the context of genome biology.

4.1.1 An Alternate View on the Interpretability

Within some application domains, interpretability is deemed to be quite important (Rudin and Wagstaff, 2013). For example, the need to extract insights from computational models is a common theme from reviewers for published journal articles. Following the movement to rely more on data-driven explanations rather than conceptual explanations, it may be more beneficial to adopt the mindset to focus on developing systems that can be queried by human experts. In this view, less emphasis is placed on the internal workings of the system. Instead, it is more desirable that the system more accurately reflects the task that is modeled. In the context of deep learning, instead of examining the parameters of a neural network and coming up with an 'interpretation', a more useful exercise would be to ask the system about relationships between inputs and outputs. For instance, whether a cell variable will increase or decrease if a particular nucleotide is changed, or whether changing a pair of nucleotides leads to a change in the cell variable that cannot be accounted for by independent, additive contributions. This question-and-answer interaction between the expert and the machine learning model provides a quantitative, data-driven interpretation.

The community's ability to derive interpretations from machine learning models is likely to improve as these models become more effective in practice. However, consider for a moment what it would mean to wait for interpretability challenges to be 'solved', to forego the benefit of more accurate models in genomic and precision medicine. Throughout history, many advances

were made by noticing a pattern without understanding the precise causal mechanisms involved. For example, in 1847, Ignaz Semmelweis found that washing hands before delivering babies was correlated with fewer maternal deaths. He achieved a two-thirds reduction in mortality rate a full 25 years before Louis Pasteur established the relationship between germs and disease. Viewed from a different perspective, if machine learning can be used to identify the genetic cause of a disease and an effective therapy, it is unlikely that the patient who benefits will care about interpretability. Nevertheless, machine learning researchers working with biologists and clinicians should be prepared for a strong bias towards interpretable models.

4.1.2 Interpreting Neural Network Models

There have been many efforts to improve the interpretability of machine learning models, and in particular deep neural networks, to take advantage of their predictive capability while making them more accessible to a broader user base. Erhan et al. introduced the idea of tuning the input to maximize the activation of a hidden unit. This enables one to see what kind of inputs a hidden unit is sensitive to (Erhan *et al.*, 2009). The method has been applied to deep architectures trained on millions of images, where neurons that correspond to face, cat, and human body detectors have been found (Le *et al.*, 2013). They do so by designing a norm-constrained input that maximizes the activity of a neuron deep inside the network. Zeiler and Fergus (Zeiler and Fergus, 2013) aim to visualize the input variations that high-level features respond to in a convolutional neural network. They do so by generating several diverse inputs that each cause high activations in a feature map deep within the network. Several compelling visualization approaches use backpropagation to efficiently visualize how deep architectures respond to input perturbations (Simonyan *et al.*, 2014; Mahendran and Vedaldi, 2015). This approach was used in the work in Chapters 2 and 3 to investigate how genomic features affect the predictions of the splicing and polyadenylation neural network models.

4.1.3 Convolutional Neural Networks for Genomics

Convolutional neural networks (CNN) have been highly successful in computer vision and are now the dominant approach in many image recognition tasks, approaching that of human performance. The purpose of its architecture is two folds (LeCun *et al.*, 2015). First, images tend to contain local motifs, such as the eyes of the face, whose presence can be beneficial for face detection. Second, a motif can appear anywhere in an image, whose position generally does not

affect common image recognition tasks, such as classifying whether a photo contains cats or not. For example, whether a cat is positioned in the top left corner or the bottom right corner of an image generally does not affect whether the image is classified as containing a cat. CNN exhibits translational invariance in its input, meaning the values of the output tend to be robust to translations of local motifs in the input, which makes it suitable for this class of problems. Note however that it is still possible to do localization of objects with a CNN, for example, by analyzing the input gradients (Simonyan *et al.*, 2014).

The characteristic of CNN is also appropriate to some problems related to the genome, which consists of motifs that the molecular machineries recognize and whose exact position along the genome can vary without significantly affecting the regulatory mechanisms. For example, in the context of the polyadenylation, the polyadenylation signal which is usually present before a cleavage site can vary by tens of bases. As a result, many works have since applied the use of CNNs to learn from genomic sequences for molecular phenotype predictions (Alipanahi *et al.*, 2015; Zhou and Troyanskaya, 2015; Kelley *et al.*, 2016; Angermueller *et al.*, 2017). Used in the work discussed in Chapter 3 for the computational model of polyadenylation, another advantage of CNN's is they are end-to-end, providing a mapping from the genomic sequences to a prediction, without additional processing steps, such as feature extraction. Feature extraction is a significant computational bottleneck in the splicing code (Xiong *et al.*, 2015). In relation to interpretability, the filters learned by the CNN's can provide the familiar sequence logos that computational biologists are used to (Alipanahi *et al.*, 2015). Other advantages of CNN's in the context of problems in computational biology can be found in (Angermueller *et al.*, 2016).

4.2 Genotype-Phenotype Modeling

In Chapter 1, we discussed the approach of predicting cell variables from genotype, which can act as an intermediate step for more complex phenotype, such as disease risks. Bridging this genotype-phenotype gap is not a new concept (Gjuvsland *et al.*, 2013), and the capability to do so has been referred to by some scientists as the Holy Grail of genetics. To build these computational models, data profiling cell variable under diverse input conditions are required. Cell variables are more difficult to measure than phenotypic observations such as whether a patient is sick. However, consider measuring few variables per patient for a large number of individuals, versus taking hundreds of thousands of cell variable measurements per patient for a smaller group of people. We believe that the latter approach gives us a better chance at deciphering the genomic instructions

of the cell, where there is much more information available about the biological mechanisms at play, and therefore more data overall for a model to learn from. In a sense, we are making the 'genomic invariance' assumption, that is, we assume that regulatory processes act the same across the entire genome and so we can learn the DNA-to-cell variable relationship by treating different locations in the genome as independent measurements.

On the other hand, even if we have computational models of many different cell variables, such as those in Table 4-1, it is not obvious how they can be combined to construct a model of the cell, much less to predict and decipher the molecular pathways of higher-order phenotypes like diseases. For example, the two computational models described in this thesis have a very local, non-overlapping, view of the genome as input. It is known however, that splicing and polyadenylation are coupled in both time and space (Bentley, 2014), that is, they occur simultaneously on a given transcript and can influence one another. Also, joint modeling of splicing and polyadenylation is arguably orders of magnitude simpler than 'long-range' interactions of different parts of the genome that collectively influence a phenotype.

Table 4-1. A sample of cell variables related to genomic regulatory mechanisms.

Cell Variable	Brief Description	Relevance to Disease	Reviews	Related Works
Identification of structural and functional regions of the genome	Attaching meaning to, or annotating, different regions of the DNA, such as marking the boundary of introns and exons, and identifying parts that have regulatory functions.	Changes in genomic sequences can cause a region which previously served a particular function to become non-functional and vice-versa, or changing its intended function, thereby affecting regulation.	(Yandell and Ence, 2012; Alexander *et al.*, 2010; Yip *et al.*, 2013)	(Sonnenburg *et al.*, 2007; Saeys *et al.*, 2007)
Binding sites for transcription regulation	Binding of proteins to specific sequence elements of the DNA controls whether transcription can occur, as well as the rate at which it happens.	Sequence variations to sequence patterns that proteins, such as transcription factors and complexes that 'unwind' the DNA, bind to can alter whether a gene is transcribed.	(Lee and Young, 2013; Maston *et al.*, 2006)	(Alipanahi *et al.*, 2015; Li *et al.*, 2010)
Splicing patterns	Splicing modifies the pre-mRNA by removing introns and selecting which exons are retained.	Changes to the regulatory elements that control splicing can change the characteristics of the gene products, and in some cases, cause them to be non-functional.	(Wang and Burge, 2008)	(Barash *et al.*, 2010; Xiong *et al.*, 2011, 2015; Leung *et al.*, 2014)
Cleavage site selection and polyadenylation	The ends of transcripts are cleaved and a stretch of adenine bases are attached before they are ready for translation. Cleavage can occur in one of multiple sites within a transcript.	Modifications to sequence elements can alter where cleavage occurs, which determines whether binding sites for regulatory proteins are present or absent on the transcript. This alters its stability and translation efficiency.	(Danckwardt *et al.*, 2008; Elkon *et al.*, 2013)	(Akhtar *et al.*, 2010; Chang *et al.*, 2011)

RNA structure	The RNA folds into three-dimensional (3D) structures, which influence how it interacts with other molecules in the cell.	The mRNA, beyond the information it contains for encoding protein, has 3D structure. This structure can affect processes that it is involved with, such as transcription, splicing, and translation.	(Laing and Schlick, 2011; Wan *et al.*, 2011)	(Lorenz *et al.*, 2011)
Protein structure	The outcome of translation is a sequence of amino acids that folds into a protein. The protein's 3D structure is crucial for its function, as it interacts with DNA, RNA, and other proteins.	Structure affects function. The ability to predict protein structure from sequences can help in understanding the biological function of a gene, and how misfolding of proteins contribute to disease.	(Floudas, 2007)	(Troyanskaya, 2014; Di lena *et al.*, 2012)

In terms of genomic medicine, it is likely that the association of the genome to some diseases might simply be too complex to be modeled from a practical number of 'inputs'. This contrasts with image or speech recognition, where we know what the prediction ought to be given the input. Furthermore, it should be noted that due to the inherent stochasticity of cellular processes, environmental factors that differ from person to person (even for identical twins), and uninherited variants from the parent that can affect offsprings, the genotype of an individual may not be sufficient to completely determine their phenotype (Burga and Lehner, 2012). Therefore, we do not expect computational methods to be able to entirely replace laboratory and clinical diagnosis, but they should greatly shorten the time required for these methods of analysis by reducing the search space of hypotheses that need to be validated.

Nevertheless, computational models of cell variables are a step in the right direction to augment an individual's genotype profile with additional information which may be useful for genomic medicine. With our current toolset, it is unlikely that a newly uncharacterized genetic disease can have its molecular mechanisms understood purely by computational approaches. Association of a cell variable with a disease will involve a collaboration of clinical, laboratory, and computational analysis to find the underlying cause. However, after a cell variable is found to be associated with a disease, such as the spinal muscular atrophy example in Chapter 1, computational models can offer a prescription to how the disease can be alleviated by therapies. For example, the models can point to a part of a transcript that should be modified or blocked, to reverse the change in a cell variable associated with a disease, for example via RNA therapeutics like antisense oligonucleotides (Kole *et al.*, 2012).

References

Abadi,M. *et al.* (2015) TensorFlow: Large-scale machine learning on heterogeneous distributed systems. *arXiv:1603.04467*.

Adzhubei,I.A. *et al.* (2010) A method and server for predicting damaging missense mutations. *Nat. Methods*, **7**, 248–249.

Ahn,S. *et al.* (2012) Bayesian posterior sampling via stochastic gradient fisher scoring. In, *29th International Conference on Machine Learning (ICML 2012)*, 1591–1598.

Akhtar,M.N. *et al.* (2010) POLYAR, a new computer program for prediction of poly(A) sites in human sequences. *BMC Genomics*, **11**, 646.

Albert,F.W. and Kruglyak,L. (2015) The role of regulatory variation in complex traits and disease. *Nat. Rev. Genet.*, **16**, 197–212.

Alberts,B. *et al.* (2002) Molecular biology of the cell. *Garland Science*.

Alexander,R.P. *et al.* (2010) Annotating non-coding regions of the genome. *Nat. Rev. Genet.*, **11**, 559–571.

Alipanahi,B. *et al.* (2015) Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning. *Nat. Biotechnol.*, **33**, 831–838.

Altshuler,D.M. *et al.* (2010) Integrating common and rare genetic variation in diverse human populations. *Nature*, **467**, 52–8.

Angermueller,C. *et al.* (2016) Deep learning for computational biology. *Mol. Syst. Biol.*, **12**, 878.

Angermueller,C. *et al.* (2017) DeepCpG: accurate prediction of single-cell DNA methylation states using deep learning. *Genome Biol.*, **18**, 67.

Bamshad,M.J. *et al.* (2011) Exome sequencing as a tool for Mendelian disease gene discovery. *Nat. Rev. Genet.*, **12**, 745–755.

Barash,Y. *et al.* (2013) AVISPA: a web tool for the prediction and analysis of alternative splicing. *Genome Biol.*, **14**, R114.

Barash,Y. *et al.* (2010) Deciphering the splicing code. *Nature*, **465**, 53–9.

Barbosa-Morais,N.L. *et al.* (2012) The evolutionary landscape of alternative splicing in vertebrate species. *Science*, **338**, 1587–1593.

Beaudoing,E. *et al.* (2000) Patterns of variant polyadenylation signal usage in human genes. *Genome Res.*, **10**, 1001–10.

Bengio,Y. (2009) Learning deep architectures for AI. *Found. Trends Mach. Learn.*, **2**, 1–127.

Bengio,Y. *et al.* (2013) Representation learning: a review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, **35**, 1798–828.

Bentley,D.L. (2014) Coupling mRNA processing with transcription in time and space. *Nat. Rev. Genet.*, **15**, 163–75.

Berger,M.F. *et al.* (2006) Compact, universal DNA microarrays to comprehensively determine transcription-factor binding site specificities. *Nat. Biotechnol.*, **24**, 1429–1435.

Bishop,C. (2006) Pattern recognition and machine learning. *Springer*.

Blanchette,M. *et al.* (2004) Aligning multiple genomic sequences with the threaded blockset aligner. *Genome Res.*, **14**, 708–15.

Brawand,D. *et al.* (2011) The evolution of gene expression levels in mammalian organs. *Nature*, **478**, 343–8.

Burga,A. and Lehner,B. (2012) Beyond genotype to phenotype: Why the phenotype of an individual cannot always be predicted from their genome sequence and the environment that they experience. *FEBS J.*, **279**, 3765–3775.

Cartegni,L. and Krainer,A.R. (2002) Disruption of an SF2/ASF-dependent exonic splicing enhancer in SMN2 causes spinal muscular atrophy in the absence of SMN1. *Nat. Genet.*, **30**, 377–384.

Caruana,R. and Niculescu-Mizil,A. (2006) An empirical comparison of supervised learning algorithms. *Proc. 23rd Int. Conf. Mach. Learn. - ICML '06*, 161–168.

Chang,T.H. *et al.* (2011) Characterization and prediction of mRNA polyadenylation sites in human genes. *Med. Biol. Eng. Comput.*, **49**, 463–72.

Cheng,Y. *et al.* (2006) Prediction of mRNA polyadenylation sites by support vector machine. *Bioinformatics*, **22**, 2320–5.

Ching,T. *et al.* (2018) Opportunities and obstacles for deep learning in biology and medicine. *J. R. Soc. Interface*, **15**, 20170387.

Clamp,M. *et al.* (2007) Distinguishing protein-coding and noncoding genes in the human genome. *Proc. Natl. Acad. Sci.*, **104**, 19428–19433.

Collins,F.S. and Varmus,H. (2015) A new initiative on precision medicine. *N. Engl. J. Med.*, **372**, 793–5.

Cong,L. *et al.* (2013) Multiplex genome engineering using CRISPR/Cas systems. *Science*, **339**, 819–23.

Cooper,G.M. *et al.* (2005) Distribution and intensity of constraint in mammalian genomic sequence. *Genome Res.*, **15**, 901–913.

Cooper,G.M. *et al.* (2010) Single-nucleotide evolutionary constraint scores highlight disease-causing mutations. *Nat. Methods*, **7**, 250–251.

Cornish-Bowden,A. (1985) Nomenclature for incompletely specified bases in nucleic acid sequences: recommendations 1984. *Nucleic Acids Res.*, **13**, 3021–30.

Costanzo,M. *et al.* (2010) The genetic landscape of a cell. *Science*, **327**, 425–431.

Crick,F.H. *et al.* (1961) General nature of the genetic code for proteins. *Nature*, **192**, 1227–1232.

Danckwardt,S. *et al.* (2008) 3' end mRNA processing: molecular mechanisms and implications for health and disease. *EMBO J.*, **27**, 482–98.

Deng,J.D.J. *et al.* (2009) ImageNet: a large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 248–255.

Deng,L. and Yu,D. (2014) Deep learning: methods and applications. *Found. Trends Signal Process.*, **7**, 197–387.

Dermitzakis,E.T. *et al.* (2003) Evolutionary discrimination of mammalian conserved non-genic sequences (CNGs). *Science*, **302**, 1033–1035.

Derti,A. *et al.* (2012) A quantitative atlas of polyadenylation in five mammals. *Genome Res.*, **22**, 1173–83.

Elkon,R. *et al.* (2013) Alternative cleavage and polyadenylation: extent, regulation and function. *Nat. Rev. Genet.*, **14**, 496–506.

Erhan,D. *et al.* (2009) Visualizing higher-layer features of a deep network. *Dept. IRO, Univ. Montréal, Tech. Rep*, **1341**.

Erhan,D. *et al.* (2010) Why does unsupervised pre-training help deep learning? *J. Mach. Learn. Res.*, **11**, 625–660.

Fairbrother,W.G. *et al.* (2002) Predictive identification of exonic splicing enhancers in human genes. *Science*, **297**, 1007–13.

Fawcett,T. (2006) An introduction to ROC analysis. *Pattern Recognit. Lett.*, **27**, 861–874.

Floudas,C.A. (2007) Computational methods in protein structure prediction. *Biotechnol. Bioeng.*, **97**, 207–213.

Gallego Romero,I. *et al.* (2014) RNA-seq: impact of RNA degradation on transcript quantification. *BMC Biol.*, **12**, 42.

Di Giammartino,D.C. *et al.* (2011) Mechanisms and consequences of alternative polyadenylation. *Mol. Cell*, **43**, 853–866.

Gibson,G. (2012) Rare and common variants: twenty arguments. *Nat. Rev. Genet.*, **13**, 135–145.

Gjuvsland,A.B. *et al.* (2013) Bridging the genotype-phenotype gap: what does it take? *J. Physiol.*, **591**, 2055–2066.

Glorot,X. *et al.* (2011) Deep sparse rectifier neural networks. *Proc. 14th Int. Conf. Artif. Intell. Stat.*, 315–320.

Glorot,X. and Bengio,Y. (2010) Understanding the difficulty of training deep feedforward neural networks. *Proc. 13th Int. Conf. Artif. Intell. Stat.*, **9**, 249–256.

Goldstein,D.B. *et al.* (2013) Sequencing studies in human genetics: design and interpretation. *Nat. Rev. Genet.*, **14**, 460–470.

Graves,A. *et al.* (2013) Speech recognition with deep recurrent neural networks. *Acoust. Speech Signal Process. (ICASSP), 2013 IEEE Int. Conf.*, 6645–6649.

Hafez,D. *et al.* (2013) Genome-wide identification and predictive modeling of tissue-specific alternative polyadenylation. *Bioinformatics*, **29**, i108-16.

Hanahan,D. and Weinberg,R.A. (2011) Hallmarks of cancer: the next generation. *Cell*, **144**, 646–74.

Harrow,J. *et al.* (2012) GENCODE: the reference human genome annotation for the ENCODE project. *Genome Res.*, **22**, 1760–1774.

van der Heijden,T. *et al.* (2012) Sequence-based prediction of single nucleosome positioning and genome-wide nucleosome occupancy. *Proc. Natl. Acad. Sci.*, **109**, E2514-22.

Hindorff,L.A. *et al.* (2009) Potential etiologic and functional implications of genome-wide association loci for human diseases and traits. *Proc. Natl. Acad. Sci.*, **106**, 9362–9367.

Hinton,G.E. *et al.* (2012) Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580*.

Hinton,G.E. and Salakhutdinov,R.R. (2006) Reducing the dimensionality of data with neural networks. *Science*, **313**, 504–7.

Ho,E.S. *et al.* (2013) A multispecies polyadenylation site model. *BMC Bioinformatics*, **14 Suppl 2**, S9.

Hofacker,I.L. (2003) Vienna RNA secondary structure server. *Nucleic Acids Res.*, **31**, 3429–31.

Hu,J. *et al.* (2005) Bioinformatic identification of candidate cis-regulatory elements involved in human mRNA polyadenylation. *RNA*, **11**, 1485–93.

Hua,Y. *et al.* (2011) Peripheral SMN restoration is essential for long-term rescue of a severe spinal muscular atrophy mouse model. *Nature*, **478**, 123–6.

Itoh,H. *et al.* (2004) Computational comparative analyses of alternative splicing regulation using full-length cDNA of various eukaryotes. *RNA*, **10**, 1005–18.

James Kent,W. *et al.* (2002) The human genome browser at UCSC. *Genome Res.*, **12**, 996–1006.

Ji,G. *et al.* (2015) Genome-wide identification and predictive modeling of polyadenylation sites in eukaryotes. *Brief. Bioinform.*, **16**, 304–313.

Johnson,D.S. *et al.* (2007) Genome-wide mapping of in vivo protein-DNA interactions. *Science*, **316**, 1497–502.

Johnson,R.C. *et al.* (2010) Accounting for multiple comparisons in a genome-wide association study (GWAS). *BMC Genomics*, **11**, 724.

Kakaradov,B. *et al.* (2012) Challenges in estimating percent inclusion of alternatively spliced junctions from RNA-seq data. *BMC Bioinformatics*, **13 Suppl 6**, S11.

Kalkatawi,M. *et al.* (2012) Dragon PolyA Spotter: predictor of poly(A) motifs within human genomic DNA sequences. *Bioinformatics*, **28**, 127–9.

Kaneko,S. and Manley,J.L. (2005) The mammalian RNA polymerase II C-terminal domain interacts with RNA to suppress transcription-coupled 3′ end formation. *Mol. Cell*, **20**, 91–103.

Katz,Y. *et al.* (2010) Analysis and design of RNA sequencing experiments for identifying isoform regulation. *Nat. Methods*, **7**, 1009–15.

Kelley,D.R. *et al.* (2016) Basset: learning the regulatory code of the accessible genome with deep convolutional neural networks. *Genome Res.*, **26**, 990–999.

Kent,W.J. *et al.* (2002) The human genome browser at UCSC. *Genome Res.*, **12**, 996–1006.

King,J.L. and Jukes,T.H. (1969) Non-Darwinian evolution. *Science*, **164**, 788–98.

de Klerk,E. and ‘t Hoen,P.A.C. (2015) Alternative mRNA transcription, processing, and translation: insights from RNA sequencing. *Trends Genet.*, **31**, 128–139.

Kole,R. *et al.* (2012) RNA therapeutics: beyond RNA interference and antisense oligonucleotides. *Nat. Rev. Drug Discov.*, **11**, 125–140.

Kooperberg,C. *et al.* (2010) Risk prediction using genome-wide association studies. *Genet. Epidemiol.*, **34**, 643–652.

Krizhevsky,A. *et al.* (2012) ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 1097–1105.

Kruppa,J. *et al.* (2012) Risk estimation and risk prediction using machine-learning methods. *Hum. Genet.*, **131**, 1639–1654.

Kumar,P. *et al.* (2009) Predicting the effects of coding non-synonymous variants on protein function using the SIFT algorithm. *Nat. Protoc.*, **4**, 1073–1081.

Laing,C. and Schlick,T. (2011) Computational approaches to RNA structure prediction, analysis, and design. *Curr. Opin. Struct. Biol.*, **21**, 306–318.

Lander,E. (2012) The Genome. *Nano-Lectures, Improbable Res.*

Lander,E.S. (2011) Initial impact of the sequencing of the human genome. *Nature*, **470**, 187–197.

Lander,E.S. *et al.* (2001) Initial sequencing and analysis of the human genome. *Nature*, **409**, 860–921.

Landrum,M.J. *et al.* (2014) ClinVar: public archive of relationships among sequence variation and human phenotype. *Nucleic Acids Res.*, **42**, D980–D985.

Le,Q. V. *et al.* (2013) Building high-level features using large scale unsupervised learning. *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, 8595–8598.

LeCun,Y. *et al.* (2015) Deep learning. *Nature*, **521**, 436–444.

LeCun,Y., Bottou,L., *et al.* (1998) Gradient-based learning applied to document recognition. *Proc. IEEE*, **86**, 2278–2323.

LeCun,Y., Cortes,C., *et al.* (1998) The MNIST database of handwritten digits.

LeCun,Y.A. *et al.* (1998) Efficient backprop. *Neural Networks: Tricks of the Trade*.

Ledford,H. (2015) End of cancer-genome project prompts rethink. *Nature*, **517**, 128–129.

Lee,J.Y. *et al.* (2007) PolyA_DB 2: mRNA polyadenylation sites in vertebrate genes. *Nucleic Acids Res.*, **35**, D165-8.

Lee,T.I. and Young,R. a. (2013) Transcriptional regulation and its misregulation in disease. *Cell*, **152**, 1237–1251.

Lehner,B. (2013) Genotype to phenotype: lessons from model organisms for human genetics. *Nat. Rev. Genet.*, **14**, 168–78.

Di lena,P. *et al.* (2012) Deep architectures for protein contact map prediction. *Bioinformatics*, **28**, 2449–57.

Leung,M.K.K. *et al.* (2014) Deep learning of the tissue-regulated splicing code. *Bioinformatics*, **30**, i121–i129.

Leung,M.K.K. *et al.* (2018) Inference of the human polyadenylation code. *Bioinformatics*, **34**, 2889–2898.

Leung,M.K.K. *et al.* (2016) Machine learning in genomic medicine: A review of computational problems and data sets. *Proc. IEEE*, **104**, 176–197.

Li,X. *et al.* (2010) Predicting in vivo binding sites of RNA-binding proteins using mRNA secondary structure. *RNA*, **16**, 1096–1107.

Lianoglou,S. *et al.* (2013) Ubiquitously transcribed genes use alternative polyadenylation to achieve tissue-specific expression. *Genes Dev.*, **27**, 2380–96.

Lin,Y. *et al.* (2012) An in-depth map of polyadenylation sites in cancer. *Nucleic Acids Res.*, **40**, 8460–71.

Lindblad-Toh,K. *et al.* (2011) A high-resolution map of human evolutionary constraint using 29 mammals. *Nature*, **478**, 476–482.

Lorenz,R. *et al.* (2011) ViennaRNA package 2.0. *Algorithms Mol. Biol.*, **6**, 26.

MacDonald,C.C. and McMahon,K.W. (2010) Tissue-specific mechanisms of alternative polyadenylation: testis, brain, and beyond. *Wiley Interdiscip. Rev. RNA*, **1**, 494–501.

Mahendran,A. and Vedaldi,A. (2015) Understanding deep image representations by inverting them. In, *Computer Vision and Pattern Recognition, Proc. of the IEEE Conf. on*.

Mali,P. *et al.* (2013) RNA-guided human genome engineering via Cas9. *Science*, **339**, 823–6.

Manning,K.S. and Cooper,T.A. (2017) The roles of RNA processing in translating genotype to phenotype. *Nat. Rev. Mol. Cell Biol.*, **18**, 102–114.

Marx,V. (2013) The big challenges of big data. *Nature*, **498**, 255–60.

Maston,G.A. *et al.* (2006) Transcriptional regulatory elements in the human genome. *Annu. Rev. Genomics Hum. Genet.*, **7**, 29–59.

Metzker,M.L. (2010) Sequencing technologies - the next generation. *Nat. Rev. Genet.*, **11**, 31–46.

Moffat,J.G. *et al.* (2014) Phenotypic screening in cancer drug discovery - past, present and future. *Nat. Rev. Drug Discov.*, **13**, 588–602.

Mortazavi,A. *et al.* (2008) Mapping and quantifying mammalian transcriptomes by RNA-Seq. *Nat. Methods*, **5**, 621–628.

Moult,J. *et al.* (2014) Critical assessment of methods of protein structure prediction (CASP) - round X. *Proteins*, **82**, 1–6.

Müller,S. *et al.* (2014) APADB: a database for alternative polyadenylation and microRNA regulation events. *Database (Oxford)*, **2014**, bau076.

Naryshkin,N.A. *et al.* (2014) SMN2 splicing modifiers improve motor function and longevity in mice with spinal muscular atrophy. *Science*, **345**, 688–93.

Ng,S.B. *et al.* (2009) Targeted capture and massively parallel sequencing of 12 human exomes. *Nature*, **461**, 272–276.

Ni,T. *et al.* (2013) Distinct polyadenylation landscapes of diverse human tissues revealed by a modified PA-seq strategy. *BMC Genomics*, **14**, 615.

Oshlack,A. and Wakefield,M.J. (2009) Transcript length bias in RNA-seq data confounds systems biology. *Biol. Direct*, **4**, 14.

Pan,Q. *et al.* (2008) Deep surveying of alternative splicing complexity in the human transcriptome by high-throughput sequencing. *Nat. Genet.*, **40**, 1413–1415.

Poduri,A. *et al.* (2013) Somatic mutation, genomic variation, and neurological disease. *Science*, **341**, 43–51.

Pollard,K.S. *et al.* (2010) Detection of nonneutral substitution rates on mammalian phylogenies. *Genome Res.*, **20**, 110–121.

Pritchard,J.K. and Przeworski,M. (2001) Linkage disequilibrium in humans: models and data. *Am. J. Hum. Genet.*, **69**, 1–14.

Proudfoot,N.J. (2011) Ending the message: poly(A) signals then and now. *Genes Dev.*, **25**, 1770–82.

Pruitt,K.D. *et al.* (2005) NCBI Reference Sequence (RefSeq): a curated non-redundant sequence database of genomes, transcripts and proteins. *Nucleic Acids Res.*, **33**, D501-4.

Rampasek,L. and Goldenberg,A. (2016) TensorFlow: Biology's Gateway to Deep Learning? *Cell Syst.*, **2**, 12–14.

Ray,D. *et al.* (2009) Rapid and systematic analysis of the RNA recognition specificities of RNA-binding proteins. *Nat. Biotechnol.*, **27**, 667–670.

Ren,B. *et al.* (2000) Genome-wide location and function of DNA binding proteins. *Science*, **290**, 2306–2309.

Richards,S. *et al.* (2015) Standards and guidelines for the interpretation of sequence variants: a joint consensus recommendation of the American College of Medical Genetics and Genomics and the Association for Molecular Pathology. *Genet. Med.*, **17**, 405–423.

Ritchie,M.D. *et al.* (2015) Methods of integrating data to uncover genotype-phenotype interactions. *Nat. Publ. Gr.*, **16**, 85–97.

Robinson,E.B. *et al.* (2014) Response to 'Predicting the diagnosis of autism spectrum disorder using gene pathway analysis'. *Mol. Psychiatry*, **19**, 859–61.

Rubin,M.A. (2015) Make precision medicine work for cancer care. *Nature*, **520**, 290–1.

Rudin,C. and Wagstaff,K.L. (2013) Machine learning for science and society. *Mach. Learn.*, **95**, 1–9.

Rund,D. *et al.* (1992) Two mutations in the beta-globin polyadenylylation signal reveal extended transcripts and new RNA polyadenylylation sites. *Proc. Natl. Acad. Sci.*, **89**, 4324–8.

Saeys,Y. *et al.* (2007) Translation initiation site prediction on a genomic scale: beauty in simplicity. *Bioinformatics*, **23**, 418–423.

Saito,T. and Rehmsmeier,M. (2015) The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PLoS One*, **10**, 1–21.

Schena,M. *et al.* (1995) Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science*, **270**, 467–470.

Shaw,G. and Kamen,R. (1986) A conserved AU sequence from the 3' untranslated region of GM-CSF mRNA mediates selective mRNA degradation. *Cell*, **46**, 659–67.

Shi,Y. (2012) Alternative polyadenylation: new insights from global analyses. *RNA*, **18**, 2105–17.

Siepel,A. *et al.* (2005) Evolutionarily conserved elements in vertebrate, insect, worm, and yeast genomes. *Genome Res.*, **15**, 1034–1050.

Silver,D. and Mercer,R. (1996) The parallel transfer of task knowledge using dynamic learning rates based on a measure of relatedness. *Connect. Sci. Spec. Issue Transf. Inductive Syst.*, **8**, 277–294.

Simonyan,K. *et al.* (2014) Deep inside convolutional networks: visualising image classification models and saliency maps. In, *Proc. of the Int. Conf. on Learn. Representations*.

Skafidas,E. *et al.* (2014) Predicting the diagnosis of autism spectrum disorder using gene pathway analysis. *Mol. Psychiatry*, **19**, 504–510.

Snoek,J. *et al.* (2012) Practical bayesian optimization of machine learning algorithms. *arXiv:1206.2944*.

Sonnenburg,S. *et al.* (2007) Accurate splice site prediction using support vector machines. *BMC Bioinformatics*, **8 Suppl 10**, S7.

Strachan,T. and Read,A. (2010) Human molecular genetics. *Garland Science*.

The International HapMap Consortium (2005) A haplotype map of the human genome. *Nature*, **437**, 1299–1320.

Tian,B. *et al.* (2005) A large-scale analysis of mRNA polyadenylation of human and mouse genes. *Nucleic Acids Res.*, **33**, 201–12.

Tian,B. and Graber,J.H. (2012) Signals for pre-mRNA cleavage and polyadenylation. *Wiley Interdiscip. Rev. RNA*, **3**, 385–96.

Tian,B. and Manley,J.L. (2016) Alternative polyadenylation of mRNA precursors. *Nat. Rev. Mol. Cell Biol.*, **18**, 18–30.

Tibshirani,R. (1994) Regression shrinkage and selection via the lasso. *J. R. Stat. Soc. Ser. B*, **73**, 273–282.

Tieleman,T. (2010) Gnumpy: an easy way to use GPU boards in Python. *Univ. Toronto Tech. Rep.*

Touw,W.G. *et al.* (2013) Data mining in the life sciences with random forest: a walk in the park or lost in the jungle? *Brief. Bioinform.*, **14**, 315–326.

Troyanskaya,O.G. (2014) Deep supervised and convolutional generative stochastic network for protein secondary structure prediction. *Proc. 31st Int. Conf. Mach. Learn.*, **32**, 745–753.

Tuskan,G.A. *et al.* (2006) The genome of black cottonwood, Populus trichocarpa (Torr. & Gray). *Science*, **313**, 1596–604.

Ule,J. *et al.* (2006) An RNA map predicting Nova-dependent splicing regulation. *Nature*, **444**, 580–586.

Ureta-Vidal,A. *et al.* (2003) Comparative genomics: genome-wide analysis in metazoan eukaryotes. *Nat. Rev. Genet.*, **4**, 251–262.

Vickers,T. a *et al.* (2001) Fully modified 2' MOE oligonucleotides redirect polyadenylation. *Nucleic Acids Res.*, **29**, 1293–9.

Visscher,P.M. *et al.* (2012) Five years of GWAS discovery. *Am. J. Hum. Genet.*, **90**, 7–24.

Wan,Y. *et al.* (2011) Understanding the transcriptome through RNA structure. *Nat. Rev. Genet.*, **12**, 641–655.

Wang,Z. *et al.* (2009) RNA-Seq: a revolutionary tool for transcriptomics. *Nat. Rev. Genet.*, **10**, 57–63.

Wang,Z. *et al.* (2004) Systematic identification and analysis of exonic splicing silencers. *Cell*, **119**, 831–45.

Wang,Z. and Burge,C.B. (2008) Splicing regulation: from a parts list of regulatory elements to an integrated splicing code. *RNA*, **14**, 802–13.

Watson,I.R. *et al.* (2013) Emerging patterns of somatic mutations in cancer. *Nat. Rev. Genet.*, **14**, 703–18.

Watson,J.D. and Crick,F.H.C. (1953) Molecular structure of nucleic aids: a structure for deoxyribose nucleic acid. *Nature*, **171**, 737–738.

Weng,L. *et al.* (2016) Poly(A) code analyses reveal key determinants for tissue-specific mRNA alternative polyadenylation. *RNA*, **22**, 813–21.

Xie,B. *et al.* (2013) Poly(A) motif prediction using spectral latent features from human DNA sequences. *Bioinformatics*, **29**, 316–325.

Xiong,H. *et al.* (2011) Bayesian prediction of tissue-regulated splicing using RNA sequence and cellular context. *Bioinformatics*, **27**, 2554–2562.

Xiong,H.Y. *et al.* (2016) Probabilistic estimation of short sequence expression using RNA-Seq data and the positional bootstrap. *bioXiv:046474*.

Xiong,H.Y. *et al.* (2015) The human splicing code reveals new insights into the genetic determinants of disease. *Science*, **347**, 1254806.

Yandell,M. and Ence,D. (2012) A beginner's guide to eukaryotic genome annotation. *Nat. Rev. Genet.*, **13**, 329–342.

Yates,A. *et al.* (2016) Ensembl 2016. *Nucleic Acids Res.*, **44**, D710–D716.

Yeo,G.W. *et al.* (2007) Discovery and analysis of evolutionarily conserved intronic splicing regulatory elements. *PLoS Genet.*, **3**, e85.

Yip,K.Y. *et al.* (2013) Machine learning and genome annotation: a match meant to be? *Genome Biol.*, **14**, 205.

Zeiler,M.D. and Fergus,R. (2013) Visualizing and understanding convolutional networks. *Comput. Vision–ECCV 2014*, 818–833.

Zhang,H. *et al.* (2005) PolyA_DB: a database for mammalian mRNA polyadenylation. *Nucleic Acids Res.*, **33**, D116-20.

Zhou,J. and Troyanskaya,O.G. (2015) Predicting effects of noncoding variants with deep learning-based sequence model. *Nat. Methods*, **12**, 931–4.

Appendix A: Splicing Code Features

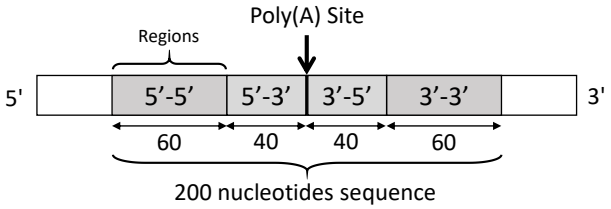
Table of features divided into 55 groups based on what they represent. Detailed definitions of these features can be found in the supplementary information from (Barash *et al.*, 2010) and (Barash *et al.*, 2013).

Group #	Name	Brief Description	Type	# of Features
01	short-seq-1mer	Frequency of nucleotide patterns of different lengths (1 to 3) in regions C1, A, C2, and their junctions.	real (0-1)	28
02	short-seq-2mer			112
03	short-seq-3mer			320
04	translatable-C1	Describes whether exons can be translated without a stop codon in one of three possible reading frames. For example, C1A means the exons of interest are C1 + A.	binary	1
05	translatable-C1A			1
06	translatable-C1AC2			1
07	translatable-C1C2			1
08	mean-con-score-AI2	Mean conservation score derived from the phastCons track (Siepel *et al.*, 2005) for mouse assembly mm8 in the UCSC Genome Browser.	real (0-1)	1
09	mean-con-score-I1A			1
10	mean-con-score-I2C2			1
11	mean-con-score-C1I1			1
12	log-length	Log base 10 lengths of regions C1, I1, A, I2, and C2.	real	5
13	log-length-ratio	Log base 10 length ratios of exons A to I1, A to I2, and I1 to I2.	real	3
14	acceptor-site-strength	Strength of acceptor and donor sites in I1 and I2, from (Itoh *et al.*, 2004).	real	2
15	donor-site-strength			2
16	frameshift-exonA	Whether exon A introduces a frameshift.	binary	1
17	rna-sec-struct	Score describing the probability of local stem-loop structure, computed using RNAfold (Hofacker, 2003).	real (0-1)	32
18	5mer-motif-down	Counts of motif clusters of different lengths (5 to 7) weighted by conservation upstream and downstream of the alternative exon from (Yeo *et al.*, 2007).	real	54
19	6mer-motif-down			76
20	7mer-motif-down			28
21	5mer-motif-up			49
22	6mer-motif-up			78
23	7mer-motif-up			29
24	ese-ess-A	Counts of exonic splicing enhancers (Fairbrother *et al.*, 2002) and silencers (Wang *et al.*, 2004).	real	4
25	ese-ess-C1			4
26	ese-ess-C2			4
27	pssm-SC35	PSSM scores of SC35 splicing regulator protein.	real	5
28	pssm-ASF-SF2	PSSM scores of ASF/SF2 splicing regulator protein.		5
29	pssm-SRp40	PSSM scores of SRp40 splicing regulator protein.		10
30	nucleosome-position	Nucleosome positioning from (van der Heijden *et al.*, 2012).	real	4
31	PTB	Counts of phosphotyrosine-binding domain motif.	real	50
32	Nova-counts	Counts of Nova motif YCAY.	integer	27
33	Nova-cluster	Nova cluster score (Ule *et al.*, 2006).	real	8
34	T-rich	Counts of motif with and without weighting by conservation.	real	24
35	G-rich			8
36	UG-rich			16
37	GU-rich			32
38	Fox	Counts of RNA binding protein motifs with and without weighting by conservation.	real	24
39	Quak			8
40	SC35			22
41	SRm160			11
42	SRrp20/30/38/40/55/75			77
43	CELF-like			2
44	CUGBP			16
45	MBNL			24
46	TRA2-alpha			22
47	TRA2-beta			22
48	hnRNP-A			44
49	hnRNP-H			22
50	hnRNP-G			22
51	9G8			22
52	ASF/SF2			11
53	Sugnet			2
54	alt-AG-pos	Position of the alternative AG and GT position.	integer	2
55	Alu	Counts of ALU repeats.	integer	12

C1 and *C2* denote the flanking constitutive exons ; *A* denotes the alternative exon ; *I1* denotes the intron between *C1* and *A* ; *I2* denotes the intron between *A* and *C2*

Appendix B: Polyadenylation Code Features

Feature Description of the Feature-Net



Regions around a PAS (Hu *et al.*, 2005) where features are extracted.

Feature Group*	**Regions**	**# Features**
Poly(A) Signal[1]	5'-5' 5'-3'	26 26
AUE Elements[2]	5'-5'	12
CUE Elements[2]	5'-3'	2
CDE Elements[2]	3'-5'	15
ADE Elements[2]	3'-3'	12
RBP Motifs[3]	All 4	18 x 4
1-mers	All 4	4 x 4
2-mers	All 4	16 x 4
3-mers	All 4	64 x 4
4-mers	All 4	248 x 4
Mean and Max Nucleosome Occupancy[4]	5' of PAS 3' of PAS Full Seq	12
Position	---	1

* redundant features that are present in multiple feature groups are removed

[1]Polyadenylation Signals (Tian *et al.*, 2005; Beaudoing *et al.*, 2000; Ni *et al.*, 2013; Derti *et al.*, 2012):

AATAAA, ATTAAA, TATAAA, AGTAAA, AAGAAA, AATATA, AATACA, CATAAA, GATAAA, AATGAA, TTTAAA, ACTAAA, AATAGA, AAAAAG, AAAATA, GGGGCT, AAAAAA, ATAAAA, AAATAA, ATAAAT, TTTTTT, ATAAAG, TAAAAA, CAATAA, TAATAA, ATAAAC

[2]Cis-Elements, from Table 1 in (Hu *et al.*, 2005):

Auxiliary Upstream Elements (AUE):	GGGGAG, GUGGGG, GGGUGG, UUUGUA, GUAUUU, CUGUGU, UAUAUA, AUAUAU, UUUAUA, UGUAUA, AUGUAU, UGUAUU
Core Upstream Elements (CUE):	UAUUUU, UGUUUU
Core Downstream Elements (CDE):	CCUCCC, CUCCCC, CACCCC, CCCGCC, CCCCGC, CCCGCG, GGUGGG, GGCUGG, GGGUGG, GGGCAG, GGCCAG, GGGGCC, GGGAGG, GGAGGG, GGGGAG
Auxiliary Downstream Elements (ADE):	GUGUCU, CUGCCU, UGUCUC, UUAUUU, UUUCUU, UGUUUU, UGUGUG, GUGUGU, CUGUGU, CUGGGG, UGUCUG, GUCUGU

[3]RNA Binding Protein Motifs, in IUPAC notation (Cornish-Bowden, 1985):

CPEB1: UUUUAU, **hnRNP-H1**: GGGAGG, **hnRNP-H2**: GGAGGG, **MBNL_v1**: GCUUGC, **MBNL_v2**: YGCY, **MBNL_v3**: YGCUKY, **PABPN1**: ARAAGA, **PTBP1**: UUUUCU, **NOVA**: UCAY, **PCBP1**: CCWWHCC, **PCBP2**: CCYYCCH, **ESRP2**: UGGGRAD, **hnRNP-F/H_v1**: GGGA, **hnRNP-F/H_v2**: UKKGGK, **hnRNP-F/H_v3**: GGSKG, **CFIm**: UGUA, **CstF-64**: UGUGU, **SRSF1**: GAAGAA

where *R* is G or A, *Y* is T or C, *M* is A or C, *K* is G or T, *S* is G or C, *W* is A or T, *H* is A or C or T, *B* is G or T or C, *V* is G or C or A, *D* is G or A or T, and *N* is any

[4]Nucleosome Occupancy, computed using Equation 1 in (van der Heijden *et al.*, 2012) with the suggested parameters in the paper. The nucleosome occupancy is a measure of the probability that a DNA region is wrapped around a histone octamer.