Université de Montréal

Modeling Meiotic Recombination Hotspots using Deep Learning

Par

Emad Takla

Département de Biochimie et Médecine Moléculaire

Faculté de Médecine

Mémoire présenté en vue de l'obtention du grade de M.Sc.

en Bio-Informatique option Recherche - Bio-informatique statistique et apprentissage machine

2021

© Emad Takla, 2021

Université de Montréal

Unité académique: Département de Biochimie et Médecine Moléculaire

Faculté de Médecine

Ce mémoire intitulé

Modeling Meiotic Recombination Hotspots using Deep Learning

Présenté par

Emad Takla

A été évalué(e) par un jury composé des personnes suivantes

Guillaume Dumas
Président-rapporteur

Julie Hussin
Directeur de recherche

Yoshiaki Tanaka
Membre du jury

Résumé

La recombinaison méiotique joue un rôle essentiel dans la ségrégation des chromosomes pendant la méiose et dans la création de nouvelles combinaisons du matériel génétique des espèces. Ses effets cause une déviation du principe de l'assortiment indépendant de Mendel; cependant, les mécanismes moléculaires impliqués restent partiellement incompris jusqu'à aujourd'hui. Il s'agit d'un processus hautement régulé et de nombreuses protéines sont impliquées dans son contrôle, dirigeant la recombinaison méiotique dans des régions génomiques de 1 à 2 kilobases appelées « hotspots ». Au cours des dernières années, l'apprentissage profond a été appliqué avec succès à la classification des séquences génomiques. Dans ce travail, nous appliquons l'apprentissage profond aux séquences d'ADN humain afin de prédire si une région spécifique d'ADN est un hotspot de recombinaison méiotique ou non. Nous avons appliqué des réseaux de neurones convolutifs sur un ensemble de données décrivant les hotspots de quatre individus non-apparentés, atteignant une exactitude de plus de 88 % avec une précision et un rappel supérieur à 90 % pour les meilleurs modèles. Nous explorons l'impact de différentes tailles de séquences d'entrée, les stratégies de séparation des jeux d'entraînement/validation et l'utilité de montrer au modèle les coordonnées génomiques de la séquence d'entrée. Nous avons exploré différentes manières de construire les motifs appris par le réseau et comment ils peuvent être liés aux méthodes classiques de construction de matrices position-poids, et nous avons pu déduire des connaissances biologiques pertinentes découvertes par le réseau. Nous avons également développé un outil pour visualiser les différents modèles afin d'aider à interpréter les différents aspects du modèle. Dans l'ensemble, nos travaux montrent la capacité des méthodes d'apprentissage profond à étudier la recombinaison méiotique à partir de données génomiques.

Mots-clés : Méiose, recombinaison méiotique, apprentissage profond, PRDM9, extraction des motifs.

Abstract

Meiotic recombination plays a critical role in the proper segregation of chromosomes during meiosis and in forming new combinations of genetic material within sexually-reproducing species. For a long time, its side effects were observed as a deviation from the Mendel's principle of independent assortment; however, its molecular mechanisms remain only partially understood until today. We know that it is a highly regulated process and that many molecules are involved in this tight control, resulting in directing meiotic recombination into 1-2 kilobase genomic pairs regions called hotspots. During the past few years, deep learning was successfully applied to the classification of genomic sequences. In this work, we apply deep learning to DNA sequences in order to predict if a specific stretch of DNA is a meiotic recombination hotspot or not. We applied convolution neural networks on a dataset describing the hotspots of four unrelated male individuals, achieving an accuracy of over 88% with precision and recall above 90% for the best models. We explored the impact of different input sequence lengths, train/validation split strategies and showing the model the genomic coordinates of the input sequence. We explored different ways to construct the learnt motifs by the network and how they can relate to the classical methods of constructing position-weight-matrices, and we were able to infer relevant biological knowledge uncovered by the network. We also developed a tool for visualizing the different models output in order to help digest the different aspects of the model. Overall, our work shows the ability for deep learning methods to study meiotic recombination from genomic data.

Keywords: meiosis, meiotic recombination, deep learning, PRDM9, motif extraction

Acknowledgements

I am truly grateful for all the people who helped me reach this point of master's work. I thank prof Julie Hussin for the liberty given to me to explore, for the encouragement to put my thoughts while guiding me all along, and for the patience until the last moment. And to Raphael for all the countless discussions we had about the project. It was a true privilege to work with all the MHI-OMICs team at the Institut de Cardiologie de Montreal, discussing with them frequently the project and getting their feedback. I thank Elaine for all her countless reminders and helping with all the administrative aspects. I am thankful for the Dataperformers team who truly helped me to go along this journey, offering to me a lot of flexibility. Finally, my family and especially my parents and my grandmother. I am sure that this journey would not have happened without you. I love you!

Table of Contents

Table of Tables

Table of Figures

List of Abbreviations

.bed	Browser Extensible Data file
.tsv	Tab separated value file
A	Adenine
AI	Artificial Intelligence
bp, bps, kbps	Basepair, basepairs, kilo basepairs
C	Cytosine
ChIP, ChIP-seq	Chromatin immunoprecipitation, Chromatin immunoprecipitation sequencing
CNN	Convolution neural network
DMC1	DNA meiotic recombinase 1
DNA	Deoxyribonucleic acid
DSB	Double Strand Breaks
G	Guanine
GRU	Gated Recurrent Unit
LD	Linkage Disequilibrium
LSTM	Long Short-Term Memory
LTR	Long Terminal Repeat
ML	Machine Learning
NDR	Nucleosome Depleted Region
PRDM9	PR domain zinc finger protein 9

PWM	Position Weight Matrix
RNN	Recurrent Neural Network
SNV	Single-nucleotide variant
ssDNA	Single stranded Deoxyribonucleic acid
SSDS	Single stranded Deoxyribonucleic acid sequencing
SVM	Support Vector Machines
T	Thymine
ZnF	Zinc Finger

Chapter 1 Introduction

Meiotic recombination is one of nature's tricks to generate genetic diversity and ensures proper segregation of chromosomes in sexually reproducing organisms. It was observed as a deviation of Mendelian genetics, but its underlying processes remained elusive until higher resolution methods working directly with DNA sequences became available. Directly working with DNA sequences has been traditionally studied using string patterns matching algorithms. However, in recent years, applying deep learning to DNA sequences gave similar results to the pure algorithmic approach, opening new doors to study these problems from a new angle.

1.1 Biology

1.1.1 Role of DNA in Life

Biology is the study of life. Defining what is life and what is a living being may seem like a trivial question at a first glance. However, detailing a complete definition for what is life is a hard task. The National Aeronautics and Space Administration (NASA), for its mission to explore extraterrestrial existence of life, has defined life to be "a self-sustaining chemical system capable of Darwinian evolution" (*NASA Astrobiology*, n.d.). This Darwinian evolution mentioned in the definition explains that organisms adapt through selection of inherited genetic variation to increase its ability to survive and reproduce. Large sized biological molecules (macromolecules) called nucleic acids are the storage medium of genetic material and thus play a central role in this Darwinian evolution. In many organisms, the nucleic acid carrying this genetic information is the deoxyribonucleic acid (DNA).

1.1.2 DNA, Proteins and Chromosomes

Polymers are long chains of macromolecules. DNA is a polymer, made of building blocks called nucleotides. The main 4 nucleotides (also referred to as "bases" and "base-pairs" (bp))

are adenine (A), guanine (G), cytosine (C) and thymine (T) (*Base Pair*, n.d.). These bases rest on a sugar-phosphate (De Bont & van Larebeke, 2004) hate backbone to form a single-strand of DNA (ssDNA). A ssDNA pairs with another strand made of the complement sequence to form the double stranded DNA helix (Watson & Crick, 1953). Base pairing happens as follows, Adenine pairs with Thymine and Cytosine pairs with Guanine. The DNA backbone is a fragile structure prone to damage. Backbone damage can happen for many reasons. For example, environmental reasons include ionizing radiation (Reisz et al., 2014) and viruses (Weitzman & Fradet-Turcotte, 2018), and endogenous factors like hydrolysis (De Bont & van Larebeke, 2004). In another example, UV energy can cause two adjacent base-pairs to bond covalently, forming a dimer. This dimer distorts the double helix structure leading to DSBs (Douki et al., 2003). Such a breakage can affect only one strand of the double helix (single strand break) or it can affect both strands (Double-Strand Breaks, or DSB).

The 4 bases of DNA can exist under different chemical conditions. For example, cytosine and adenine can get methylated (Bird, 2002) (a methyl group gets attached to the DNA base). DNA base methylation plays an important role in the functioning of DNA, as well as being a source of mutation in the sequence. One pathway of such mutation is that 5-methyl-cytosine undergoes spontaneous deamination, which converts it to thymine. Unmethylated cytosine deamination creates a uracil, which is a foreign base for DNA and therefore corrected and no mutation happens (Lander et al., 2001).

If the distribution of the 4 nucleotides was totally random, we would model their distribution as a uniform distribution. For example, we would expect each base pair to be represent roughly 25% of any sequence with sufficient length (sample size). We would also expect that two consecutive base pairs to have the same nucleotides 6.25% of the time. Sequence homology is when DNA sequences share an evolutionary source. We can identify them by a high similarity of their sequence that cannot happen due to chance alone. Sequence homology can happen in many ways, including speciation (orthologs), duplication (paralogs) or gene transfer (xenologs).

The genetic information is organized into different regions, and each region has a certain functional role and work with each other to accomplish the vital operations. For example,

there are regions encoding for end products (usually a protein) called genes. Other functional parts of the genome control the genes, like promoters and enhancers regions. Other genomic regions of interest are regions where the statistical distribution of base pairs is not uniform. For example, there are regions unusually rich in C-G pairs called CpG islands (Lander et al., 2001), and we usually find these near promoter regions (Saxonov et al., 2006). Some DNA regions exhibit high mutability and mobility. For example, regions with repeated motifs called long tandem repeats (LTR) have a high mutation rate. Depending on the length of the LTR, they can sometimes be referred to as microsatellite and minisatellite DNA. Other interesting genomic region are regions that can migrate on the genome called retrotransposons. These retrotransposons contain repeats as well and are classified into families. One example being the low-repeat-family THE1 (Smit, 1993).

1.1.2.1 Proteins and Protein-DNA Interaction

Another important family of macromolecules for life are proteins. Proteins are a sequence of amino acids (a.a.) chained together, forming a polymer. Such polymers take a 3-dimensional conformation, and such structures dictate the function of the protein (Janin & Wodak, 1983). The overall structure has different stable parts, called domains (Janin & Wodak, 1983). Zinc fingers (ZnF) (Klug & Rhodes, 1987) are one important protein domain. Intramolecular forces between a ZnF and a stretch of DNA can attract and attach a ZnF to a specific sequence of DNA, making them an important tool to recognize DNA regions by cellular machinery (Klug, 2010). The more matching a DNA sequence to the ZnF, the higher the binding affinity (Klug, 2010). We can use such a relationship to predict binding affinity of a DNA sequence to a ZnF by knowing the exact order of amino-acids in the ZnF array (Luscombe et al., 2001). Proteins that recognize specific regions of DNA play an important role in regulating the functioning of DNA. One example is transcription factors, which are proteins that regulate the activity of genes. But protein recognition of a stretch of DNA is not the only way for a protein to interact with a DNA. There are non-specific DNA-protein interactions where the protein interacts with the DNA regardless of its sequence. Such non-specific interactions play an important role in keeping DNA in order. One such DNA-protein union is the chromosome, where histones, a protein complex, group together to form a basic DNA packaging called nucleosome and bind to the DNA (Luger et al., 1997). This regulates DNA's 3-dimensional

structure and its spatial position inside the nucleus. Some organisms have the total of their genetic information packaged in only one chromosome (Crosland & Crozier, 1986), while others have theirs stored on multiple separate chromosomes.

1.1.2.2 Ploidy

In sexually reproducing organisms, each chromosome has multiple copies coming from the parents. Ploidy is the number of copies of each chromosome (Otto, 2007). Humans are a diploid organism, and therefore in most cells these chromosomes are in pairs (46 in total). The only exception cell where there is only one copy of the 23 chromosomes are the sex cells (ovules in female humans and sperms in male humans). Aneuploidy is the abnormal count of chromosomes in cells (Otto, 2007). It is important to establish the nomenclature in the right context. Sister chromatids are copies of the same base chromosome and are attached through their centromeres. Sister chromosomes are the homologous chromosomes. Tetrads are sister chromatids attached to their homologous chromosome, and a bivalent is a chromosome within a tetrad.

1.1.3 Effect of DSBs on Cell

As explained above, the DNA backbone can break leading to DSBs. DSBs can lead to detrimental results for the organism. For example, the loose part of the chromosome can get erroneously fused on a different chromosome (translocation). One known translocation from chromosome 9 to chromosome 22 causes over one form of leukemia in humans (Kang et al., 2016). Therefore, cells have developed multiple lines of defense to fix DSBs. Some examples of such DSB mending mechanisms are non-homologous end joining and homologous recombination (Sung & Klein, 2006). The homologous recombination process, in principle, happens when two identical or nearly identical stretches of DNA are attracted to each other and exchange genetic information. In fixing a DSB, homologous recombination depends on using the homologous chromosome (and sometimes the sister chromatid) (Shrivastav et al., 2008) as a template to fix the DSB on the affected chromosome. Homologous recombination is a definition of a general process, but the process details can take multiple paths to fix the DSB. Examples of these paths include single-strand annealing

(SSA), break-induced replication (BIR) and gene conversion (with or without crossover). A break-induced replication process involves important processes (Sung & Klein, 2006).

When a breakage happens, the first step of repair is that the 5' DNA stretch at the breaking gets degraded, leaving a stretch of 3' DNA exposed. Multiple proteins get involved in the process after. For example, a process called strand-invasion involves Rad51 (and DMC1 during meiosis as well, (Figure 1.1 b and c)) proteins recruited over the single-stranded DNA. These proteins guide the ssDNA towards another stretch of a nearly identical double-stranded DNA sequence. Searching for homology is usually very efficient, and some work has shown that matching the sequence can happen within 30 minutes (Hicks et al., 2011). Once the stretch of ssDNA is in the vicinity of the nearly identical homolog, it gets inserted into this homologous region, pairing with the strand containing the complementary sequence. The other strand of the homologous sequence forms a displacement loop (D-loop). Endonuclease proteins finally resolve a Holliday junction. This resolution can happen in two different ways. In one way, non-crossover resolution would give back each strand to its original pairing one. The other resolution would cause a switch between the strands leading to what is called a crossover. Crossovers result in the exchange of stretches of DNA between the two chromosomes, leading to a shuffle of the genetic information.

1.1.3.1 Chromosome accessibility

The nucleosome is the starting point to bend the DNA double helix into an organized structure called chromatin. Nucleosomes are not evenly distributed throughout the chromosomes, with regions having no nucleosomes called nucleosome-depleted-regions (NDR) (Lee et al., 2004). Because nucleosomes help package the DNA into a more compact form, NDR regions are more accessible to cellular machinery. NDR is not the only way to make a stretch of DNA more accessible to the cellular machinery. Nucleosome rich regions can become more open as well: this histone protein-complex can be chemically modified during its lifetime, and such histone modifications will change the chromatin structure, in a process called "chromatin remodeling". One example of such modifications, referred to as H3K4me3, triggers a chromatin remodeling, allowing access to the stretch of DNA by other cellular mechanisms.

1.1.3.2 Studying Protein-DNA Interaction: CHiP-SEQ

For the importance of the role protein-DNA interaction plays in genetics, studying such interaction such as DNA sequence binding affinity to a certain protein became important. An approach called chromatin immunoprecipitation (ChIP) proved to be very useful. We first create a solution containing fragmented DNA and the protein. Then, we add antibodies (Y-shaped proteins with high binding capabilities to other proteins) with high binding affinity to the protein of interest to the solution. Then the protein-antigen is precipitated in the solution. The precipitated content will have the DNA sequences bound to the protein, and therefore this precipitation is filtered for its DNA content. Finally, this DNA is sequenced. ChIP-seq is a method where massive parallel sequencing is used to sequence ChIP results (Johnson et al., 2007; *Whole-Genome Chromatin IP Sequencing (ChIP-Seq)*, n.d.) and this method has become the main approach to study DNA-protein interaction. Histone modifications, such as H3K4me3, can also be studied using ChIP-seq.

1.1.4 Cell Division

Over the life span of an organism, its cells will go through different stages (*Cell Division*, n.d.). Most of the time, they are in a state called interphase. At this stage, a cell would grow, and among other things, it would copy each chromosome. There may be some confusion in the nomenclatures when a chromosome has replicas in a cell, since we have two equivalent homologous chromosomes and each of these has an exact copy. So, a human cell ready for division has 92 chromosomes, of which there are the expected 46 sister chromosomes, each having its own sister chromatid.

After interphase, the cell will undergo cell division, a process by which they give rise to new daughter cells. It gives rise to new cells that would have their genetic code based on the parent cell's one. In eukaryotes, this process can happen in either two ways: mitosis or meiosis, each having a difference in purpose and the number of chromosomes in the resulting daughter cell. In mitosis, the purpose is to create two identical cells to the parent cell, while in meiosis the purpose is to create daughter cells with a reduced number of chromosomes (and hence the name meiosis, from Greek, means reducing). They both go through essentially the same steps. These steps are prophase, metaphase, anaphase,

telophase and, finally, cytokinesis. At prophase, the chromosomes, which are already duplicated into two sister chromatids, condense and microtubules become visible. Also, homologous chromosomes must pair up in the case of meiosis. At metaphase, the homologous chromosomes line up at the center of the nucleus, with each one facing an opposite pole of the cell. At anaphase, the spindles pull away each daughter's cell genetic material. In mitosis, the spindles separate the chromatids, while in meiosis, they pull the homologous chromosomes towards the poles of the cell. At telophase, two new nuclei form around each set of chromosomes at the pole. Finally, cytokinesis is when the cytoplasm splits into two cells, each having one nucleus formed at telophase. In meiosis (Figure 1.1 a), these steps happen two times in succession, and therefore are numbered. So in meiosis, the cell division steps are: prophase I, metaphase I, anaphase I, telophase I, cytokinesis I, prophase II, metaphase II, anaphase II, telophase II and finally cytokinesis II. Prophase I stage itself is divided into five stages: leptotene, zygotene, pachytene, diplotene and diakinesis.

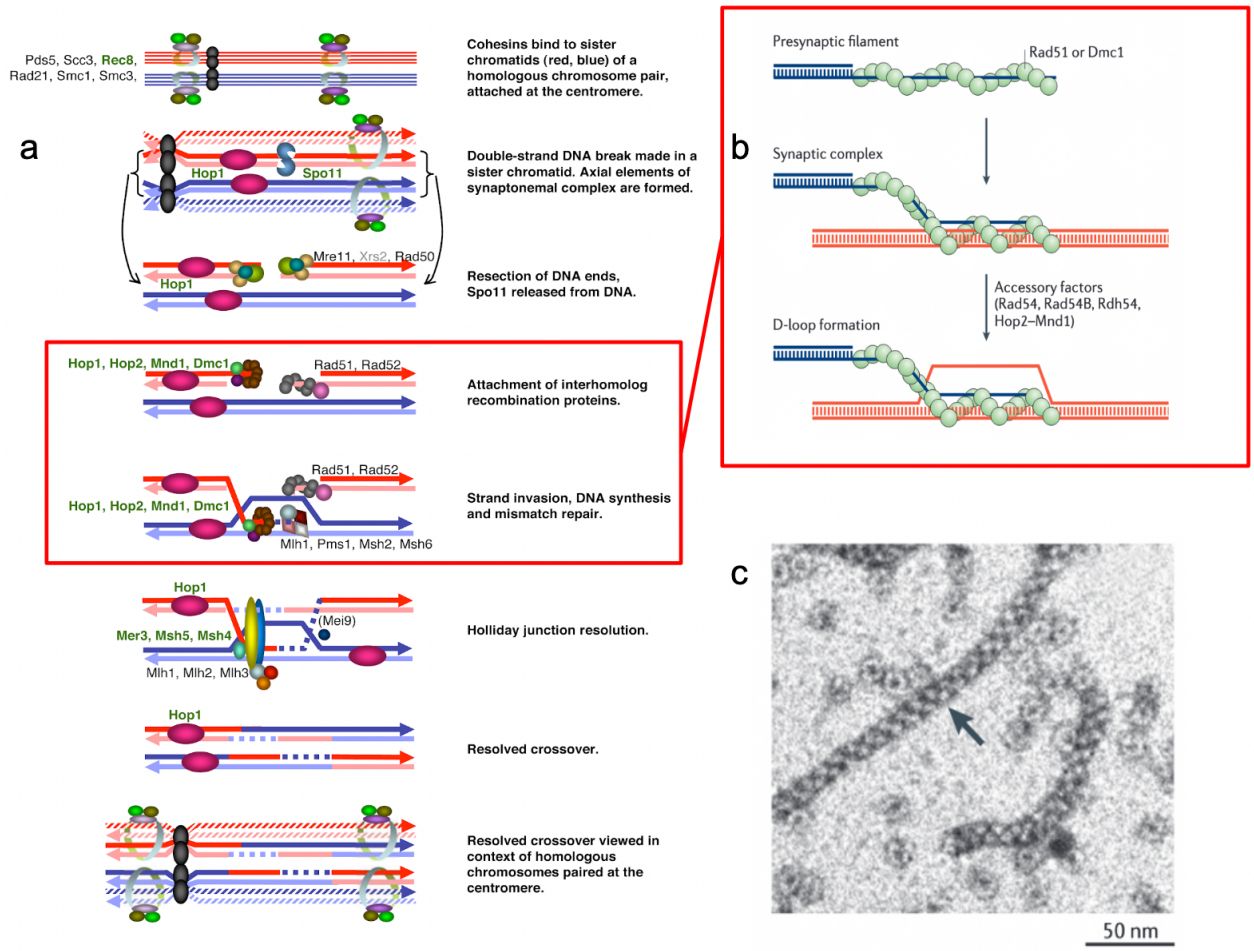

Figure 1.1 Meiosis and the role of DMC1. (a) Shows the stages of meiosis (Malik et al., 2007) (b) DMC1 coils around SSDS guiding the search for homology (Sung & Klein, 2006). The red box around the figure links this step to figure a. (c) Electron microscopy of DMC1 loops (Sung & Klein, 2006)

1.1.4.1 Meiosis Details

Prophase I is an important stage in meiosis. During that stage, homologous chromosomes come closer to each other in a process called synapse, and a synaptonemal complex forms between them. This complex holds both of the homologous chromosomes to become a bivalent.

At leptotene, the replicated chromosomes condense into thin threads (leptotene means thin threads in Greek) and these threads coil up form loops. These loops are the beginning of the

formation of the synaptonemal complex, an important protein structure that facilitates the adhesion of both sister chromosomes (synapsis). During meiosis, double-strand breaks happen. Of course, the usual DNA damage causes can be one reason for it. However, a different type of highly regulated DSB formation happens as well. Such a control is important because DSBs in principle cause damage to the DNA, and may cause shuffling of the DNA information. Therefore, it is imperative that they do not occur at random, but in predetermined safe regions. When these pre-programmed DSBs are initiated, the cellular machineries responsible for fixing DSBs intervene. Few DSBs are resolved through homologous recombination, a process called meiotic recombination. For the DSBs that will be fixed through recombination, during leptotene, the process is started all the way to the strand invasion point.

Next, at zygotene, the tetrad becomes much more tightly linked by the full formation of the synapsis process. In male meiosis, the sex chromosomes (chromosomes X and Y) play the role of homology for each other, partially pairing at a homologous region called the pseudo-autosomal region (PAR). Another sex-specific difference is that the number of recombination events is higher in female meiosis than in male ones. After, at pachytene, the chromosomal crossover continues by linking up the chromosomes through the Holliday junction. Holliday junctions, in the context of meiotic recombination, are called chiasma (plural is chiasmata). Since not all DSBs are resolved using homologous recombination, therefore not all DSBs will create a chiasma. These chiasmata hold tight both homologous chromosomes together during meiosis I.

In the diplotene stage, the synaptonemal complex disintegrates, and the chromosomes would separate from each other again. However, chiasmata formed during the DSB repair process is still holding them together. These chiasmata should remain until anaphase I.

After diplotene, the diakinesis phase starts, and this is where the nucleus disappears and the meiotic spindles form.

In anaphase I, the spindles would pull sister chromosomes away from each other. But for this to happen correctly, each sister chromosome's centromeres should face an opposite pole of the cell.

Such precise opposite orientation is only guaranteed by the mechanical rigid attachment between both sister chromosomes through the chiasmata. Improper orientation is detrimental for proper segregation of the chromosomes, and disoriented chromosomes would suffer from a first division nondisjunction, leading to a trisomy in the daughter cells (and a constantly defective segregation causes infertility, explained below). And this shows one of the important roles of meiotic recombination, it creates the chiasmata, and proper segregation needs at least one chiasma per tetrad. The other important role of chiasmata is their role as a major source of genetic diversity in the offspring. The resolution of Holliday junctions can be a crossover, which exchanges the material between homologous chromosomes, giving rise to a new hybrid chromosome.

During the last phase of meiosis, cytoplasmic division is distinct between males and females. In males, the division is more or less equal, resulting in four spermatids of similar size. In females, this division is highly asymmetric, resulting in one cell having most of the cytoplasm and is viable for reproduction. The three remaining daughter cells become a polar body and are removed. Another noteworthy sex-specific difference is that in females, the oocytes form at the embryo stage. But they get arrested at the point of chiasmata formation at prophase I until further development is triggered through hormones when it is the time to release the egg.

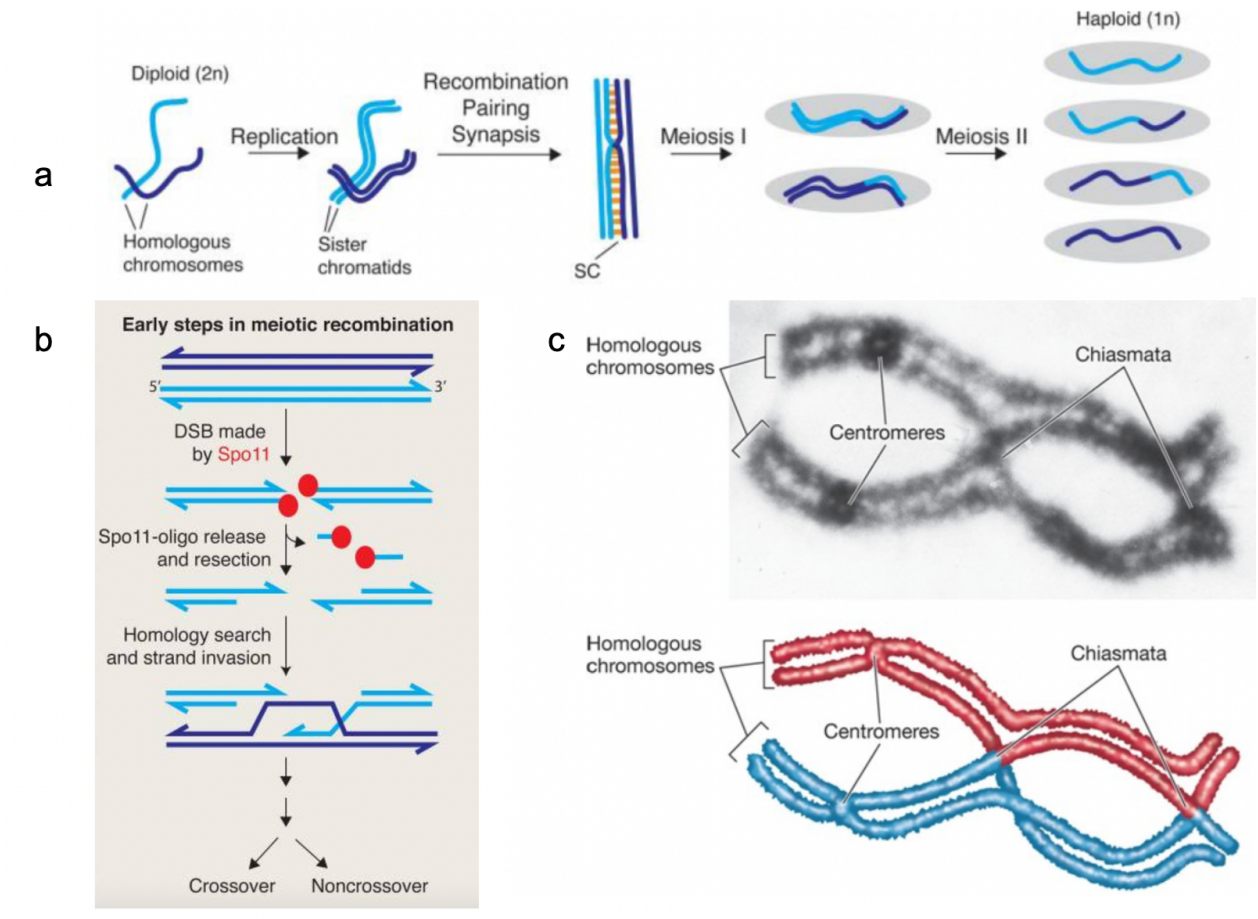

Figure 1.2 Meiotic recombination leads to the formation of chiasma. (a) An overview of meiosis (Keeney et al., 2014) (b) Early steps of meiosis (Keeney et al., 2014) (c) An electron microscopy image of a tetrad and an overlayed rendering of the bivalents (Sung & Klein, 2006)

1.1.4.2 Meiotic DSB Formation and Regulation

1.1.4.2.1 How we study DSB Hotspots

Meiotic recombination was well inferred for a long time: according to Mendelian heredity, the principle of independent assortment meant we should expect traits mixing to be roughly equal. However, it was observed that certain traits were almost always inherited together. Further studies showed that these genetically linked traits are in proximal DNA regions, leading to the development of the theory of linkage disequilibrium (LD). We can study LD through pedigrees using genetically related individuals, and the genetic maps built using LD can identify recombination hotspots with a coarse resolution of a few kbps. There are other

methods that can yield similar results as well, like admixture-based genetic maps. An interesting fact is that the results of recombination hotspots are not consistent between these methods because of their different sensitivities. For example, pedigree methods results are more sensitive to selection, and they represent currently active hotspots.

Studying LD has shown that LD locations persist for temporal time only then they change locations. The reason behind this remained unclear until finer resolution methods became available. Finer resolution methods rely on directly working on sequences of gamete cells from individuals. Many of these methods are ChIP-seq based, working by using laboratory assays targeting the proteins involved in the process of recombination. Direct ChIP-seq usage is more a more challenging process for a variety of reasons including the low frequency of recombination events and high heterogeneity of the sex cells (Khil et al., 2012). So other approaches tailored for DSB detection were developed as well, one called Single Stranded DNA Sequencing (SSDS). In SSDS, an extra step is added to the ChIP-seq process. This step takes advantage of molecular properties of ssDNA (hairpin formation) to remove most of the double-stranded DNA from the analysis, leaving mostly ssDNA for the sequencing. The results are higher sensitivity and specificity to the identification of DSB locations.

There is another method called sperm-genotyping can also be used to study DSBs, however it does not scale to genome-wide analysis (Khil et al., 2012).

In practice, such direct methods only study sperm cells. This is due to the relative ease of obtaining a large sample of cells, and also that female oocyte has already passed through the initial phases of recombination (as they are arrested at prophase I). On the other hand, one important counter-advantage of pedigree and admixture methods is that they work on data coming from both male and female individuals.

1.1.4.2.2 Mechanics of Meiotic Hotspots

All the aforementioned methods helped explain the distribution of programmed recombination hotspots. It is now understood that DSBs are concentrated in clusters of 1-2 kbps (kilo-base-pairs) regions called recombination hotspots. These hotspots are much more abundant in telomeric regions and become scarcer as they get closer to the centromere.

Studies show that there are well over 30,000 recombination hotspots in the human genome (~34k Myers et al., 2007 and ~39k Pratto et al., 2014). It is worth noting that despite the presence of a large number of DSBs hotspots, only a few do actually sustain an actual crossover during meiosis. As explained, a tetrad needs at least one for proper segregation, but having too many DSBs is linked to cellular problems. The average number of chiasmata per chromosome is estimated to be 1.56 (Beye et al., 2006). Other alternatives for DSB repair in meiosis can happen through sister-chromatid exchange (SCE) as well, and this mechanism becomes useful for DSBs at highly polymorphic regions (Goldfarb & Lichten, 2010), where homologous regions between sister chromosomes are not nearly identical. One study estimated that up to one third of meiotic DSBs end up getting fixed using SCE in budding yeast (Goldfarb & Lichten, 2010).

Recombination hotspots in general have on average about 100kbps (Arnheim et al., 2003) in between, but of course since hotspots are not uniformly distributed across a chromosome, they may be separated by a shorter distance towards the telomere. Hotspots are negatively correlated with regions rich in genes, however they also tend to be not too distant from a gene, they just happen rarely within a coding region (CDS) (Myers et al., 2005).

1.1.4.2.3 Molecular Processes Driving Meiotic Hotspots

Early research on DNA sequences at these hotspots showed the repeated presence of a 5-9mer short motifs (CCTCCCT and CCCCACCCC) that was overrepresented in THE1A/B retrotransposons present at meiotic hotspots (Myers et al., 2005). A later study by the same authors using phase 2 HapMap have revealed 13bp DNA motif that was thought to be present in 40% of the hotspots, called the Myer motif (Myers et al., 2008) (5′-NCCNCCNTNNCCNCN-3′). It also showed that the motif is strongly present at disease causing locations such as nonallelic homologous recombination and common mitochondrial deletion locations (Myers et al., 2008). Higher resolution methods later have shown that 70% of hotspots centers lie with 250 bps from that motif. This motif was later shown to bind to a protein called PRDM9 (Pratto et al., 2014). PRDM9 motif is not the only DNA sequence marker related to recombination hotspots. There are other motifs that are known to repulse nucleosomes (Anderson & Widom, 2001) and therefore are more abundant in NDRs that are present as

well. One such sequence is the consecutive repeats of A bp (poly-A) (Heissl et al., 2019) and in another study showed that the repeats of poly-pyrimidine and poly-purine to be present in human and yeast hotspots (Bagshaw et al., 2006).

1.1.4.2.3.1 PRDM9

PRDM9 is a protein only expressed in germ cells entering prophase I (Diagouraga et al., 2018). The protein is active across many species (but not all), such as in humans, great apes and rats. Multiple domains make up the PRDM9 protein, two of which are a ZnF and a PR/SET (Thibault-Sennett et al., 2018). The PRDM9 gene contains a minisatellite at the region coding for the ZnF, leading to a high mutation rate (on an evolutionary scale) at this location (Úbeda & Wilkins, 2011). Studies have showed that the ZnF multi-domain in the PRDM9 can form a long-lived complex with its bound DNA region. PRDM9 proteins do not work in single units, rather they form a polymer chain with other PRDM9 proteins and they work as a long unit (Baker et al., 2015, p. 9).

When the PRDM9's ZnFs amino-acid sequence was analyzed, it was expected to bind to a slightly different DNA sequence than the one inferred by the DNA sequence analysis of the hotspots (Baudat et al., 2010). Further structural studies of PRDM9 protein have shown it has a binding plasticity, meaning that it can bind to mismatched sequences, albeit with a reduced binding affinity (Patel et al., 2016). But it is important to note that mutations in the motif can affect the binding affinity at different levels, with substitutions at the most important bps in the motif completely disrupting the activity of its hotspots (Patel et al., 2016).

Such binding plasticity may be one explanation for why not all hotspots contain the motif and why there is a mismatch between the Myer motif and the expected canonical motif. On the other hand, studies done using sperm typing show a complete lack of the motif at certain hotspots (Berg et al., 2010). Also, in mice (thought to share a similar meiotic recombination like the one in humans), $PRDM9_{CST}$ allele activates multiple hotspots that do not share an exact consensus motif, suggesting a hidden complex mechanism for DNA-PRDM9 binding (Billings et al., 2013). Other proofs for non-PRDM9 motif hotspot activity exist. For example, it was shown that the PRDM9 motif exists outside of hotspots, proving that the presence of

the motif alone is not the reason for activity of the hotspot and that there must be other factors involved such as chromatin environment (Brick et al., 2012). Another extremely importance proof is a fertile human female with a mutation in her PRDM9 causing the protein to cause its ZnF. A small fraction of crossovers in that individual happened at PRDM9 dependent hotspots, and the remaining happened outside (Narasimhan et al., 2016).

The role of methylation has some contradictory evidence. One study showed that DNA methylation plays a coarse role in determining hotspot regions. It found a high correlation when the calculations used large bins (500 kbp bins) but with a low correlation at smaller windows (J. Zeng & Yi, 2014). In another study, it was shown in vitro that methylation in CpG islands reduced PRDM9's binding affinity to its motif (Diagouraga et al., 2018).

Another chromosomal feature with high correlation to DSBs is NDRs (Pan et al., 2011) (Baker et al., 2014). They are known for having high DNA accessibility, and DNA accessibility seems to be a pattern for hotspots. Even regions with nucleosomes show signs of chromatin modification that increase accessibility. The PR/SET component of the PRDM9 protein can induce histone-modifications, especially an H3K4me3 methylation (Hayashi et al., 2005), causing a more accessible DNA to the cellular machinery. But the exact role of the H3K4me3 is not completely understood. These marks asymmetrically flank from both directions many of the meiotic recombination hotspots regardless of the PRDM9 motif orientation, leading to the belief that they are an important meiotic recombination marker (Baker et al., 2014). H3K4me3 is not the only methylation that is found at hotspots, but also H3K36me3 are also found in hotspots despite being mutually exclusive with H3K4 modifications elsewhere in the genome. In vitro experiments showed PRDM9 can put H3K36me3 marks, albeit at a much slower rate than making H3K4 marks (Powers et al., 2016). Other potential factors include the chromosome size. Although the number of recombination events seems to be very close across chromosomes of different sizes, smaller chromosomes have a higher density of hotspots. One exception to this rule is chromosome 19 (Myers et al., 2005), where it has a lower density for hotspots. Chromosome 19 is very rich in genes, which is an interesting exception since hotspots happen near genes. Another exception is chromosome X, which has higher than expected hotspots.

1.1.4.2.4 PRDM9 Evolution Across Species

The process of recombination causes the destruction of the sequence at the recombination site. There are multiple reasons for this, including gene conversion, that happens when the cleaved site is fixed using the sequence from the homologous chromosome that does not contain the exact same motif. Another reason is that naked ssDNA (before it gets repaired) is more susceptible to mutation because of deamination. This process of motif erosion is observed when comparing ortholog sequences of hotspot regions in human and chimpanzees (Myers et al., 2010, p. 9).

Such destruction of motifs is referred to as the "hotspot paradox". The paradox here is that since PRDM9's activity depletes the motif it needs to work, it should at some point run out of target regions and therefore the hotspot activity should have been very low. Yet, the activity is still high, which means that the PRDM9 constantly finds new regions to work on. The Red Queen theory of recombination hotspots (Úbeda & Wilkins, 2011) explained this paradox, that as the genome gets depleted from the correct motif, the PRDM9 evolves in parallel to recognize new motifs. Such evolution migrates the hotspots to new regions, and that also explains the non-constant LD identified hotspots through pedigrees. Such fast evolution can be also explained that the ZnF coding region has a highly mutating minisatellite, helping to keep pace with the sequence erosion.

Such a high rate of mutation has a cost, though: PRDM9 is accepted to be a speciation gene (Mihola et al., 2009). It can cause hybrid sterility (or partial sterility) in offspring of individuals with evolutionary distant enough PRDM9 ZnF. In fact, this process was inferred long before understanding PRDM9, to which it was referred to Hst1 (Hybrid sterility gene 1) locus (Forejt & Iványi, 1974). The reason for this sterility is that in a hybrid's gamete, each PRDM9 allele finds plenty of non-eroded sequences in the non-self-chromosome. This results in asymmetric hotspot distribution between homologous chromosomes, which may delay the resolution of all these DSBs and therefore the cell gets discarded (Mihola et al., 2009). Another study (Davies et al., 2016) showed this PRDM9 difference and hybrid sterility relationship. In an experiment where a hybrid rat (M6 and PWD) that is known to be a sterile hybrid had its ZnF sequence modified to that of the human's $PRDM9_B$ allele. The human

PRDM9 is distant enough to have its target motif equally present in both homologous chromosomes, and this solved the sterility problem. The exact reasons for repair delays are not understood, but one study hypothesized that both homologous chromosomes should be targeted in order to facilitate homology search (Davies et al., 2016).

1.1.4.2.5 PRDM9 Alleles in Humans

The human PRDM9 gene has many variations, with the A, B and C alleles being the most frequent (Pratto et al., 2014). The allele PRDM9A is most frequent in European populations and the C allele in African populations. Alleles' A and B ZnF recognize very similar motifs, and therefore their hotspots intersect. The C allele codes for a very different motif and therefore its hotspots occur at different places (Pratto et al., 2014).

These different alleles not only recognize a different motif but also, they have different binding affinity to these motifs. For example, $PRDM9_C$ has a higher affinity for its motif than $PRDM9_A$'s affinity for its own motif. This in parts can explain why the $PRDM9_C$ is dominant over $PRDM9_A$ in heterozygote individuals (Baudat et al., 2010).

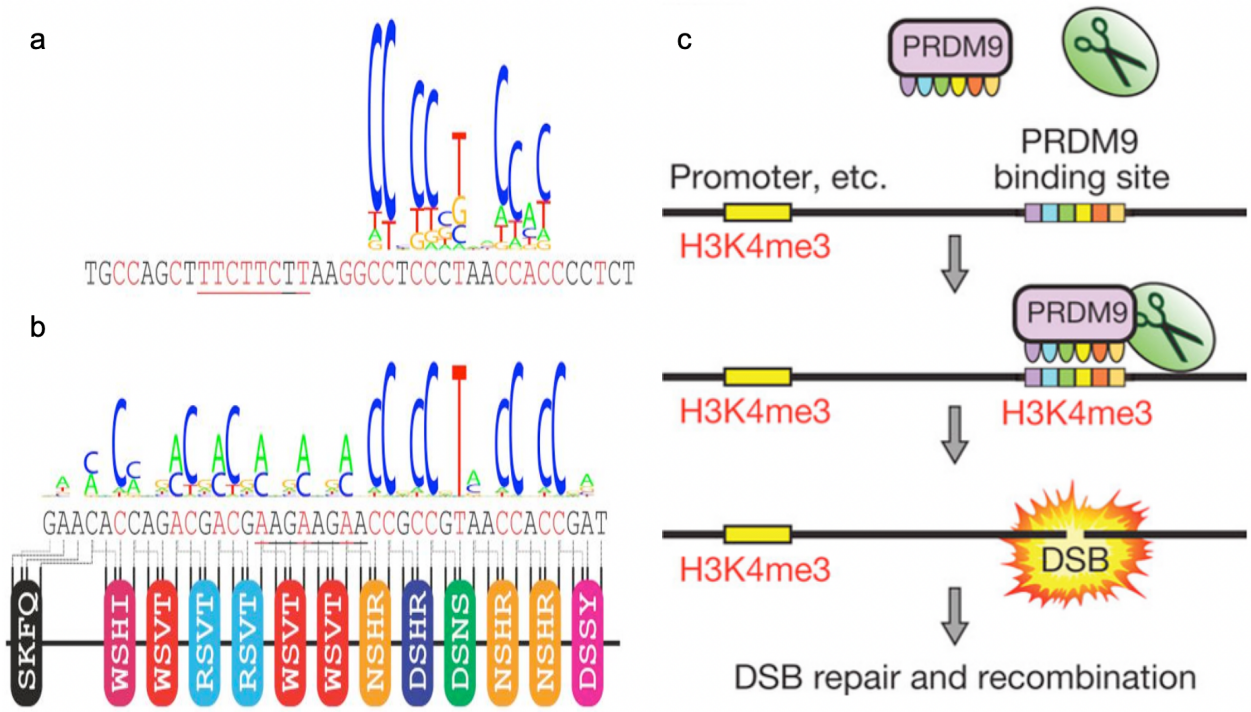

Figure 1.3 PRDM9 motifs and how it works (a) PRDM9 Myers sequence (Myers et al., 2010, p. 9) as a logo plot from the Position Weight Matrix (PWM) (b) ZnF sequence (in amino acids

shown at the bottom) and its expected motif derived from the amino acid a.a. Sequence shown at the bottom , identical ZnF share the same color (Myers et al., 2010, p. 9) (c) PRDM9 binds to the motif, and recruits SPO11 (represented as the scissors) to nick the DNA and produce the double strand break (Brick et al., 2012)

1.1.4.2.6 Overview of the remaining PRDM9 mechanism of meiotic DSB creation

PRDM9, as explained, recognizes the region where a hotspot would happen, and it would bind there and induce the histone modifications. The PRDM9, potentially along with the presence of the H3K4 histone modification as well interaction with other proteins such as SPP1 and Rec8, then recruits a protein complex responsible for breaking the DNA backbone. This protein is the DNA topoisomerase VI complex, part of which a protein complex made of spo11 and top6B proteins acts as an endonuclease that breaks the DNA itself. SPO11 will nick the DNA backbone, causing the DSB to happen. The 5' ends will get trimmed, then afterwards, DMC1 and RAD51 proteins will coil up around these sites. DMC1 is active only during meiosis. These two proteins handle the search for homology, and once they bind the DNA, the process of recombination continues till the end, as described earlier. Since DMC1 binds to ssDNA at DSB sites, it is a good proxy for studying the hotspot sites on a genome.

1.1.5 Position Weight Matrices

As discussed in the previous section, DNA motifs play a highly important role and extracting them involves performing a sequence of operations. When we have many of these highly similar sequences, we can use alignment algorithms to derive a consensus sequence, which is a sequence that maximizes the similarity with all the other sequences. Another way to represent such highly similar sequences is a matrix called position weight matrix (PWM, also referred to as a position-specific weight matrix (PSWM) or position-specific scoring matrix (PSSM)). PWM has one row per nucleotide, and one column per position. To compute the PWM, we start by calculating the position frequency matrix (PFM), by counting the occurrences of each bp at each position. Next, we use the PFM to compute the probabilities per bp per position. Assuming independence between the value at each position, we get

$$p_{i,j} = \frac{count(bp_{i,j})}{\sum_{k=1}^{k=|k|} b\, p_{i,k}}$$

where i is the position in the sequence (the column), j is the bp (the row) and |k| is the number of possible values (4 in the case of DNA). The problem with such a model is that if a bp was never seen in any of the sequences, it will have a probability of zero, which should not be correct. To correct this problem (which is likely to happen if we have a small number of motifs), we use pseudo counts

$$p_{i,j} = \frac{count(bp_{i,j}) + \lambda}{\sum_{k=1}^{k=|k|} b\, p_{i,k} + |k|\lambda}$$

The values are usually represented as the log-likelihood of seeing a specific bp at this position. To do this, we use a background model. The simplest background model is the uniform distribution (that all bps are equally likely to appear at that position). However, it is better to use a more informed background model that considers the gc content. We compute the logarithm of the PFM by the background value to get the log-likelihood, which is the PWM.

$$p_{i,j} = \log_2\left(\frac{p_{i,j}}{b_j}\right)$$

The problem of such a representation is that all positions are weighed with respect to the probabilities, which must sum to one. However, converting the representation to the Shannon information carried by each position yields a more intuitive result, as it attenuates the values as they get closer to the background model. We can compute the information carried at a specific position in the sequence as

$$I_i = \sum_{j=1}^{j=|k|} p_{i,j} \log_2\left(\frac{p_{i,j}}{b_j}\right)$$

which is the Kullback-Leibler divergence.

Graphically, we can represent these by drawing the letter with a proportional height to the value. An example of such a representation is in figures 1.4 and 1.5 where they depict the logos of a probability and information matrices respectively.

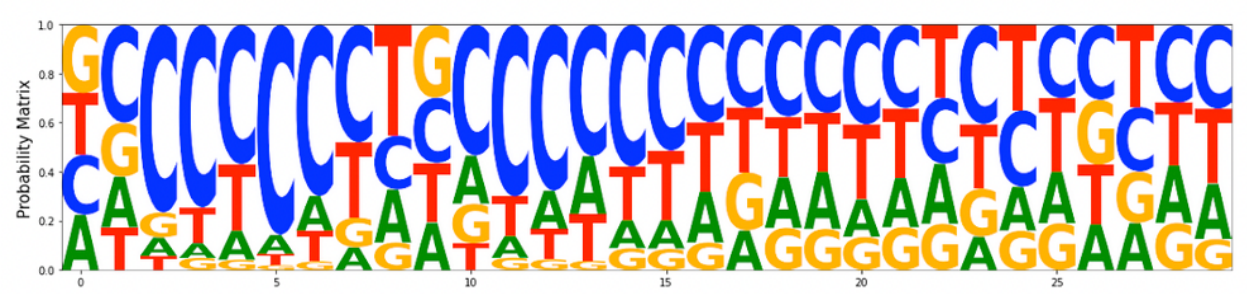

Figure 1.4 PWM Probability Example

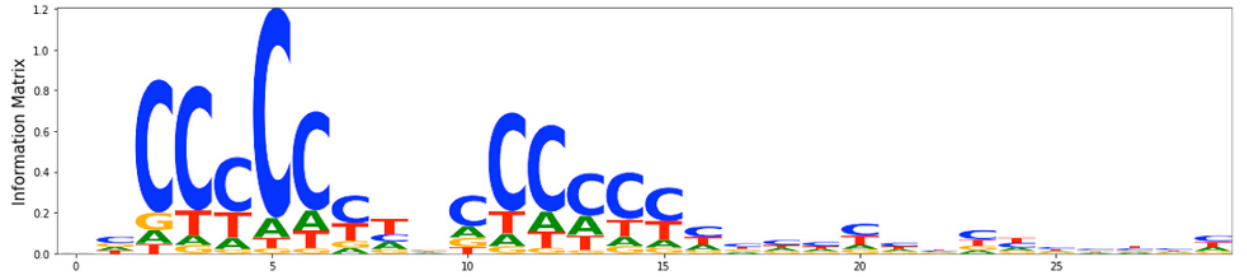

Figure 1.5 PWM Information content

We can see that as the position yields little information with respect to the background model, the position is plotted with a lower height.

1.2 Machine Learning

1.2.1 Reasoning

Humanity has examined reasoning and logic since antiquity. Logic is the process of coming up with conclusions based on a set of premises. Formal logic has defined multiple approaches for reasoning, including deductive reasoning, inductive reasoning, and abductive reasoning (Mill, 2011).

Deductive reasoning aims to draw conclusions in a top-down fashion, i.e., from the generalization to the specific. A conclusion from a deductive process is necessarily true if the premises are also true. For example, in the syllogism, “All men are mortal. Socrates is a man. Therefore, Socrates is mortal” (Mill, 2011), the conclusion that Socrates is mortal is necessarily true if the premises are also true.

Inductive reasoning takes a bottom-up approach going from specifics to generalization. Conclusions from inductive processes are uncertain, but probable. We must update the conclusions coming from inductions if a new body of evidence comes up. Statistical generalizations (drawing a conclusion about a population based on a sample) are a form of inductive reasoning. Inductive reasoning does not explain observations, rather, it stops at stating what is probable, and not the cause.

Abductive reasoning, in its modern usage, refers to "inference to the best explanation" (Douven, 2021). In this approach, we reach conclusions because there are no hypotheses that offer a better explanation. An important distinction from the previous forms is that abduction is used to going backwards in reasoning, to explain causes based on effects. Abductive reasoning has strong links to Bayesian statistics. Inductive and abductive reasoning can be referred to as non-necessary reasoning.

1.2.1.1 Computational Reasoning

After the invention of computers and as they became an important part of the human process of decision making, the field of artificial intelligence (AI) has developed to mimic formal logic. Computational logic, for example, is the branch of AI that studies deductive reasoning in computers (Robinson, 1971).

1.2.1.2 Approaches to Computational Non-Necessary Reasoning

Computers have shined at non-necessary reasoning as well. Non-necessary inference is statistical in nature, as explained above. It is seen to be tightly linked to learning. Examples are machine learning (ML) and statistical learning. One definition for ML has linked it to learning from experience gained through seeing data. It states that "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E" (Mitchell, 1997).

Statistical learning is broadly defined as learning a generalization from examples (data or dataset). Leo Breiman defined (Breiman, 2001) two approaches of statistical learning, based on the assumptions of the processes that have generated the dataset. In the first approach,

the assumption is that such processes are model based, and therefore we would fit a statistical model to make inferences.

The second approach is algorithmic. It does not make any assumptions about the statistical distributions of the processes that generated the dataset. Rather, they are treated as unknown. We can infer that this second approach is ML, as he states ML algorithms such as random forests. Other examples of ML algorithms are support-vector-machines (SVM) and artificial neural networks. These algorithms can learn in a variety of settings, such as supervised learning and unsupervised learning.

Breiman continues that the second approach is result oriented. For example, the goodness of the model is decided based on the prediction accuracy. However, it may fall short in terms of mathematical rigour that proves the validity of the models.

1.2.2 Supervised Task Learning in Artificial Neural Networks

The goal of supervised machine learning is to generalize the patterns within the input data– the features, in order to predict an outcome (commonly called the target variable, or objective). Artificial neural networks are one form of machine learning and are the main approach we are considering in this work. They are made of successive layers of affine transformations, followed by a non-linearity function. When the number of layers is greater than two, these artificial neural networks are designated as "deep", and hence the term deep learning. Neural networks are used to model very complex models, and analytical solutions are not practically feasible. Therefore, they are trained incrementally by minimizing a loss function that describes how far is the network's output from the desired one, and in which direction to change weights (the gradient) to come closer to the desired outcome using the backpropagation algorithms (Bengio et al., 2013).

1.2.3 Neural Networks

The goal of supervised machine learning is to generalize the patterns within the input data– the features, in order to predict an outcome (commonly called the target variable, or objective). Artificial neural networks are one form of machine learning that can be trained in

a supervised manner and are the main approach we are considering in this work. They are made of successive layers of affine transformations, followed by a non-linearity function. When the number of layers is greater than two, these artificial neural networks are designated as "deep", and hence the term deep learning.

At the heart of ANNs is the neuron. The model of a neuron is similar to that of a linear regression, with an extra non-linearity function added to the output. This non-linear function is often referred to as the activation function.

$$\hat{y} = f(X, W) = \phi(W^T X + b)$$

Where W is the learned weights of the model, and b, the bias, is a constant.

The There are a variety of functions ϕ that can be used as an activation function. An important criterion for such functions is that they must be differentiable. The following table shows some examples of these

Table 1.1 Most commonly used activation functions in deep learning.

NAME	Equation	Range
Sigmoid	$S(x) = \frac{1}{1+e^{-x}}$	[0,1]
Tanh	$S(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	$[-1,1]$
Rectified Linear Unit (ReLU)	$S(x) = \frac{1}{1+e^{-x}}$	$[0,\infty]$
Softmax	$S(x) = \frac{e^{x_i}}{\sum_j e^{x_j}}$	[0,1]

A single neuron alone cannot model complex equations but stacking multiple neurons together in parallel adds modeling power. Such a stack is called a layer. According to the universal approximation theorem, such a layer is capable of approximating any arbitrary function. In practice, however, it is easier to train a succession of neural networks layers. The mathematical model of 3 layers of neurons is

$$\hat{y} = f^3(f^2(f^1(X, W^1), W^2), W^3) = \phi^{(3)}\left(W^3\phi^{(2)}\left(W^2\phi^{(1)}(W^1X + b^1) + b^2\right) + b^3\right)$$

Where the superscript represents the layer number, and not an exponent.

The loss function (Also referred to as cost function) is a function that quantifies a distance between the model's error. Such a function for our 3 layers network example is

$$C(y, \hat{y}) = C\left(y, \phi^{(3)}\left(W^3\phi^{(2)}\left(W^2\phi^{(1)}(W^1X + b^1) + b^2\right) + b^3\right)\right)$$

Where C is the loss\cost function.

The learning process consists of minimizing such a function. Because finding an analytic solution is not practically feasible, neural networks are trained incrementally by minimizing the loss function. To minimize the loss function, we compute the gradient of the error with respect to the inputs. Applying the chain rule, we get

$$\frac{dC}{dX} = \frac{dC}{d\phi^3} \cdot \frac{d\phi^3}{d\phi^2} \cdot \frac{d\phi^2}{d\phi^1} \cdot \frac{d\phi^3}{X}$$

which is used to know how much we should change W's and b's in order to reduce the errors. This algorithm is called the backpropagation (Rumelhart et al., 1986) algorithms.

This learning is performed using iterative methods, like the gradient descent approach and its derivatives, where at each training step we aim to reduce the loss function by a small step.

There are no mathematical guarantees to find the best workable solution (A global minimum of the loss function), which is why we must take some precautions to make sure that the model would perform as expected on unseen points. One of these measures is holding out a portion of the data for testing the performance of the model (A test set) (LeCun et al., 2015).

If the model is good, then we should get a comparable performance to the training data. Another serious problem with training deep neural networks is overfitting (Goodfellow et al., 2016). Overfitting is when the network learns (memorizes) the specifics of the training examples rather than the generalized signal common within the data points. Many techniques are used to fight overfitting in deep neural networks, including L1 and L2 regularization, early stopping and dropout (Goodfellow et al., 2016). Although dropouts (Srivastava et al., 2014) were originally intended to be used during model training as a way to combat overfitting, further research showed that using dropouts during inference

approximates a deep gaussian process. Such approximation means that we can get an uncertainty estimation about the estimation, and not just a point estimate.

The other part of training a deep learning model is architectural considerations, such as the number of layers and the number of units in each layer. Hyperparameters search is finding the best possible parameters concerning these decisions. The term comes from the statistical term describing the parameters of the parameters in Bayesian models.

1.2.4 Number of Tasks in Supervised Learning

1.2.4.1 Single Task Learning

Single task learning is the most common approach for doing machine learning. In single task learning, the model is trying to learn a single outcome.

1.2.4.2 Multitask Learning

Multitask learning (MTL) refers to the approach of learning multiple objectives at the same time. It can be done in different ways, such as joint-learning more than one task together, adding the other tasks as an auxiliary task to the side to the one that interests us or by "learning to learn" (Caruana, n.d.). MTL improves the model's generalization by being forced to pay attention to different signals within the dataset needed to be learned for different tasks (Abu-Mostafa, 1990). MTL can be approached either by hard parameters sharing or soft parameters sharing (Caruana, n.d.). In hard parameter sharing, we have a single network with multiple outputs, where the initial layers are shared among all tasks. This approach reduces the risk of overfitting by the order of the number of tasks being learned (Caruana, n.d.). The second type of MTL is the soft parameters sharing. In this approach, the tasks' models are kept separate, but they are used to regularizing each other (Caruana, n.d.). Some reasons the MTL approach is promising is that it enables the model to better identify the patterns from the noise in the data, since it needs to find the relevant signal that is useful for all tasks at hand (Caruana, n.d.). This is also related to using hints (Kaiser et al., 2017), where some features are easier to be extracted in the light of certain tasks than others, although both tasks make use of the same features. Another reason is the representation bias, a consequence coming from the fact that the model will have to prefer to learn the features

that apply to all tasks at hand. This also acts like a regularizer, helping to reduce the risk of overfitting. For applying the multitask learning approach, we are not constrained by only using two related tasks. We can also use two opposite tasks if such data are the only ones available. This approach is called adversarial training, in which the model is trained using an adversarial loss, in which the optimization gradient for the task at hand is the opposite of the adversarial task. Multitask training has proved beneficial in many domains. For example, machine translation models were trained in many languages, they learned to translate to new languages within only one iteration of learning, a behavior called few\one\zero learning (Li et al., 2016).

1.3 Machine Learning in Genomics

Purely wet-lab experimental approaches for uncovering the functional role of a given genomic region are expensive and time-consuming. This makes computational approaches highly desired (LeCun et al., 2015). In this review, we will go through the different deep learning architectures and approaches that aimed to decipher the functional role of genomic regions using mainly the genomic sequence. These trained models can be used not only to annotate newly sequenced genomes but also in predicting the effects of single nucleotide variants (SNV) on the overall function of the region.

1.3.1 ML Success in Biology is Proved on Different Data Sources

Biological data can come from different functional levels, lowest ones being DNA level, passing through single-cell level data, multicellular data all the way to population level data (Zitnik et al., 2019). Examined DNA functional annotation papers mainly on the low-level datasets. One of the most important data sources is the DNA sequence itself, which alone can be mined for a wealth of information, including predicting epigenetic information (Angermueller et al., 2017). Sequencing technologies have provided the opportunity to get several sequencing-based epigenetic data that can augment the sequence include:

DNase-seq (DNase I hypersensitive sites sequencing, also sometimes referred to using simply DNase I cleavage since it is the base method combined with next generation sequencing) experiments measure chromatin accessibility (Boyle et al., 2008), which hints

to regions that can bind to transcription factors, shall the DNA sequence is receptive to it. Assay for transposase-Accessible Chromatin using sequencing (ATAC-seq) can also be used for the same purpose (Buenrostro et al., 2013).

Another potentially useful dataset is the RNA-seq (Lister et al., 2008). RNA-seq data can quantify gene expressions, and therefore they can augment other genomic datasets. For example, RNA-seq data was provided as part of the ENCODE-DREAM challenge and was used by Quang and Xie (Quang & Xie, 2017) in their model.

Finally, Hi-C technology can reveal regions of DNA that are spatially near each other in the cell, revealing long-range interactions between different chromosome regions (Lieberman-Aiden et al., 2009). Chromatin Interaction Analysis by Paired-End Tag Sequencing (ChIA-PET) (Fullwood & Ruan, 2009) and Hi-ChIP (Mumbach et al., 2016) can also be used for the same purpose.

These datasets have already been used alone or together in deep learning tasks for genomic regions mapping and predicting their epigenetic properties. For example, Zhou and Troyanskaya (Zhou & Troyanskaya, 2015) used DNase I sensitivity with the DNA sequence from the GRCh37 assembly to predict transcription factor binding and from there they used this to evaluate the effect of non-coding variants. Using the same input, Kelley et al. (Kelley et al., 2015) took a multitask approach by predicting simultaneously the DNA activity in 164 cell types. Zeng et al. (H. Zeng et al., 2016) used ChIP-seq data, DNA sequence, and Hi-C data for motif discovery and motif occupancy. Wang et al. (M. Wang et al., 2018) used ChIP-seq and the DNA sequence to predict the DNA\Transcription factor binding intensities. Min et al. (Min et al., 2017) used only the DNA sequence to detect enhancer sites from input DNA sequences. For the same task, Cohn et al. (Cohn et al., 2018) also used the sequence along with the ChIP-seq data. In Umarov and Solovyev (Umarov & Solovyev, 2017) work, the DNA sequence of 5 distant organisms (a mix of prokaryotes and eukaryotes) was used to train a neural network on promoter regions recognition, and they showed that trained features on one organism are useful to be transferred to another. Angermueller et al. (Angermueller et al., 2017) used cell methylation profiling (Smallwood et al., 2014) of neighboring regions to predict the CpG methylation state of a DNA sequence. Quang and Xie (Quang & Xie, 2017)

have used more data side by side, including genomic sequence, genome annotation, gene expressions and epigenetic signals at the single nucleotide level such as DNase I cleavage to predict transcription factor binding. In Xiong et al. (Xiong et al., 2015) work, DNA sequence is used to identify the splicing of RNA (for labeling the data, RNA-seq data was also used) revealing insights about three diseases.

Although the approach of interest is deep learning, which is known for its ability to do automatic feature engineering, some tasks are still inherently hard without our intervention in feature transformation. One of those interventions is transforming sequence features into a more suitable representation that encodes for inferred notions, called a latent representation. Sequence constituents (tokens) usually do not have an inherent order, which makes numeric encoding for them generally unfavoured (Ng, 2017). Instead, they should be expanded into mutually exclusive binary variables, a technique called one hot encoding. Such one hot encoding increases the dimensionality of the input, so, for example, if we want to study the DNA sequence in terms of k-mers, we will have to have our input increase to 2k. High dimensionality is not the only problem here, also the fact that one-encoded vectors are all equidistant from each other and not continuous makes the representation lose some of the signal to be learned. A proven approach in natural language processing is to build an n-dimensional continuous vector to represent these one-hot-encoded tokens. There are two general approaches for this using neural networks: either given the token we try to predict its context (i.e. the tokens that are around it) (Mikolov et al., 2013) or given the context we try to predict the token in the middle. The first approach is faster to train but the second one is better with rare words (Ng, 2017). After the training process is complete, the weights of the inner hidden layer of the trained network hold the information about the latent representation of each token. This approach yields very interesting representations of words, and even doing arithmetic between tokens is possible. For example, the following operation is true in (Vylomova et al., 2016): king – man + woman = queen. This approach applied to biological sequences, and the results were very interesting. In Biovec (Asgari & Mofrad, 2015), the skip-gram approach was applied to protein sequences, using k-mer size of 3. The resulting latent representation, called embedding, was used by a separate task to predict the protein's family. Based on the sequence embeddings alone, it performed as good

as an SVM approach that needs more extra features like hydrophobicity, secondary structure and solvent accessibility, showing that the embeddings captured meaningful structural 3D features that could be inferred from the sequence alone (Asgari & Mofrad, 2015). An application of the approach to the DNA sequence is seq2vec (Kimothi et al., 2016). They have found that the trained embedding distances between k-mers had very similar results to global alignment scores coming from Needleman-Wunsch algorithm (Needleman & Wunsch, 1970) and local alignment scores coming from the Smith-Waterman (Smith & Waterman, 1981) one. In a different approach (Ng, 2017), Patrick Ng generalized the concept by training on variable k-mer DNA fragments instead. Another insight for feature engineering was proposed by Shrikumar et al. (Shrikumar et al., 2017). In this work, they suggested that training the network with separate filters of the forward and reverse complement while forcing later layers to have the same signal for both patterns increased the accuracy over the test set and stating that taking advantage of known biological information can increase the efficiency of a trained model. Sequence embeddings are not the only possible feature engineering approach, there are other opportunities of using machine learning to increase the quality of the data for further modelling, for example by using generative adversarial models to filter bias in the data by learning the original distribution of the data from biased one (Zitnik et al., 2019).

Deep learning has been used to explore meiotic recombination hotspots as well, through population and genetic mapping data. Nath et al. (Nath & Karthikeyan, 2018) used yeast data to feed autoencoders for feature extraction, then performed the classification using tree-based ML algorithms. Using the same dataset, Khan et al. (Khan et al., 2020) encoded the input sequence using Gapped Di-nucleotide composition before passing the resulting features to a fully connected neural network for the classification. Li et al. (Li et al., 2021) in a preprint used a DanQ-like hybrid CNN-RNN architecture with the addition of a multi-head attention on 7 different genetic maps datasets in addition to histone modification data and showed that deep learning gave highly accurate and interpretable results. Brown and Lunter (Brown & Lunter, 2019) used simulated and LD datasets using a CNN variation called equivariant CNN networks for predicting the DNA sequences and reported improvements to both accuracy and motif finding compared to normal convolution. All of these predict

location of crossovers hotspots and would miss recombination activity resulting in gene conversion events. To our knowledge, nobody has attempted using ChIP-Seq DMC1 data with deep learning methods to classify DNA sequences for their recombination activity, which would reflect both crossovers and gene conversion events.

1.3.2 Architectures

1.3.2.1 Fully Connected Neural Networks

Fully connected neural networks are the most straightforward approach in deep learning. In this approach, all neurons in each layer are connected to all neurons in the previous and the following layer, and the weights of those connections are updated during the learning phase. Fully connected architecture is powerful, but because it does not take any advantage of the structure of the input, they tend to have the highest number of parameters to train. Liu et al. (Liu et al., 2016) trained a model to predict enhancers, reaching state-of-the-art accuracy on the task. It was trained on 1114 heterogeneous features to test the approach of learning from different data sources. Then they used the same approach to learn enhancers prediction from 22 cell types\tissues. They further commented that including chromosomal conformation data (like Hi-C data described above) should further increase the accuracy of the model. Li et al. (Li et al., 2016) used a fully connected model to distinguish enhancers, promoters, and background sequences.

1.3.2.2 Convolutional Neural Networks

Convolutional neural networks (CNN) rely on the idea of learned parameter sharing, reducing the number of needed parameters in an equivalent fully connected network. The way they work is very similar to the visual cortex (Hubel & Wiesel, 1968). Early application of this approach involved using human-designed kernels (Lin & Inigo, 1991), but soon the way of learning the network-weights automatically through gradient descent was created (LeCun et al., 1989). The CNN have the advantage of being able to recognize its target features regardless of their position within the input image (translation invariant) (LeCun et al., 2015), which makes it ideal to locate objects of interest when their initial position within

the context is not known beforehand. Convolutional neural networks have many convolution kernels (filter) per layer, each will be trained to identify a useful feature. These kernels scan the image doing a matrix multiplication operation called convolution, hence the name of the architecture. The size of the step of the scan is called a stride. The higher the value of the matrix multiplication, the more similar is the region to the filter. Each convolution layer learns to use the features captured by its previous one to recognize more complex features, eventually able to recognize very complex patterns, as in the case of computer vision applications. In most cases, a convolutional layer is followed by a pooling layer, which acts as a summary of the similarity of the region with the filter. The most common pooling layer is the max-pooling (LeCun et al., 2015) (Sakoe et al., 1989). The original convolutional architectures need a fixed input size, and to comply with this, all explored papers have fixed the input sequence to a fixed length.

To process a sequence of nucleotides using a CNN, we transform the sequence into a one-hot-encoded matrix, effectively having an equivalent format of an image of 4 pixels high and the same-sequence-length pixels long. For genomic regions annotation, there have been different approaches to the problem (Alipanahi et al., 2015; Blum & Kollmann, 2019; Cohn et al., 2018; Kelley et al., 2015, p. 20; Min et al., 2017; Umarov & Solovyev, 2017; M. Wang et al., 2018; H. Zeng et al., 2016; Zhou & Troyanskaya, 2015). The first convolutional layer of the network works as a position-weight-matrix that scans for motifs (Alipanahi et al., 2015). With transcription factor binding papers (Cohn et al., 2018; Min et al., 2017; M. Wang et al., 2018; H. Zeng et al., 2016), the motifs were verified against known motif databases such as JASPAR (*JASPAR*, n.d.), and the results were similar to each other and outperforming the earlier SVM methods (Quang & Xie, 2015). The most common kernel size employed was 24 basepair long, but Zhou and Troyanskaya (Zhou & Troyanskaya, 2015) used a window size of 8 basepairs for example. With the exception of the method of Umarov and Solovyev (Umarov & Solovyev, 2017), all other papers used 3 layers of convolutions. Something to note in here is that these numbers are small compared to the usual ones related to computer vision tasks. For example, VGGNet (Simonyan & Zisserman, 2015), a 2014 network, had 16 convolutional layers and ResNet (He et al., 2016) had 152 layers. This hints that the level of complexity of the task is much smaller than computer vision (H. Zeng et al., 2016). Zeng et

al. (H. Zeng et al., 2016). have experimented with the number of convolutional filters to use and have noted that there was an improvement of the network's performance with the increase of the number of filters, and this improvement reached a saturation by having 128 filters. Although convolutional neural networks are typically used to detect features that are spatially positioned next to each other, an interesting approach by Paggi et al. (Paggi & Bejerano, 2017) to overcome this was the use of dilated convolutions, which can scan for features having some gaps between them. Another very interesting approach was applied by Blum et al. (Blum & Kollmann, 2019). In their paper, they have created what they have called circular filters. They argued convolution filters learn sometimes the required pattern but rolled to the left or right in a circular manner (i.e., shifted in a certain direction, but the shifted nucleotides that get pushed out of the frame from one end are reinserted from the other end), because that the gradient descent approach is greedy in essence, and this shifted representation of the motif is where the local optimum is. Trained networks can still do the prediction with these semi-correct filters because they partially recognize parts of the correct motifs, and then higher layer can mix and match these fractions of the correct motif together to create an accurate prediction. They have suggested what they called circular convolutions, in which they create a shifted k times filters of the learned filters, and then train the network. They found that the network always chose the most correct form of the motif and ignored the rest, and that also the network trained faster and needed fewer parameters to train.

1.3.2.3 Recurrent Neural Networks

Recurrent neural networks RNN (LeCun et al., 2015) are networks that contain special components that have a feedback loop called recurrent blocks. This feedback loop (sometimes referred to as the hidden state) is used when the next input is passed, allowing it to keep a summary of the context (For example, the subject\object and their gender in a language sequence) of the previous inputs. This ability makes them the most suitable for sequence processing, for example, the sequence of nucleotides representing the genome. The early versions of RNN (Rumelhart et al., 1986) were very limited in the amount of context they could keep as new information always overwrote the previous ones, making them not very practical (Bengio et al., 1994). The other problem was that this recurrent layer can be

unrolled like a graph. Analysis of this graph shows that each new token in the sequence has the effect of adding an extra layer to the network, effectively making it a very deep network. Such deep neural networks suffer from the vanishing\exploding gradient (Pascanu et al., 2013). A major development was the creation of the Long-Short-Term-Memory RNNs (Hochreiter & Schmidhuber, 1997) (LSTM). These have the capability of recognizing context switches, so they are trained to selectively retain or forget the context (even retaining only a fraction of it) and this allowed for vast improvements in the performance. LSTMs were hard to train, so a new simplified version called Gated Recurrent Units GRU (Cho et al., 2014) was introduced. These networks have a simpler structure than LSTMs, making them easier to train with little deterioration in the performance. Another variant that was introduced to the original RNNs and also available in LSTMs and GRUs is the bidirectional RNN (Schuster & Paliwal, 1997). Such networks scan the input sequence from both ends, allowing subsequent layers having a representation of both past and future tokens.

In all the examined work, RNNs were not used alone but with CNNs, sometimes referred to as hybrid networks. Quang and Xie (Quang & Xie, 2015). The approach they took was to create two separate branches in the network, one for the enhancers and the other for the promoters. It is also noteworthy that they have picked a relatively large filter size for their convolution layer (40 nucleotides wide). The output of these two convolutional branches is then concatenated into a single matrix, which is then fed to the subsequent bidirectional LSTM layer, which is where the long-range interaction is identified. The final different variation to present in the hybrid architectures is Chen et al.'s (Chen et al., 2019), where they introduce an attention module (Bahdanau et al., 2016) after the bidirectional LSTM. Attention mechanism was created from experience with the natural language translation effort, a type of sequence to sequence (i.e., both the input and the output are sequences, sometimes abbreviated as seq2seq) problem. In the seq2seq problem, the architectural approach is using separate recurrent blocks to generate the output sequence, while sharing only the final hidden state of the encoder as the starting point of the decoder. The attention mechanism was a breakthrough, and it works: instead of only keeping the last hidden state of the encoder, we keep all the intermediate states as well. Then we train a set of weights (a mini-internal neural network) to select which of these intermediate hidden states to use for

the current output token. This intermediate representation is a matrix of m x n dimensions, where m is the length of the input sequence and n is the length of the output. The approach's scalability is acceptable in natural language processing, as the length of sentences does not yield huge matrices. In the genomic context, to make this approach feasible, the input training is cut into sets of hundreds or few thousands base pairs. In their work, they start with the same Siamese architecture of Quang and Xi's (Quang & Xie, 2017), where each side works over the forward and reverse complement independently, then merge and pass the flow of information to the bidirectional LSTM. The attention weights are then computed using the LSTM hidden states, where they are normalized using a SoftMax function (Goodfellow et al., 2016) and then each token position is averaged. Afterwards, they add the fully connected layer. They have reported improvement in 9/13 of the TFs compared to FactorNet (Quang & Xie, 2017), and generally better than the top 4 contestants in the ENCODE-DREAM challenge. They have extended their performance analysis to other CNN\RNN models as well and show the same. An interesting part in their work is that they also analyzed the results per transcription factor and not on the whole dataset. They found that the attention approach performed much better on TFs that had low accuracy in other networks. Further investigation for which of the input signals did attention rely on the most, they discovered it had better "sensitivity" to DNase-Seq peak signal. The last benefit of using the attention mechanism is in the ability to look at which token had the highest responsibility to the network's output, demystifying a bit of what the network learned. This is discussed in more details in the section below about interpreting the model's output.

1.3.3 Metrics

For binary classification tasks, we compare the model's prediction to the true label. We can categorize the result into four types depending on the correctness and the class of the example. For a positive example, a correct prediction is referred to as a true positive (TP) and a misclassified example is called a false positive (FP). FP is sometimes referred to as type I error. For negative examples, a correct prediction is called true negative (TN) and an incorrect prediction is a false negative (FN). FN is also referred to as type II error. Since ML is a culture 2 according to Breiman's classification, its usefulness heavily depends on the

choice of metrics. The most straightforward metric is classification accuracy, where we compute the proportion of the correct predictions across both classes.

$$Accuracy = \frac{T_P + T_N}{T_P + T_N + F_P + F_N}$$

But accuracy may sometimes be less useful than other metrics, or even misleading. For example, sometimes the model's usefulness greatly decreases if it has a high type II error, while type I error is less costly. Other metrics can be used in such cases where we measure the correctness of TPs (Precision, also called positive predictive value or PPV) or the percentage of positive cases that were identified (Recall, also sometimes referred to as sensitivity) independently.

$$Precision = \frac{T_P}{T_P + F_P}$$

$$Recall = \frac{T_P}{T_P + F_N}$$

Higher interest in the positive examples may be more important in cases of disease and fraud detection, where FPs cost further investigation to verify the error, but the cost of FNs are higher. In cases where there is a higher interest in the negative cases, we can use specificity and the negative predictive value (NPV)

$$Specificity = \frac{T_N}{F_P + F_N}$$

$$NPV = \frac{T_N}{F_N + T_P}$$

But even when both classes equally matter, accuracy may be misleading if both classes are not roughly equally present. In imbalanced datasets (i.e., there is a majority class in terms of numbers), accuracy can give a false sense of high performance. For example, if the negative class represents 95% of the cases, a model giving a constant negative prediction would be 95% accurate. In these cases, it is better to use the F score:

$$F_\beta = (1 + \beta^2).\frac{Precision.Recall}{(\beta^2.Precision) + Recall}$$

where β is a positive floating point parameter. A special case is when β is equal to one, also referred to as F1 score. It computes the harmonic mean of the precision and recall, and thus it is less sensitive to class imbalance while putting equal emphasis on both positive and negative classes

$$F_1 = \frac{Precision.Recall}{Precision + Recall}$$

When β is 0, we consider only the precision, and when β is infinity, we consider only the recall.

1.3.4 Model Explanation

A major benefit from having accurate models is reverse engineering their insight about the problem. Shedding light on such insight is especially challenging when the phenomenon is highly non-linear or when the data does not have an intrinsic structure, such as images. There are two things that we would want to explain: a local explanation in which we want to understand why a model behaved in response to a specific input, and a global explanation in which we want to understand how the model sees the data. For local explanation on such unstructured data, work has focused on studying the strength\path of the signal coming from the input that contributed most to the output. Examples of such approaches are occlusion sensitivity (Habibi Aghdam & Jahani Heravi, 2017), Grad-CAM (Selvaraju et al., 2017), saliency maps (Simonyan et al., 2014), and integrated gradient (Sundararajan et al., 2017) which averages the gradient of the path of a certain input all the way to the output.

Chapter 2 Research Questions and Hypothesis

There are unanswered questions about both meiotic recombination and best practices for building and interpreting deep learning projects for genomic sequences. In this work, we will work on investigating if the prediction of meiotic recombination hotspots in humans can be done using deep learning. Specifically, we will train deep learning models whose inputs include the DNA sequence, and their output is the binary classification. The binary output is whether the input DNA sequence is a meiotic recombination hotspot or not. Such an approach can help researchers further understand the meiotic recombination process in a non-invasive manner and incur cheaper costs. Since different PRDM9 alleles recognize different hotspot regions, we will build separate models for combinations of PRDM9 alleles. We will also investigate if different PRDM9 alleles have varying difficulties for a model to learn.

We will investigate if a successfully trained model can provide more biological insights than what is already known from model-based approaches. For example, if it will be able to learn the reasons behind different behaviors across the chromosome, like chromosome 19's distinct patterns than the rest of the genome. To analyze such a question, we will investigate if adding the sequence's positional information as an input feature to the network would increase the predictive power or not. Next question to address is the effect of the input sequence length. Since hotspots vary in length, an important question to examine is the effect of the input sequence length on the network's learning.

We will investigate if introducing a bias in the train/validation splits influences the model's performance. We introduce such a bias in different ways. First, we will investigate the effect of training a model with sequences that are non-randomly sampled, for example training with sequences mostly from the extremities of the chromosome and testing on the examples coming from the mid-sections. This parallels the biological knowledge that hotspots occur more towards the telomeres. We will also contrast the difference between a total random sampling of the train/validation set versus sampling from each chromosome independently. This strategy preserves the exact train/validation ratios per chromosome.

Finally, because of the biological importance of DNA motifs, we explored extracting the learnt motif by the network by investigating the method provided by Alipanahi et al. (Alipanahi et al., 2015) versus simply visualizing the CNN weights directly.

Chapter 3 Methods

3.1 Used Dataset

The dataset we used was generated using the SSDS method from Pratto and colleagues (Pratto et al., 2014) (NCBI Accession: GSE59836). It targets the DMC1 protein in five unrelated male individuals. The data is made publicly available by the authors in a ".bed" format file. This format is widely used to store information about genomic intervals. It must contain at least four columns. The first column contains the chromosome ID. Second and third columns define the interval's start and end. The remaining one or more columns describe target values for the model. The provided .bed file was derived from the peak-calling of the SSDS experiment. DMC1 SSDS is a good proxy for the frequency of hotspots, despite that the count at a specific hotspot can be affected by other factors as well such as ssDNA intermediates life cycle (Pratto et al., 2014). These individuals have different PRDM9 alleles. Two are homozygous $PRDM9_A$ allele (referred to AA1 and AA2 individuals), two heterozygous $PRDM9_A$ and $PRDM9_B$ alleles (AB1 and AB2 individuals) and one heterozygous $PRDM9_A$ and $PRDM9_C$ alleles (AC individual). The dataset provides a high-resolution map of meiotic hotspots. It identifies about 39,000 hotspots per individual, about double the number found in mice by the same research group (Brick et al., 2012). The signal strength was much higher on sex chromosomes than on autosomes. In autosomal chromosomes, the counts varied in a range that is 3 orders of magnitude between the weakest and strongest signals. There are about 100 hotspots having stronger signals than any previously known hotspots. The study mapped the SSDS result to the hg19 reference genome, and therefore we used this human reference genome to build the DNA sequence of hotspots. The dataset also provides extra information, for example AA hotspots, which is the merging of the hotspots AA and AB individuals.

3.2 Software

Machine learning development process is still a work in progress, with a lot of missing best practices. For example, Gartner, a leading research and consulting company, foresees that

85% of artificial intelligence projects will not be successful for a variety of reasons (*Gartner Says Nearly Half of CIOs Are Planning to Deploy Artificial Intelligence*, n.d.), including data problems such as bias or because of the algorithms. This problem in the industry is also present in academia, with one notable side-effect known as the reproducibility crisis in machine learning. Therefore, a significant effort in the project was put into robust software development. One end goal was to increase automation for the project, allowing faster experiment cycles and more time and flexibility for results analysis. Other goals included increasing the ease of reproducibility and code reuse in future projects. The goal of being able to decrease the time between two consecutive experiments was especially important, as one of the initial goals was to build a custom dataset based on a collection of bed files in a user-friendly manner. We based the software design on the principle of separating the code from the experiment configuration. This means that the code will not define any values or constants; these are defined in external files. This separation means that the same code can be reused in multiple different projects, and reproducing an experiment requires only the configuration files (and of course using the same version of the code, which can be kept track of using different approaches: for example, we can use a commit hash sha if the code is kept track of using a git-based system). In this context, we developed two sets of tools. The first set of tools is related to model training. This includes a tool for creating deep learning suitable datasets out of one or more bed files and another to train a model according to training parameters. We also defined some supportive tools for the scientific work to help with drawing conclusions. These supportive tools are for dataset exploratory data analysis and model results exploration. We go through these tools in the following sections.

3.2.1 Dataset Explorer

The aim of this tool is to increase the user's understanding of the input ".bed" dataset. This task can help with hypothesis generation and refinement. The tool shows some statistics about the dataset, such as how many intervals there are overall and the cumulative count of the interval's length. For datasets for multi-task models (i.e., defining multiple outputs for each genomic interval), the correlation of outputs is displayed as well. The tool is made of two parts. The first one is a script that analyses the input file and saves the results of the

analysis. Second is a dashboard to display the results of the analysis. The reason for separating the analysis from the display is to avoid recomputing the analysis every time the tool is used.

3.2.2 Dataset Builder

The Dataset Builder tool ensures that the file is self-contained. It will save the dataset's metadata entries, such as creation date, information about the conversion between the bp and its numerical encoding. It also enforces saving a separate testing dataset, which should not be used until the final stages of the project.

Bed file data is not suitable for direct use with ML models. One reason is that the intervals do not have a fixed length, and most machine learning models cannot handle such variable length inputs. Therefore, this tool converts the input bed files into a format that is suitable for ML. The tool expects to be a tab-separate-value ".tsv" format (".bed" files are ".tsv" based), containing at least all 4 mandatory columns.

3.2.2.1 Negative Examples

Another problem with bed files is that they define only positive intervals. We cannot use the control data provided by Pratto et. al (input DNA) to define negative examples as these data only measure noise in the experiment, as this is generally a sample that has been cross-linked and sonicated but not immuno-precipitated. In our case, negative examples are all the remaining regions of the genome. However, an ML model also needs a negative class in order to learn the task. Thus, the tool will sample negative examples. The sampling of negative examples is not totally random, rather, we control it to be within a certain distance from the nearest positive example. The tool can save multiple negative examples based on different ranges from the positive regions.

There are more requirements for the negative examples. A negative example must not overlap with a positive example interval as defined in the original bed file, and not the positive example defined in the machine learning dataset. This distinction is important, because there may be instances where the original interval is longer than the sequence

length provided to the tool. This risks a stretch of a positive sequence that may end up as a negative example.

Another requirement is that we need to control the number of generated negative examples. The tool can accept a direct number of negative examples to generate, or a percentage of the positive examples, or it can automatically balance the number of negative and positive examples to be closely equal. The percentage option is useful for multiple-output datasets, where there are a lot of negative examples per output indirectly defined in the dataset. This happens when a positive interval in one output is negative for some of the other outputs. So adding a certain number of negative sequences for all outputs may be desired.

3.2.2.2 Separating SubDatasets by Chromosome

For both training and testing datasets, each chromosome's data is saved separately. The reason behind this decision is that, in many instances, the behavior of chromosomes could be different, such that separating the data of each chromosome can help with designing more biologically relevant experiments.

3.2.2.3 HDF5 Dataset

The final output of this tool is an HDF5 file. The aim of the HDF5 format is to contain all needed information, which allows enough flexibility for later stages of the software. HDF5 files are binary files, with a structure similar to that of a folder organization on a hard drive. One advantage of HDF5 files is that they appear as a single file for the operating system, but internally they can host multiple datasets and their metadata. The metadata is a python dictionary saved inside the HDF5 file.

The root directory of the HDF5 file contains two folders, one for training data and another for the testing data. Each folder will contain the data separated by chromosome. We separate each chromosome's data into three HDF5 datasets. The first one contains the model's input, which is an integer encoded DNA sequence. Second dataset is the model's output. The third table contains some metadata about the examples. For example, it includes the start and end indexes for both the original positive region and that of the ML dataset.

3.2.3 Model Trainer

Once we generated the HDF5 dataset, it is ready for modeling. What comes next are scripts to load and prepare the data from the HDF5 (Dataloader) and the model's training loop. They can work in multiple different ways, depending on the configurations we passed to the scripts.

3.2.3.1 Dataloader Module

The dataset configuration file includes information such as which outputs to use for the model (An HDF5 dataset can define multiple outputs, however, we may decide to use only a subset of these). Note that discarding an output would not remove its positive intervals from the dataset, it will still be present during the training.

It also contains the training and validation ratios and the possibility of setting a random seed.

The dataset-loader will also take extra configurations. For example, it can create the reverse-complement of the sequences and pass them to the model as a second input, which is the most widely used approach in literature. The dataloader can also accept other custom transformations to the input, as long as we define them as a python function.

Another input that the data-loader can create is the embedding of the genomic location of the sequence. This is represented by a one-hot-encoded vector for the chromosome, and another value encoding the normalized location between 0 and 1 over the chromosome.

Other necessary parameters are defined in the configurations as well, like the model's batch size.

This module returns a dictionary containing the training examples by chromosome. The next stage of the pipeline will split the dataset into training and validation datasets. The tool supports multiple ways to split the data. First one is "Whole Chromosome Shuffle", where we shuffle each chromosome independently then we sample the training and validation set. The second strategy is "Whole Genome Shuffle", where we shuffle the dataset as a whole, then we create the splits. This can lead to a slight imbalance in terms of seeing examples from specific chromosomes, especially the smaller ones. Third strategy is "Partial Chromosome

Contiguous" (figure 3.1), where we split the of each chromosome into three equal parts without shuffling. This means that at each fold, the model would not see one contiguous portion of all the chromosomes. For example, in the fold where the validation set is the middle fold, the model will be trained only on extremities data, and would be tested over the examples that are closer to the centromere. The final strategy, "Partial Chromosome Contiguous Alternate" (Figure 3.2) is very similar to the previous one, except that we alternate the validation set for each chromosome. For example, in the first fold we validate on part 1 of chromosome 1, part 2 of chromosome 2, part 3 of chromosome 3, part 1 of chromosome 4 and so on.

Figure 3.1 The strategy for training on whole contiguous sections of chromosomes. In each fold, the same region is kept constant across all chromosomes. Note that these boundaries are just for illustration, in reality since the split is done by an equal amount of examples, the middle section is much larger

Figure 3.2 *The strategy for training on whole contiguous sections of chromosomes, while alternating these regions from one chromosome to the next. Note that these boundaries are just for illustration, in reality since the split is done by an equal amount of examples, the middle section is much larger*

After the split, we shuffle the resulting dictionary to avoid any bias coming from training the data in order. The tool also supports K-Fold cross validation, in which case the training data will be split into k chunks, and the validation set will alternate across these. The tool creates an extra-validation set as well, other than the one used in early stopping. This extra validation set will be referred to as validation2 (Sebastien Lemieux, personal communication). If the configuration file defines the problem as a binary classification, the dataset builder will cast the outputs into a binary format, if not they are left as is.

3.2.3.2 Model Training

Once we construct the dataset, the next phase in the pipeline is to fit a model to the data. Any *Keras* model, the Python deep learning Application Programming Interface (https://keras.io/) model is currently supported; however, the tool can generate a deep learning model based on the passed configurations. The current function always forces the first layer to be a CNN, acting as a PWM motif scanner. The model creation also handles extra passed inputs. For example, it will share the initial CNN layer if the model takes the reverse complement sequence. In the case of passing the genomic coordinate location as an input feature, a small embedding layer will be added after the one-hot-encoded chromosome ID vector. Next, it can optionally add more CNN layers, optional GRU layers, and then at least one fully connected hidden layer before reaching the output layer. The configuration file also defines the activation functions, and the loss function and custom weighing the loss for each output. It also defines an early-stop patience, which is the number of training epochs without improvement over the validation set before we halt the training. We will refer to the validation set used for early-stopping as the "early-stopping validation set".

The final part of the training pipeline is the training and evaluation scripts. This part takes the already created dataset and instantiated model and will fit the model to the data. With K-fold validation, it is possible to launch training all folds in parallel using multiprocessing. However, it is important to be sure that all folds will fit in memory. The script has integrated support for MLFlow (https://mlflow.org/). MLFlow tracks the experiment's results and the model's artifacts. The model trainer reports to MLFlow each epoch's performance. The tool supports an optional early stop on validation, as well as MC-Dropout evaluation. In the MC-Dropout evaluation, we predict the whole dataset n times (n is defined in the configuration files) with the dropouts turned on. Once we train the model, the script can optionally run an evaluation script. The evaluation script is going to predict all examples (training, early-stop-validation and the validation2 set) and will save the output probability of the examples. It can also save the trained model for further inspection. For example, we can inspect the learnt motifs by the first convolution layer. The training configurations are also saved as an artifact in MLFlow.

3.2.4 Models Comparisons

This tool compares multiple models and performance. The tool is a dashboard that displays different metrics of each model. Like in the model performance tool, this tool compares the different models by the performance over the whole dataset, and by chromosome. Since the choice of the validation2 dataset is not random, this means that all experiments evaluated over validation2 are tested over the same examples. The tool currently supports binary models, computing the following metrics: accuracy, precision, recall, sensitivity, specificity, area-under-curve (AUC) and F1 score.

3.2.5 Predictions Visualization Tool

This module is a web browser-based explorer of the predictions of the model per chromosome. We built the tool using the *bokeh* library, and it takes a model's prediction along the HDF5 dataset and plots the performance metric as bar plots superimposed on each chromosome's layout. The tool draws the chromosome's ideogram layout with the cytobands for easier visual recognition of the region.

3.2.6 Motif Extraction

The first convolution layer of a CNN model applied to DNA sequences can be interpreted as a PWM. There are multiple ways to interpret the learned motif. The straight-forward way is to inspect the convolution weights directly. The tricky part about this approach is that the filters include negative numbers, which are not straight forward to interpret. First option for handling negative values is to just ignore the negative numbers and rely on the fact that if we use a ReLU activation, such negative numbers are not passed to the subsequent layers. Another approach to explore is the method introduced by Alipanahi et al. (Alipanahi et al., 2015). In this approach, we pass the convolution filter over the dataset (the referenced work used the training set), and extract locations with high activation. Then we perform a post-hoc multi-sequence-alignment over these extracted sequences in order to deduce the learnt motif. To implement this method, we created a keras model with a single convolution layer, and we passed the learnt weights to this new model. After, we predicted the training dataset, and looked at the activation map, which is the layer's output after the activation function. We

extracted all the positions that were 2 standard deviations away from the mean, and then we used the index of positions with a positive activation and extracted its DNA sequence from the input example. We accumulated all the activations of the training examples then we performed a multiple-sequence-alignment using the biopython package. We display the results using sequence logos (Schneider & Stephens, 1990), where the numerical value of each nucleotide is replaced by a letter that has a height proportional to the numerical magnitude.

3.3 Experiments

The goal of the experiments is to test our hypotheses to assess both the biological and informatics questions. The informatics questions are the benefits of using longer or shorter sequence lengths, and which dataset shuffling approaches yielded better results and whether using the genomic location of the sequence can increase a model's performance.

First, we ran a hyperparameter search to find the best model for each dataset. We have used the Google cloud hyperparameter optimization tool, which uses a Bayesian algorithm. We have run the search for 300 models, running a maximum of 3 trials in parallel. Then, we used the best hyperparameters of the AA allele model to investigate the effect of different questions on the hypothesis. The structure of the model is shown in figure 3.3. Each question was considered an experiment. Launching the experiments was done using bash scripts. The main experiment that was done was training the models of each dataset of the 4 high-quality individuals (AA1, AA2, AB1, AC) using 3 different input sequence lengths (800 bp, 2000 bp and 3500 bp). We decided to use these 3 sequence lengths based on three criteria. First, we found in the literature that often, the sequence lengths were less than 1kb, therefore we included one sequence length to be in that range. Second, the established average of hotspots size is between 1.5 and 2kbp in our dataset, so we used the upper limit. Third, after exploring the dataset, almost 99% of the examples were less than 3500bp. We included this value to inspect whether including this longer sequence towards the extremity of the hotspot would increase the performance or not. For each of these datasets, we applied the four different data splitting strategies described above (Whole chromosome, shuffled whole genome, shuffled chromosomes and contiguous chromosome) and switched on using the sequence

genomic location. Each model was trained on a 3-fold cross validation approach, and the performance of each fold was saved as well as the validation set. The resulting effort was training 4 x 3 x 4 x 2= 92 experiments, and each experiment had 3 models, one per fold, resulting in an evaluation of 276 models. The complexity of managing such a large number of experiments and models was eased by the developed software and open-source libraries used, such as MLFlow for experiment tracking. We based model comparison on the average performance of a 3-fold cross validation.

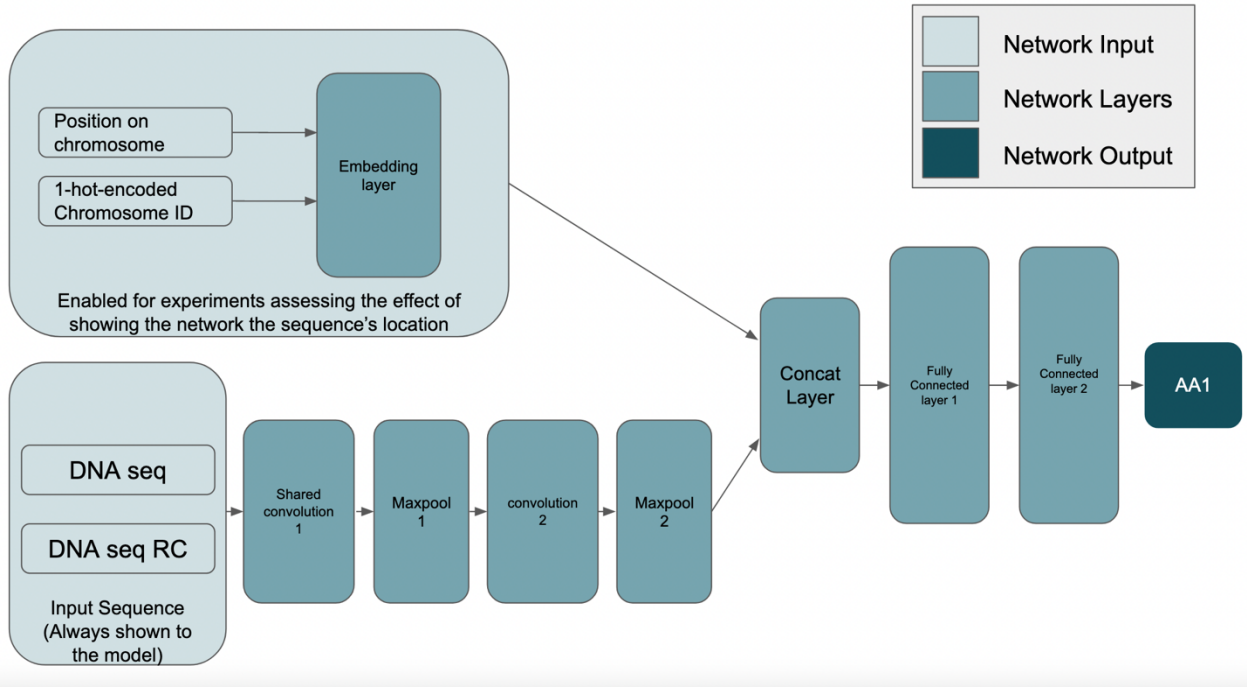

Figure 3.3 The final model architecture consists of a main input that represents the one-hot-encoded DNA sequence and its reverse complement. Both representations are scanned through the same convolution layer and their respective activation maps are added together. An optional second input that represents the sequence's input is turned on only for the subset of experiments where we evaluate the advantage of showing the model this feature. In that case, this input is concatenated to that of the convolution filters, and then they proceed to the fully connected layers. A final layer with a sigmoid activation function outputs the classification of the example.

Chapter 4 Results

4.1 Exploratory Data Analysis

Exploring the dataset, generated using the SSDS method from Pratto and colleagues (2014), has provided important information that guided the project. AA individuals shared about 89% of the hotspots locations, and they shared 88% of the hotspot locations with the AB individuals while sharing only 43% with the AC individual. Inspecting the target columns' correlation (Figure 4.1) verified that the individual with the C allele (individual AC) had less overlap with the remaining individuals. Individual AB2 had much lower correlation strengths (around half the strength) to other columns when compared to AB1 correlation's strength to the same other columns. Even correlation between AB1 and AB2 was lower than AB1's correlation with AA1 and AA2, which was not expected from individuals with the same PRDM9 alleles. After inspecting the number of hotspots per individual (Figure 4.2), it was clear that AB2 was off distribution as it had a very low count of hotspots. Therefore, we decided to discard this individual from the modeling phase. Individual AB1 had the highest intersection with allele-A dependent hotspots (0.9 correlation).

The other important conclusion that came from the exploratory data analysis was related to deciding the input sequence lengths for the deep learning models. First, inspecting the length of hotspots per chromosome revealed that despite the presence of very long hotspots, the average size was similar (Figure 4.3). Further investigations over the cumulative distribution (Figure 4.4) showed that about 99% of the dataset had a sequence length of 3537 bps. We concluded that a maximum length of 3500 should be enough to evaluate the effect of the input sequence length to the model.

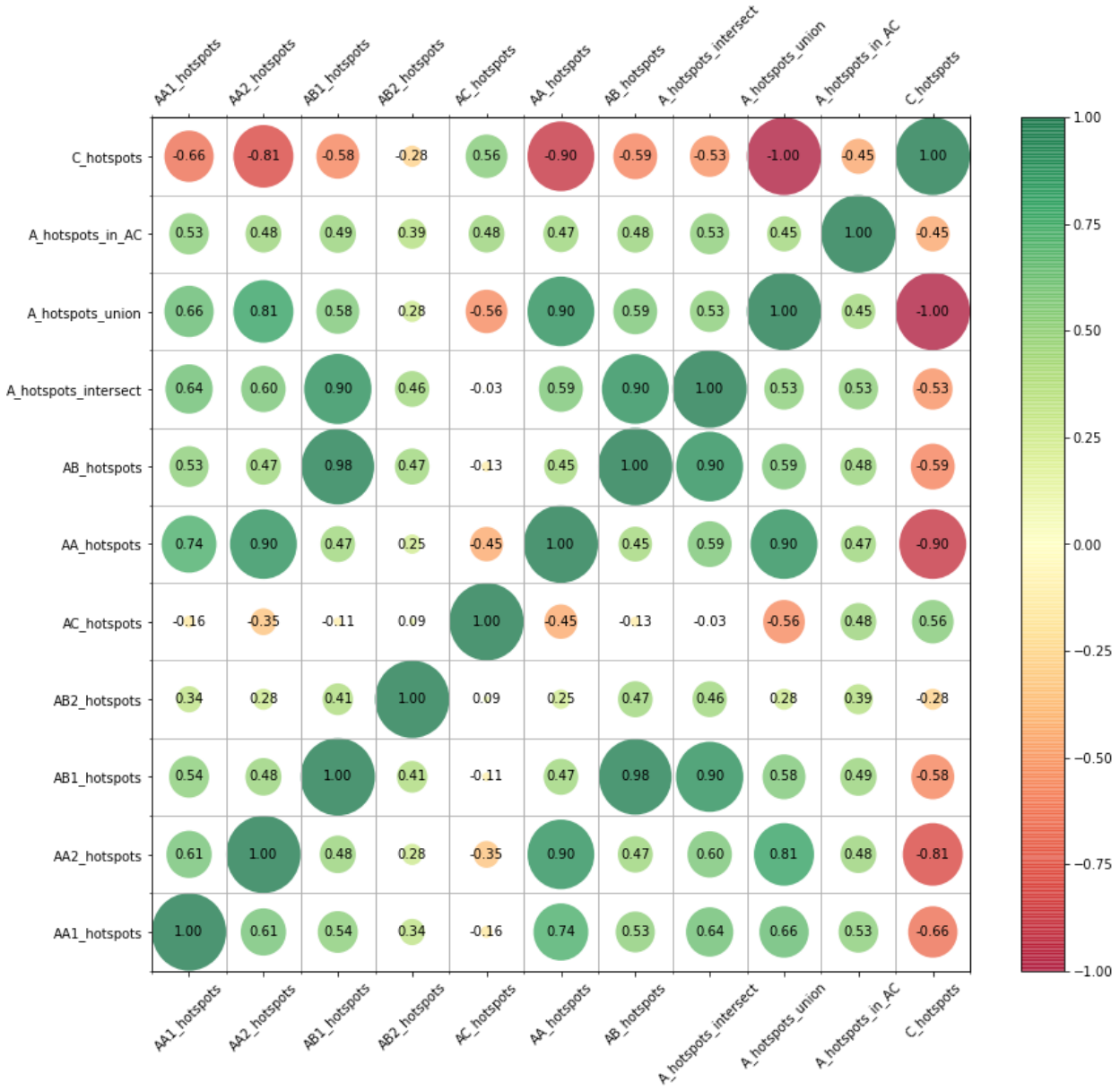

Figure 4.1 Correlation heatmap between different individuals in the dataset. We can see that heterozygous individual AC (row/column AC_hotspots in the figure) is slightly negatively correlated with the rest of the individuals. Allele-C dependent hotspots (C_hotspots entry in the heatmap) are highly negatively correlated with the rest of the non AC entries. Note that we discarded AB2 individuals due to its outlier number of hotspots

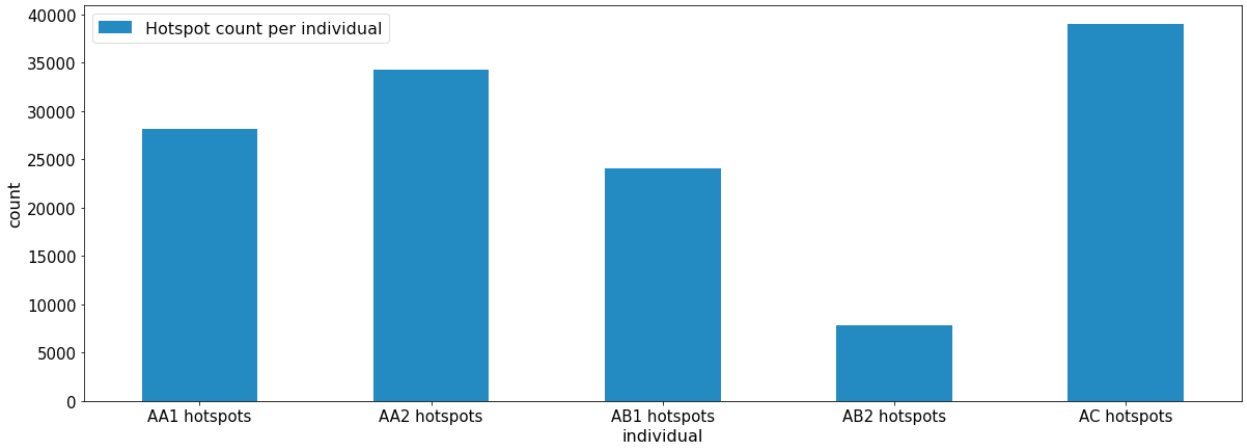

Figure 4.2 Number of hotspots per individual. Note the low count for individual AB2.

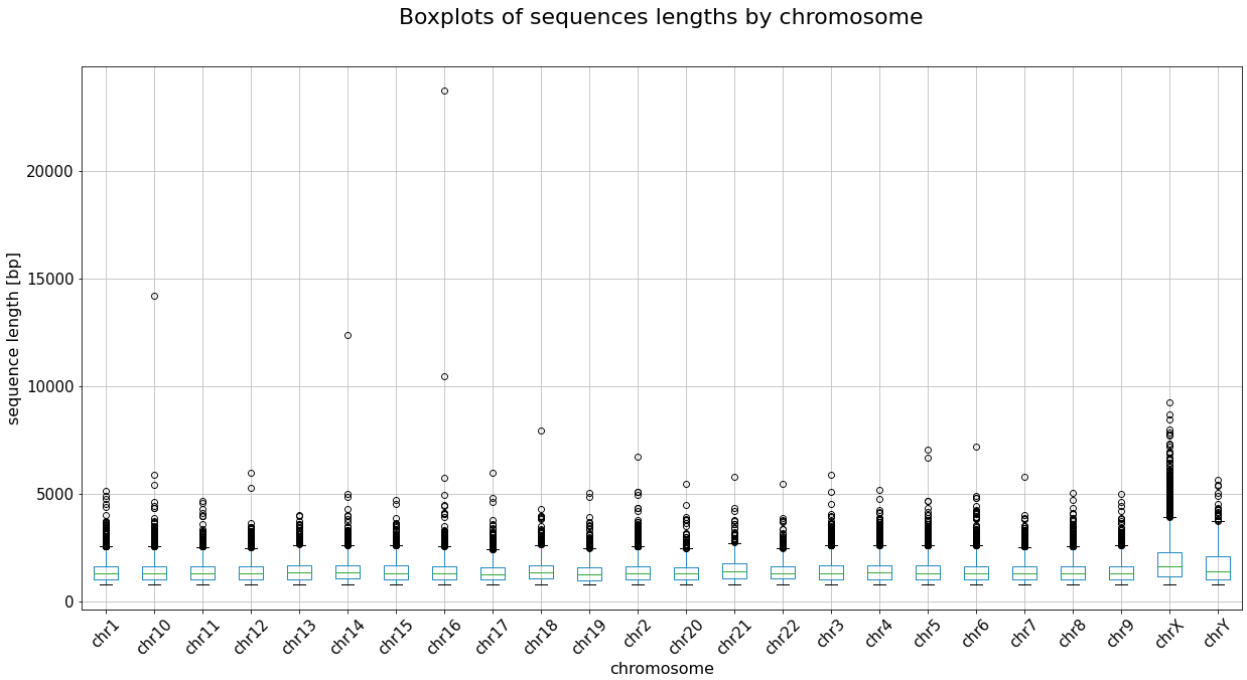

Figure 4.3 Boxplots of hotspot lengths per chromosome. Some outliers with hotspot lengths reaching over 20kbps exist. Chromosome X has the highest number of outliers.

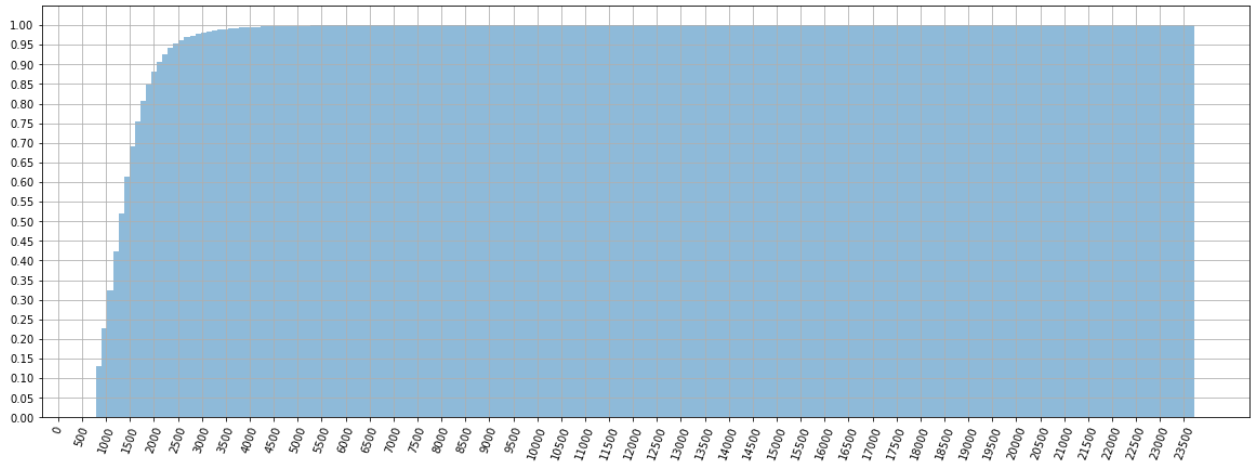

Figure 4.4 Cumulative distribution of hotspot lengths. The 99th percentile is equal 3537.9 basepairs

4.2 Hyperparameters Search

The best model in terms of architecture was on the simpler side of the search space. The input convolution filter size was 30bps and using a recurrent layer did not give an advantage, so we opted for the simpler model using only two convolution layers. Providing the model with both the input and its reverse complement at once seemed to accelerate the learning process, although passing the reverse complement sequences as extra examples still yielded comparable results.

The final model used to evaluate the performance over different datasets had two convolution layers followed by two fully connected hidden layers. The first convolution layer made of 120 convolution filters of size 30 each. Following layer had a filter size of 5, and the fully connected layers had 250 and 400 neurons respectively.

4.3 Experiments Results

There are two main experiments in our study. The first one is to explore the best parameters to generate examples on which to train a model. These parameters are the input model sequence, length, the best train/test split strategy and finally if providing the model extra information about the genomic location of the input sequence would improve its predictive

power. The second experiment is to train a model on AA1 individual using these best parameters and to investigate what the model learns. As a side note, we ran the pipeline to the remaining individuals (AA2, AB1 and AC) and obtained some preliminary results. These were close in performance to the reported results of AA1, with variations are in the vicinity of 2%).

4.3.1 Overall Results

We evaluated the three parameters over two sets. The first set is the validation set that was used for early-stopping and the second is a test set (referred to as test2, to differentiate it from the held-out test set). The test2 set is every 10th example in the dataset before the train/validation split. Therefore, test2 examples are spread over the genome with roughly with the same locations of the original dataset. All possible permutations of these parameters were used, and for each of these sets we trained a 3-fold cross-validation and report the average of these folds.

4.3.1.1 By Sequence Length

Aggregating the results by the sequence length revealed the underperformance of the 800bps model with respect to the other two. The 800bps models still learned a lot about the task, achieving on average about 86% in terms of accuracy in both early-stop-validation and test2 sets. More importantly, 800bps models achieved an average of a little over 90% in recall, which means it learned to recognize most of the hotspot sequences. However, it is visibly worse than both other sets of models (Figures 4.5 and 4.6). Table 4.1 shows the best performance of a sequence length on the early-stop validation step, with respect to other parameters. The 800 bps never shows up, with the results being a mix between 2000 and 3500 bps. A similar conclusion on the test2 table (table 4.2) can be achieved. The one important exception to note is that the shorter sequences had a better precision performance (2000 bps in early-stop set and 800 bps in test2 set).

4.3.1.1.1 Validation

Table 4.1 Experiments Winners on the early-stop validation set, averaged over all individuals datasets. The input sequence length of 800 never outperformed the other two, showing that there is important information the model learns from that are located away from the hotspot's center

fold_fn_name	chrom_idx	lowest loss	highest accuracy	highest AUC	highest precision	highest recall
partial chrom contig	FALSE	3500bps	3500bps	3500bps	2000bps	2000bps
partial chrom contig	TRUE	2000bps	3500bps	2000bps	2000bps	2000bps
partial chrom contig alternate	FALSE	3500bps	3500bps	3500bps	2000bps	3500bps
partial chrom contig alternate	TRUE	3500bps	3500bps	3500bps	3500bps	3500bps
partial chrom shuffled	FALSE	3500bps	3500bps	3500bps	3500bps	3500bps
partial chrom shuffled	TRUE	2000bps	2000bps	2000bps	2000bps	3500bps
whole genome shuffled k fold	FALSE	3500bps	3500bps	3500bps	2000bps	3500bps
whole genome shuffled k fold	TRUE	3500bps	3500bps	3500bps	3500bps	3500bps

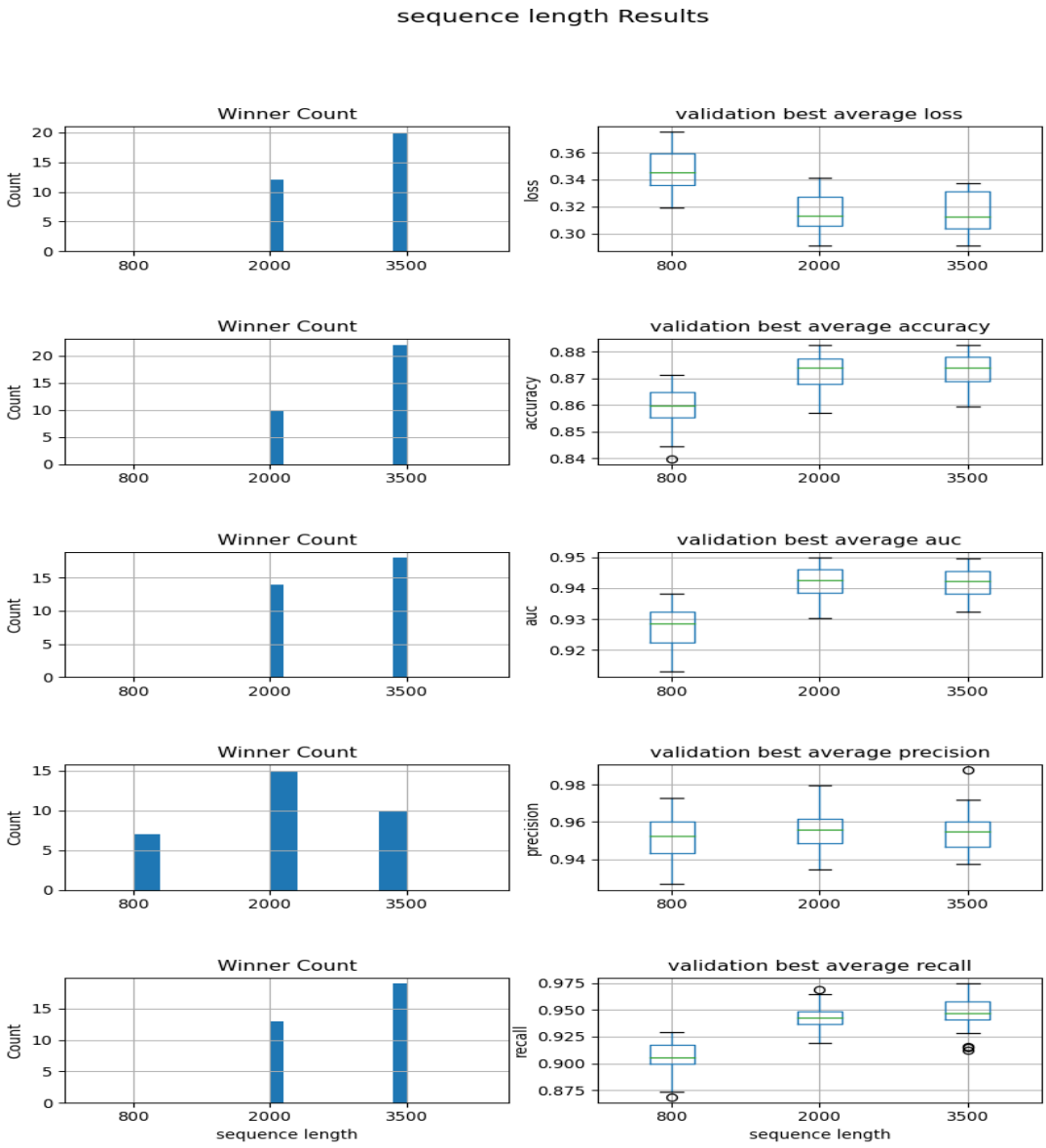

Figure 4.5 Models by input sequence length performance on early-stop validation set. The winner plots (Left column) represent the count of the instances when a specific input sequence length outperformed the other two when they had the same other two parameters. Each boxplot represents the performance of a set of models with a specific input length

4.3.1.1.2 Test 2

Table 4.2 Experiments Winners on the test2 set, averaged over all individuals datasets. Despite disagreeing sometimes with the early-stop-validation set, they both show that the 800bp input sequence never had a higher performance than the longer sequences

fold_fn_name	chrom_idx	lowest loss	highest accuracy	highest AUC	highest precision	highest recall
partial chrom contig	FALSE	3500bp	3500bp	2000bp	800bp	3500bp
partial chrom contig	TRUE	2000bp	2000bp	2000bp	800bp	2000bp
partial chrom contig alternate	FALSE	3500bp	3500bp	2000bp	2000bp	3500bp
partial chrom contig alternate	TRUE	3500bp	3500bp	2000bp	800bp	3500bp
partial chrom shuffled	FALSE	3500bp	2000bp	2000bp	3500bp	2000bp
partial chrom shuffled	TRUE	2000bp	3500bp	2000bp	2000bp	3500bp
whole genome shuffled k fold	FALSE	3500bp	3500bp	3500bp	2000bp	3500bp
whole genome shuffled k fold	TRUE	3500bp	3500bp	2000bp	800bp	3500bp

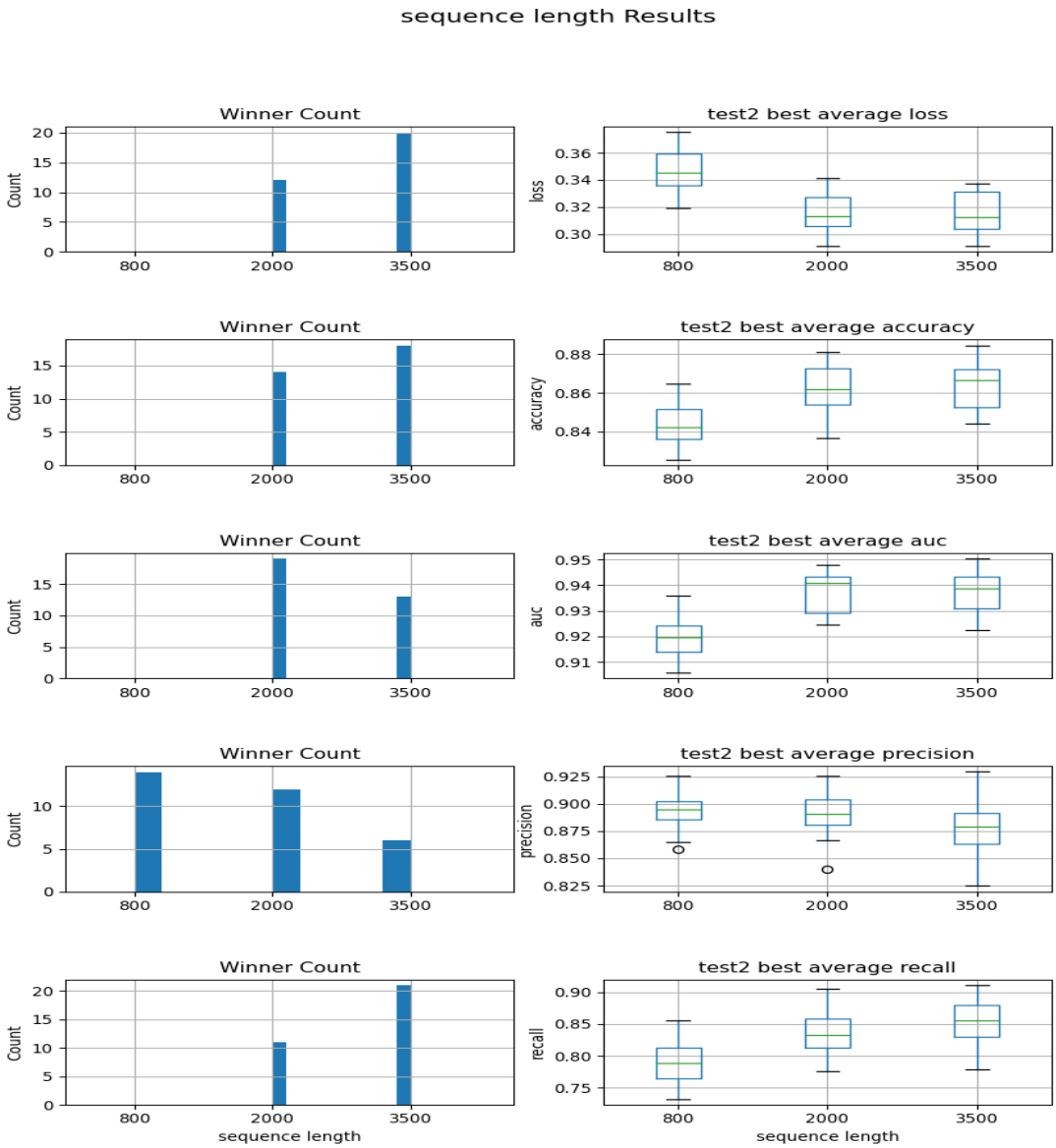

Figure 4.6 Results on test2 (non-early-stop) validation set. Like in 4.5, the left column represents the count of the instances when a specific input sequence length outperformed the other two when they had the same other two parameters. Each boxplot *represents the performance of a set of models with a specific input length. Note that for the loss plots, lower is better but in the remaining ones, higher is better.*

4.3.1.2 By Train/Validation Splitting Function

The next parameter to evaluate is the effect of the train-validation split scheme on the model's performance. We tested four strategies. First strategy is to trains on contiguous regions of a chromosome and test on another contiguous region as well (Partial Chromosome Contiguous). The second strategy (Partial Chromosome Contiguous alternate) is similar to the first one, except that we alternate the validation fold in each chromosome. Next strategy (Whole Chromosome Shuffle) shuffles each chromosome data and do the split by chromosome before merging the whole dataset and reshuffling again. The final one (Whole Genome Shuffle) shuffles all the data, then creates the split.

The results (table 4.3) favored the whole genome split, which is the classical way to split the dataset. One interesting result was that the model trained on the contiguous regions (strategy 1) had the best precision on the early-stop validation set and not at the test2 dataset (Figures 4.7 and 4.8). We analyze this difference in section 4.3.2

4.3.1.2.1 Validation

Table 4.3 Experiments splitting-functions winners on the early-stop validation set, averaged over all individuals datasets.

seq_len	chrom_idx	lowest loss	highest accuracy	highest AUC	highest precision	highest recall
800	FALSE	Whole Genome Shuffle	Whole Chrom. Shuffle	Whole Genome Shuffle	Partial Chrom. Contig	Whole Genome Shuffle
800	TRUE	Whole Genome Shuffle	Whole Genome Shuffle	Whole Genome Shuffle	Partial Chrom. Contig	Whole Chrom. Shuffle
2,000	FALSE	Whole Chrom. Shuffle	Whole Chrom. Shuffle	Whole Chrom. Shuffle	Partial Chrom. Contig	Partial Chrom. Contig
2,000	TRUE	Whole Genome Shuffle	Whole Chrom. Shuffle	Whole Chrom. Shuffle	Partial Chrom. Contig	Whole Genome Shuffle
3,500	FALSE	Partial Chrom. Alternate	Whole Chrom. Shuffle	Whole Chrom. Shuffle	Partial Chrom. Alternate	Partial Chrom. Alternate
3,500	TRUE	Whole Genome Shuffle	Whole Genome Shuffle	Whole Genome Shuffle	Whole Genome Shuffle	Whole Chrom. Shuffle

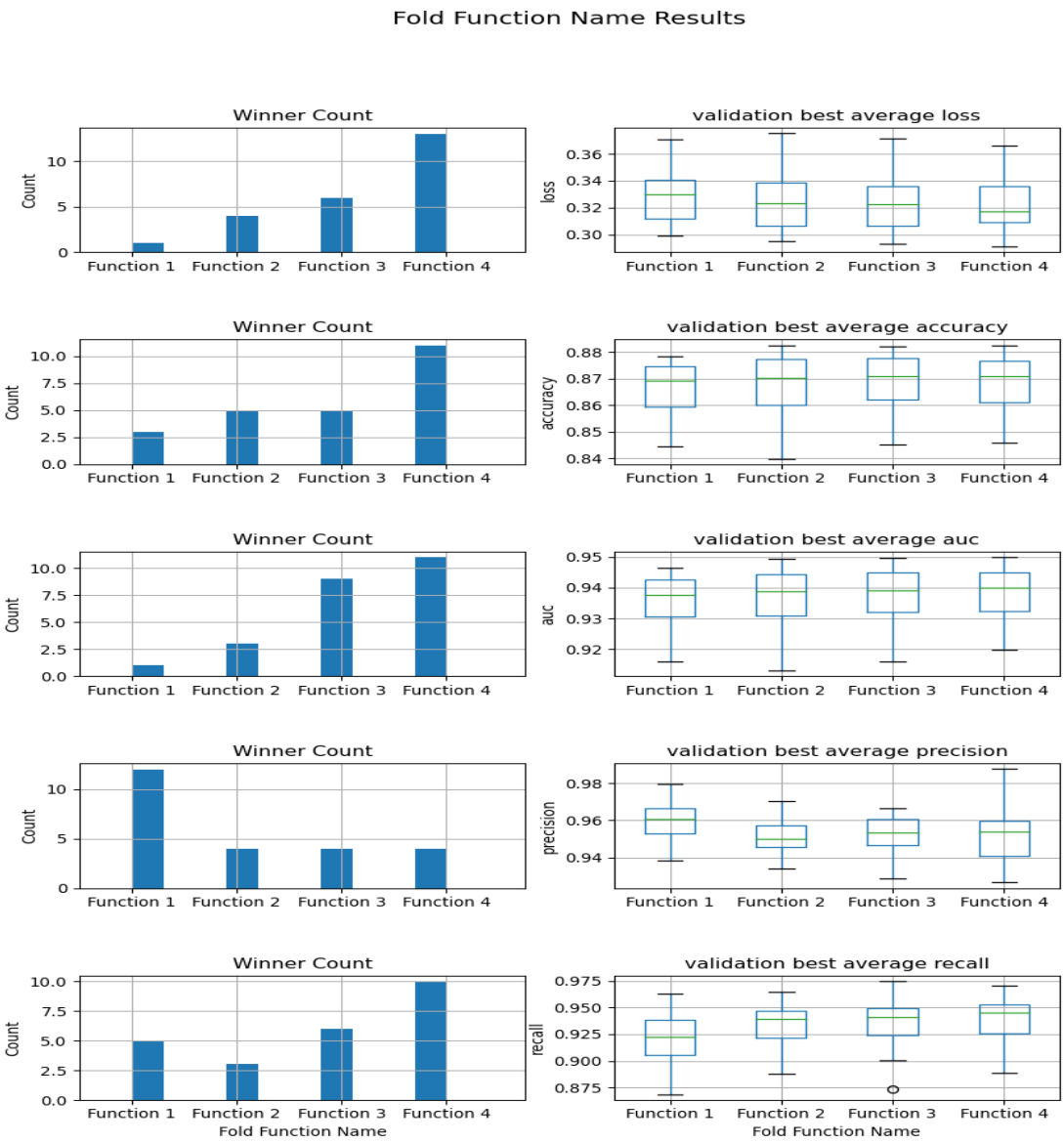

Figure 4.7 *Results on early-stop validation set. For figure clarity, the functions were renamed as follows: function 1 is the "partial chrom contig" in the table, function 2 is "partial chrom contig alternate", function 3 is "partial chrom shuffled" and function 4 is "whole genome shuffled k fold"*

4.3.1.2.2 Test 2

Table 4.4 Experiments splitting-functions winners on the test2 set, averaged over all individuals datasets.

seq_len	chrom_idx	lowest loss	highest accuracy	highest AUC	highest precision	highest recall
800	FALSE	Whole Genome Shuffle	Whole Chrom. Shuffle	Whole Chrom. Shuffle	Whole Genome Shuffle	Whole Chrom. Shuffle
800	TRUE	Whole Genome Shuffle	Whole Chrom. Shuffle	Whole Genome Shuffle	Partial Chrom. Alternate	Whole Chrom. Shuffle
2,000	FALSE	Whole Chrom. Shuffle	Whole Chrom. Shuffle	Whole Chrom. Shuffle	Partial Chrom. Alternate	Whole Chrom. Shuffle
2,000	TRUE	Whole Genome Shuffle	Whole Genome Shuffle	Whole Genome Shuffle	Whole Chrom. Shuffle	Partial Chrom. Contig
3,500	FALSE	Partial Chrom. Alternate	Partial Chrom. Alternate	Whole Chrom. Shuffle	Whole Chrom. Shuffle	Partial Chrom. Contig
3,500	TRUE	Whole Genome Shuffle	Partial Chrom. Alternate	Whole Genome Shuffle	Whole Chrom. Shuffle	Partial Chrom. Alternate

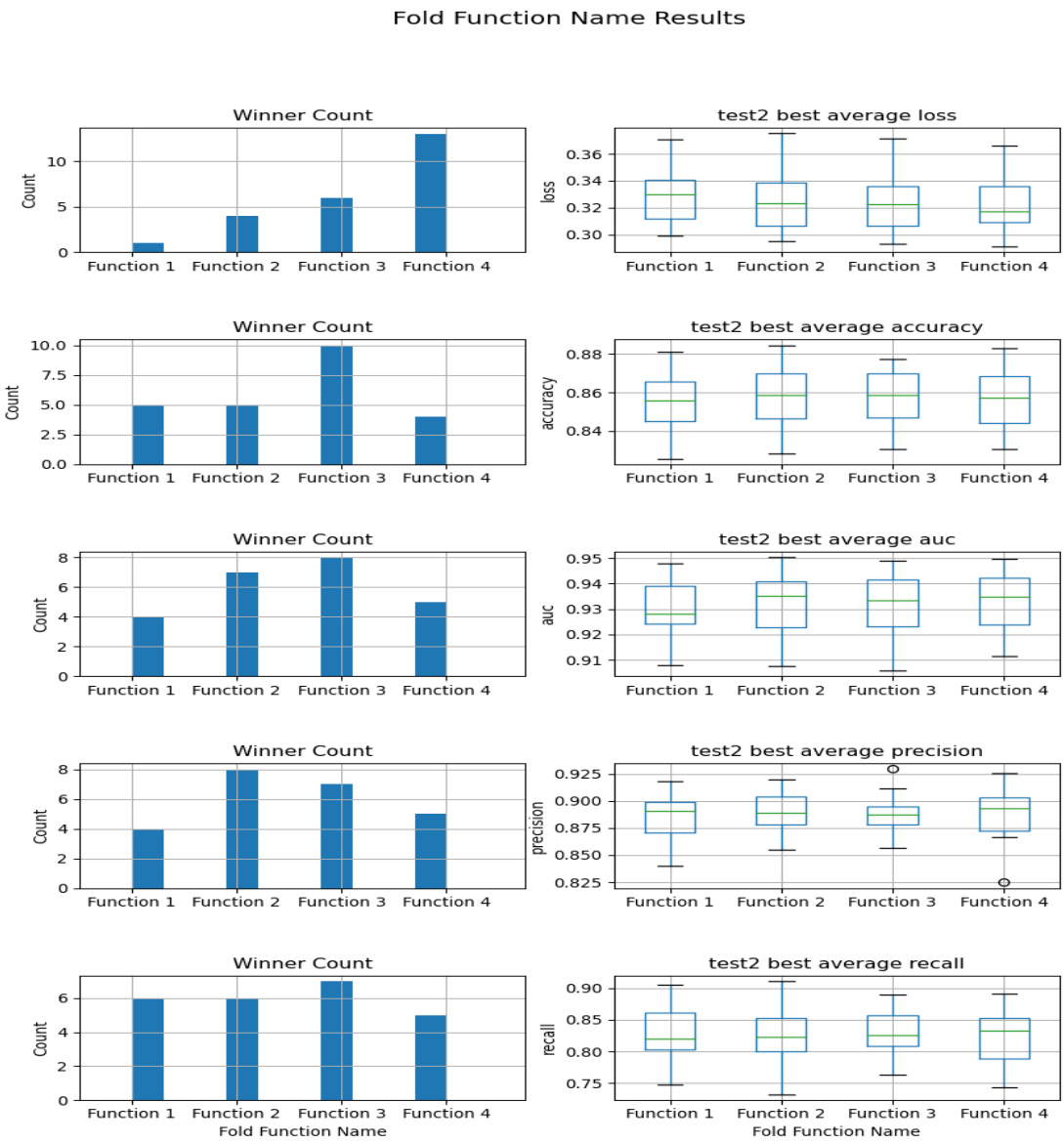

Figure 4.8 Results on holdout (non-early-stop) validation set

4.3.1.3 By Including Sequence Location

The final parameter is the inclusion of the location of the sequence. The model received a one-hot-encoded vector representing the chromosome ID, and another contiguous value

representing the index of the middle-point of the input sequence. The result (Table 4.5) shows that providing such information slightly helped the model improve its prediction. One interesting exception was that the partial chromosome contiguous strategy performed better without showing the location. The interesting thing about this strategy is that the model predicts sequences within locations that were never seen in the training dataset. That implies that the model has reached false hypothesis about the effect of such information.

4.3.1.3.1 Validation

Table 4.5 Experiments winners for showing sequence location, evaluated on the early-stop validation set, averaged over all individuals' datasets.

seq_len	fold_fn_name	lowest loss	highest accuracy	highest AUC	highest precision	highest recall
800	partial chrom contig	Loc not provided	Loc not provided	Loc not provided	Loc provided	Loc not provided
800	partial chrom contig alternate	Loc provided	Loc provided	Loc provided	Loc provided	Loc not provided
800	partial chrom shuffled	Loc provided	Loc provided	Loc provided	Loc not provided	Loc provided
800	whole genome shuffled k fold	Loc provided	Loc provided	Loc provided	Loc not provided	Loc not provided
2,000	partial chrom contig	Loc not provided	Loc not provided	Loc not provided	Loc provided	Loc not provided
2,000	partial chrom contig alternate	Loc provided	Loc provided	Loc provided	Loc not provided	Loc provided
2,000	partial chrom shuffled	Loc provided	Loc provided	Loc provided	Loc provided	Loc provided
2,000	whole genome shuffled k fold	Loc provided	Loc provided	Loc provided	Loc not provided	Loc provided
3,500	partial chrom contig	Loc not provided	Loc not provided	Loc not provided	Loc provided	Loc not provided
3,500	partial chrom contig alternate	Loc provided	Loc provided	Loc provided	Loc provided	Loc provided
3,500	partial chrom shuffled	Loc provided	Loc provided	Loc provided	Loc provided	Loc provided
3,500	whole genome shuffled k fold	Loc provided	Loc provided	Loc provided	Loc provided	Loc provided

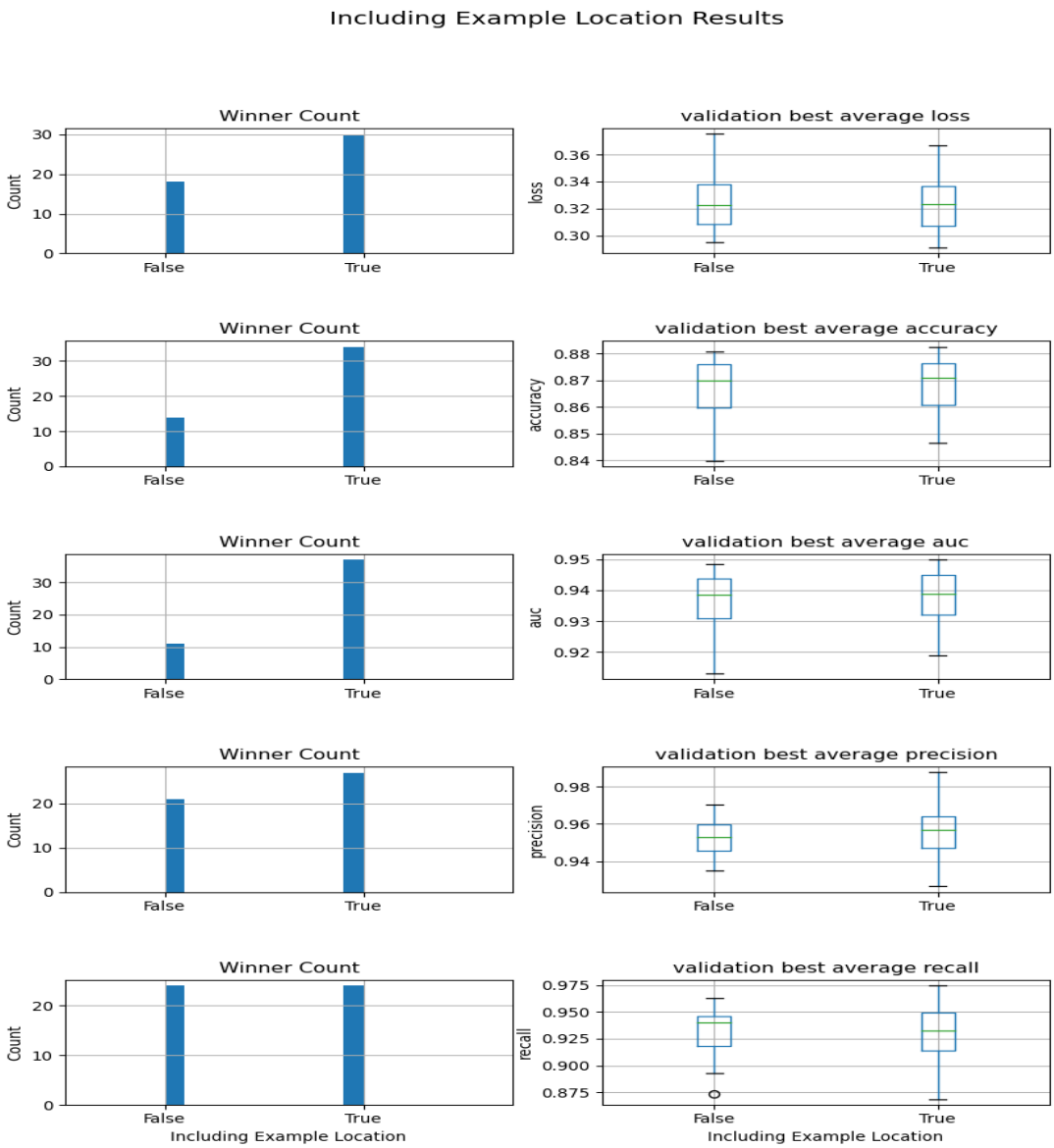

Figure 4.9 Results on early-stop validation set

4.3.1.3.2 Test 2

Table 4.6 Experiments winners for showing sequence location, evaluated on test2 set, averaged over all individuals datasets

seq_len	fold_fn_name	lowest loss	highest accuracy	highest AUC	highest precision	highest recall
800	partial chrom contig	Loc not provided	Loc provided	Loc provided	Loc provided	Loc not provided
800	partial chrom contig alternate	Loc provided	Loc provided	Loc provided	Loc provided	Loc not provided
800	partial chrom shuffled	Loc provided	Loc not provided	Loc provided	Loc provided	Loc not provided
800	whole genome shuffled k fold	Loc provided	Loc provided	Loc provided	Loc provided	Loc provided
2,000	partial chrom contig	Loc not provided	Loc provided	Loc not provided	Loc not provided	Loc provided
2,000	partial chrom contig alternate	Loc provided	Loc provided	Loc provided	Loc not provided	Loc provided
2,000	partial chrom shuffled	Loc provided	Loc not provided	Loc provided	Loc provided	Loc not provided
2,000	whole genome shuffled k fold	Loc provided	Loc provided	Loc provided	Loc not provided	Loc provided
3,500	partial chrom contig	Loc not provided	Loc provided	Loc not provided	Loc provided	Loc not provided
3,500	partial chrom contig alternate	Loc provided	Loc provided	Loc provided	Loc not provided	Loc provided
3,500	partial chrom shuffled	Loc provided	Loc provided	Loc provided	Loc provided	Loc provided
3,500	whole genome shuffled k fold	Loc provided	Loc provided	Loc provided	Loc provided	Loc not provided

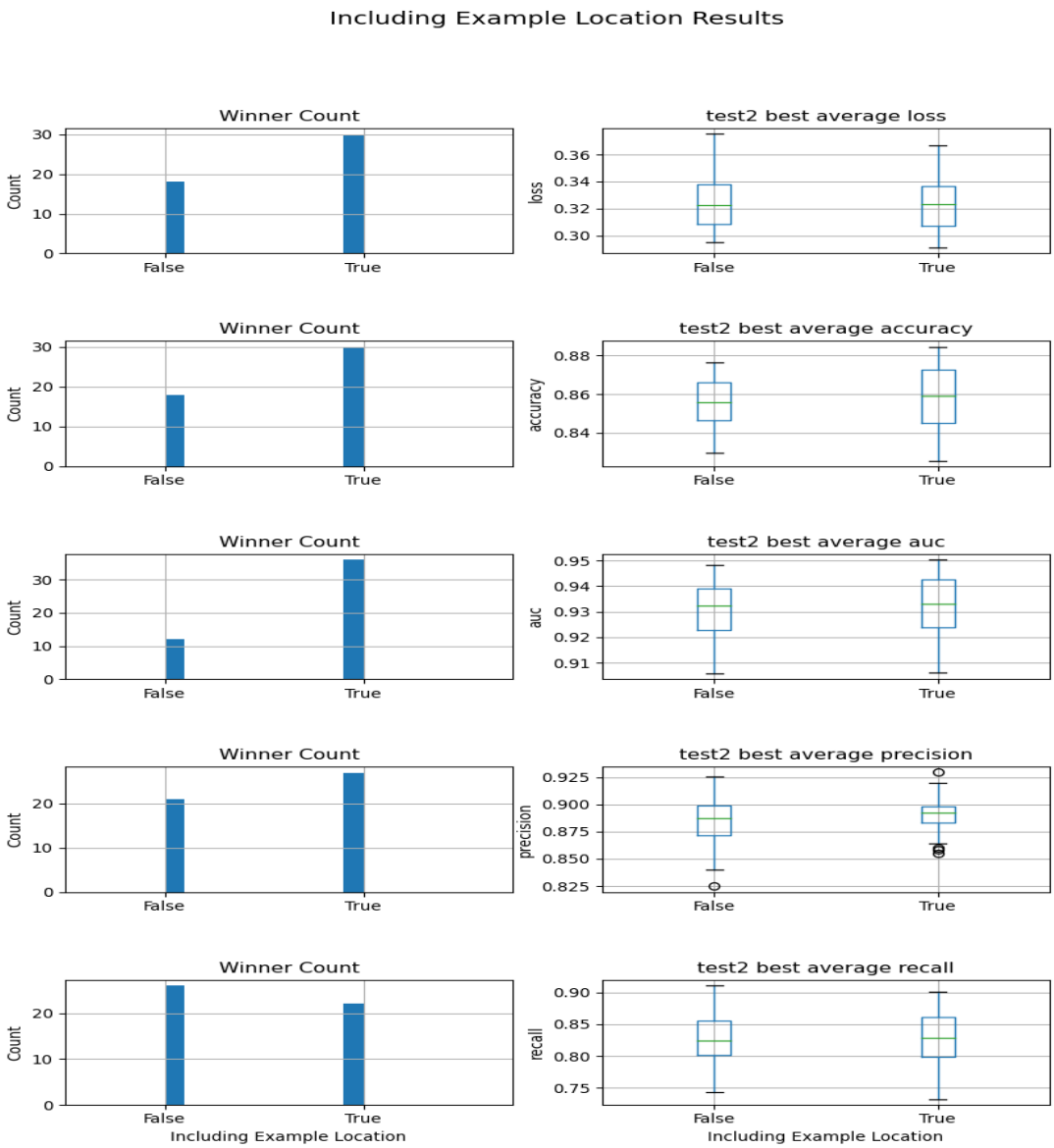

Figure 4.10 Results on holdout (non-early-stop) validation set

4.3.2 Importance of Test2 Set

The main motivation of using the test2 dataset was that since early-stopping is decided by the performance over the validation set, there is an indirect leakage in the metrics which

may lead to overoptimistic performance. Such over-optimistic performance can be observed, one example is figures 4.11 and 4.12. By contrasting the performances in these two plots, we can see that the performance of the test2 set is worse than that of the early-stop validation set. Another important observation is related to the "Partial Chromosome Contiguous" strategy. In the early-stop validation set, we can see how the middle fold (where validation was performed over the middle fold) shows an increase in performance over the two other folds. However, such an increase is not as pronounced in the test2 dataset.

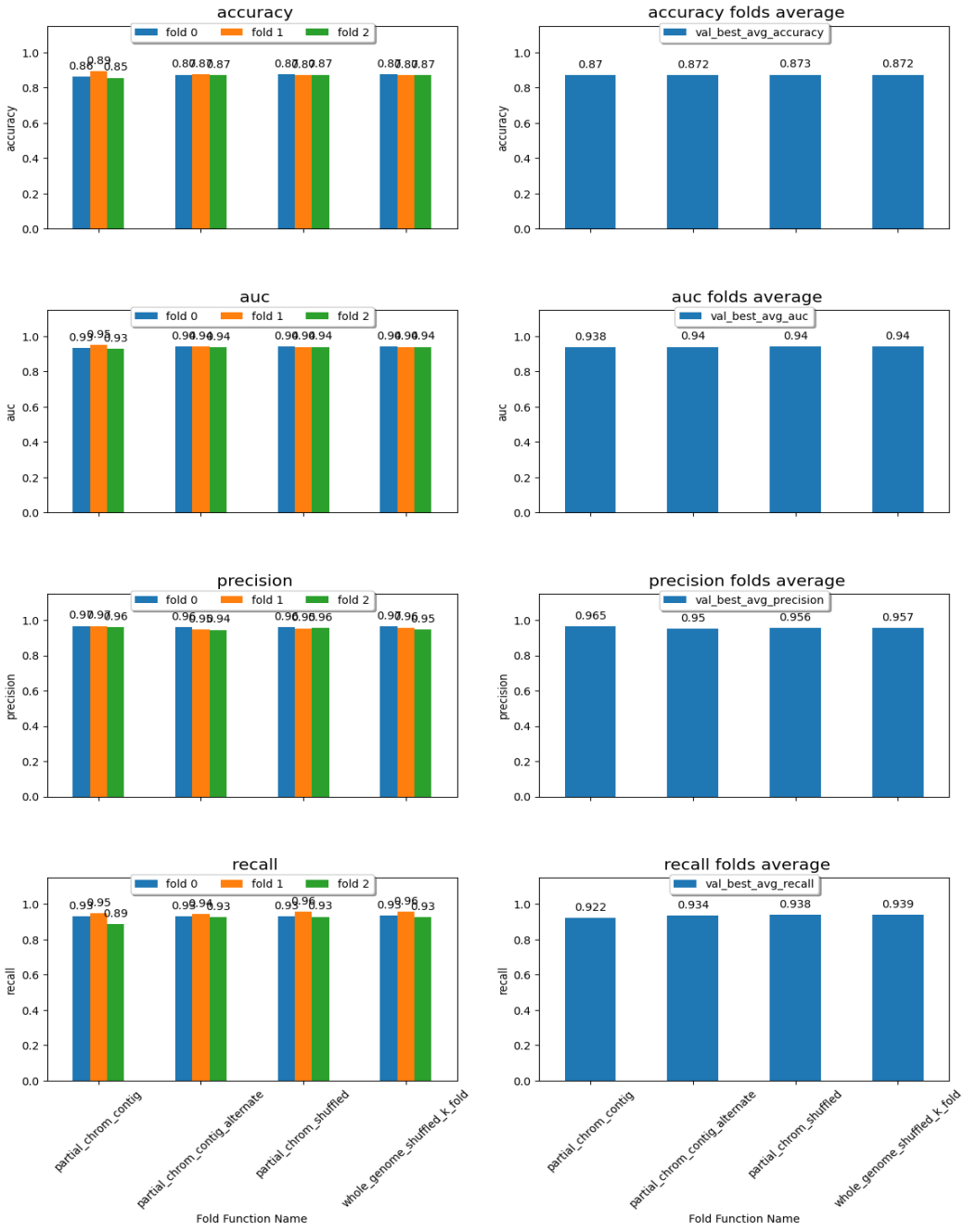

Figure 4.11 AA1 Dataset, validation fold performance of different splitting function

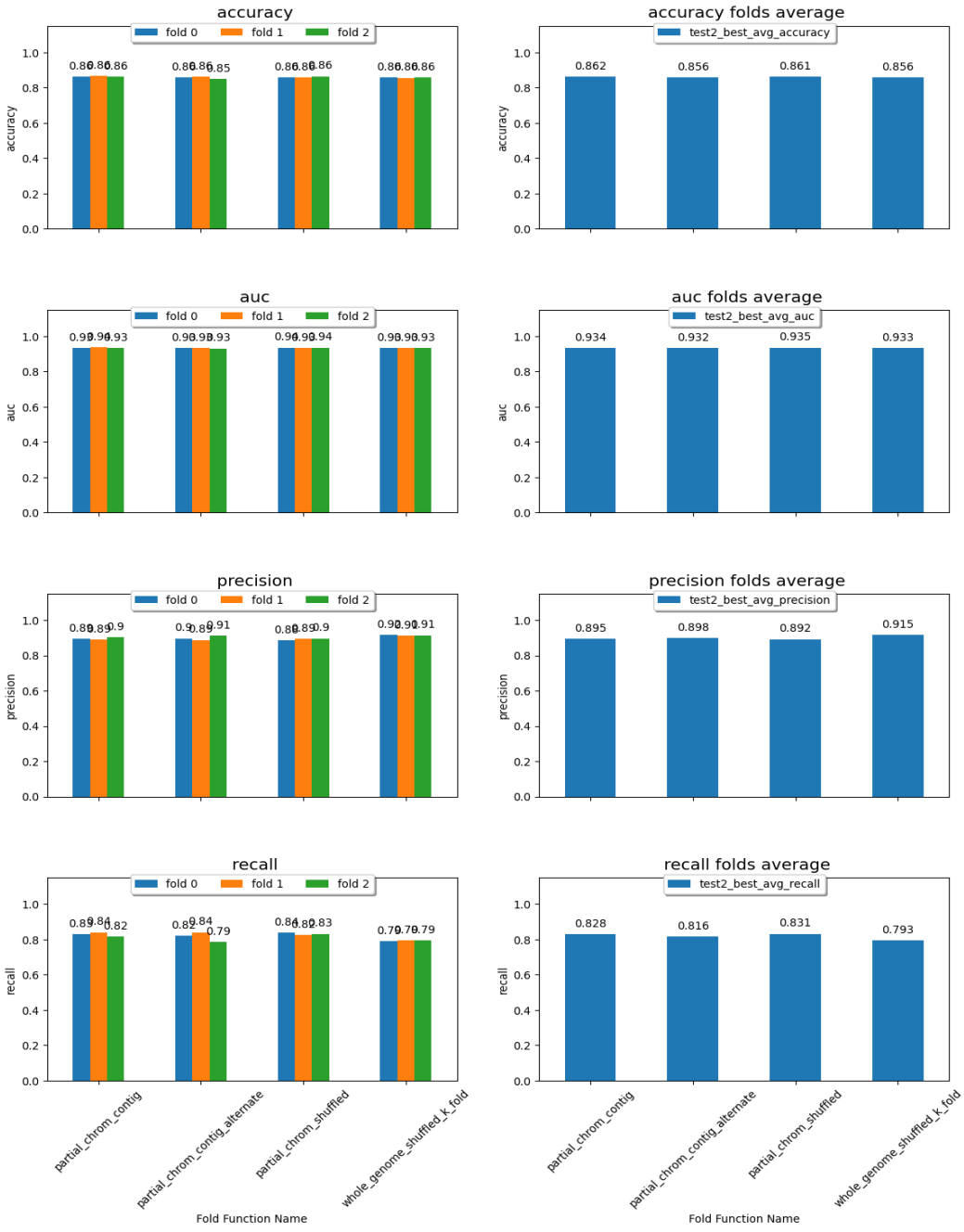

Figure 4.12 AA1 Dataset, test2 fold performance of different splitting function

4.3.3 Extracted Motifs

We worked on multiple ways to extract the learnt DNA motifs and contrasted the benefits of each. The intuitive way was to inspect the convolution filters values directly. Although in many instances we can clearly see the learnt motifs when plotting them as logograms, interpreting the negative values in the filter was the part the needed the most work. The straightforward interpretation of negative values is to ignore them, for the simple reason that we use ReLU as an activation function and therefore the subsequent layers will not be aware by these values anyway, meaning that they should not contribute to the prediction. However, closer inspection showed some interesting insights that we discuss in the next section

After performing the multiple-sequence-alignment of the sequences that activated each filter, we had a count matrix, and we can interpret such an output in many ways. The first one is the consensus sequence of all the positions, which we display at the top sub-image in the motif result figures 4.13 to 4.20. We also used the traditional PWM calculations, the count-matrix, the weight-matrix, the probability-matrix and the information matrix. These are the third, fourth, fifth and sixth subplots in the figures, respectively.

In the following figures showing the chosen learnt features, each figure is made of 6 subplots. Each subplot is in the logo form, constructed using the logomaker python library.

4.3.3.1 PRDM9

On a model trained on the AA1 individual, we ran our procedure and manually inspected the results. Four filters (Figures 4.13, 4.14, 4.15 and 4.16, were easily recognizable just by looking to the simple convolution weights when visualized as a logogram (second subplot). They represented the motif of the $PRDM9_A$ allele. In Figure 4.13, we can see the Myers motifs starting at the third position (index 2 on the x-axis). The learnt motif looks most similar to the weight-matrix (subplot 4), although the most interpretable plot was the information-matrix (subplot 6). One interesting observation that can be seen in this figure is related to the negative weights. Positions with high negative values for 3 basepairs corresponds to a very high value of the 4th basepair in the information-matrix plot. For example, at index 5,

the C bp has a height that is not too different from the ones before it, however the 3 other basepairs have a much higher negative value. Looking at the information matrix at these positions, we can see that the magnitude of the C is more proportional the negative values rather than to its own positive value. This can be a starting point for further work to inspect the meaning of negatively learnt values in the convolution filters as they potentially may indicate high confidence from the network.

In Figure 4.14, we can see a filter where the model has learned only the first half of the motif. The region preceding the CCxCC part seems distantly similar to the ZnF inferred sequence but dominated with thymine instead of Adenine or Cytosine. Figures 4.13 and 4.16 tell the same story but in the reverse complement sequences.

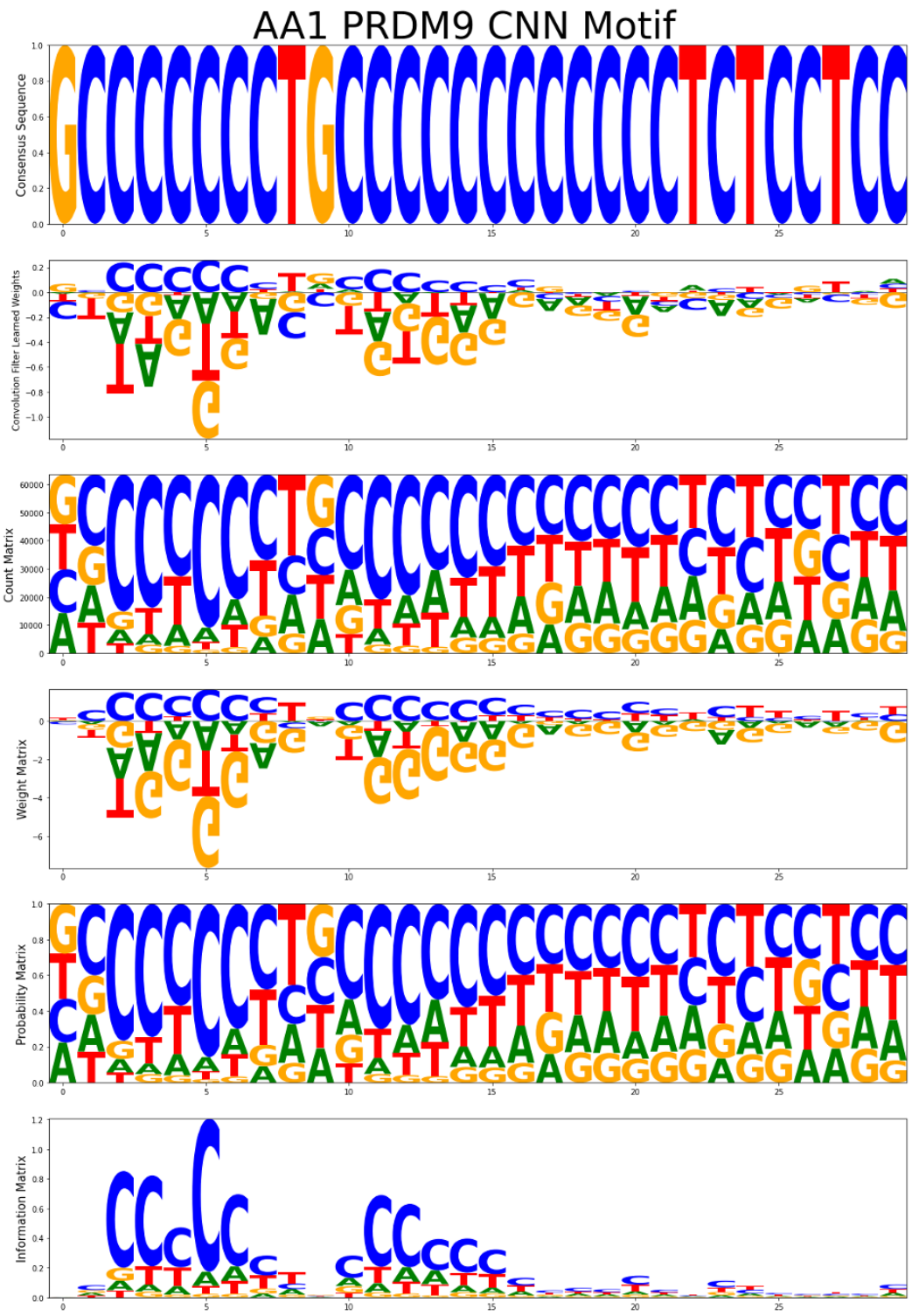

Figure 4.13 : Full Myer motif (PRM9$_A$ allele) learned by the model over the data coming from the AA1 individual

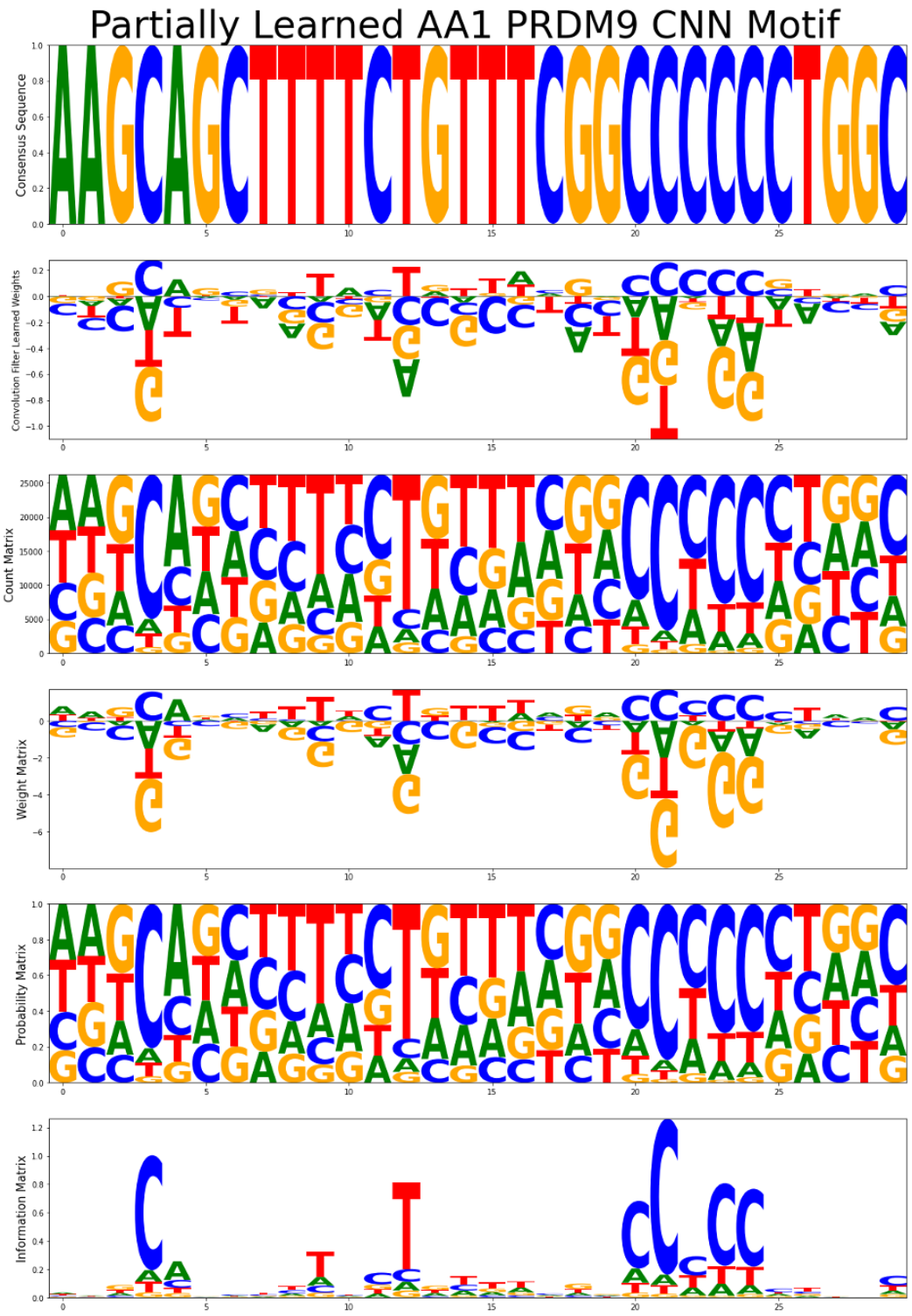

Figure 4.14 Partial Myer motif (PRM9_A allele) learned by the model over the data coming from the AA1 individual

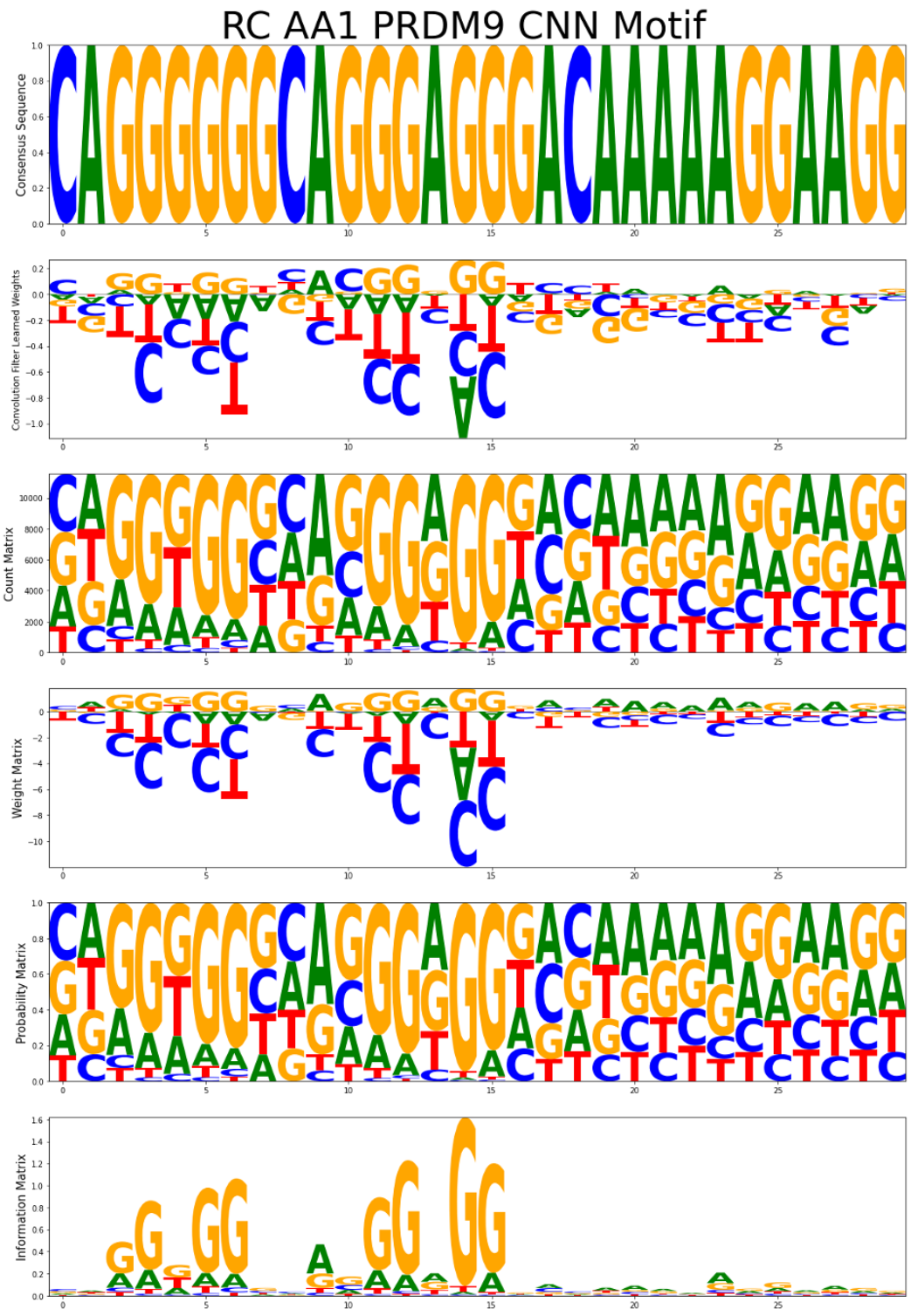

Figure 4.15 Reverse complement full Myer motif (PRM9_A allele) learned by the model over the data coming from the AA1 individual

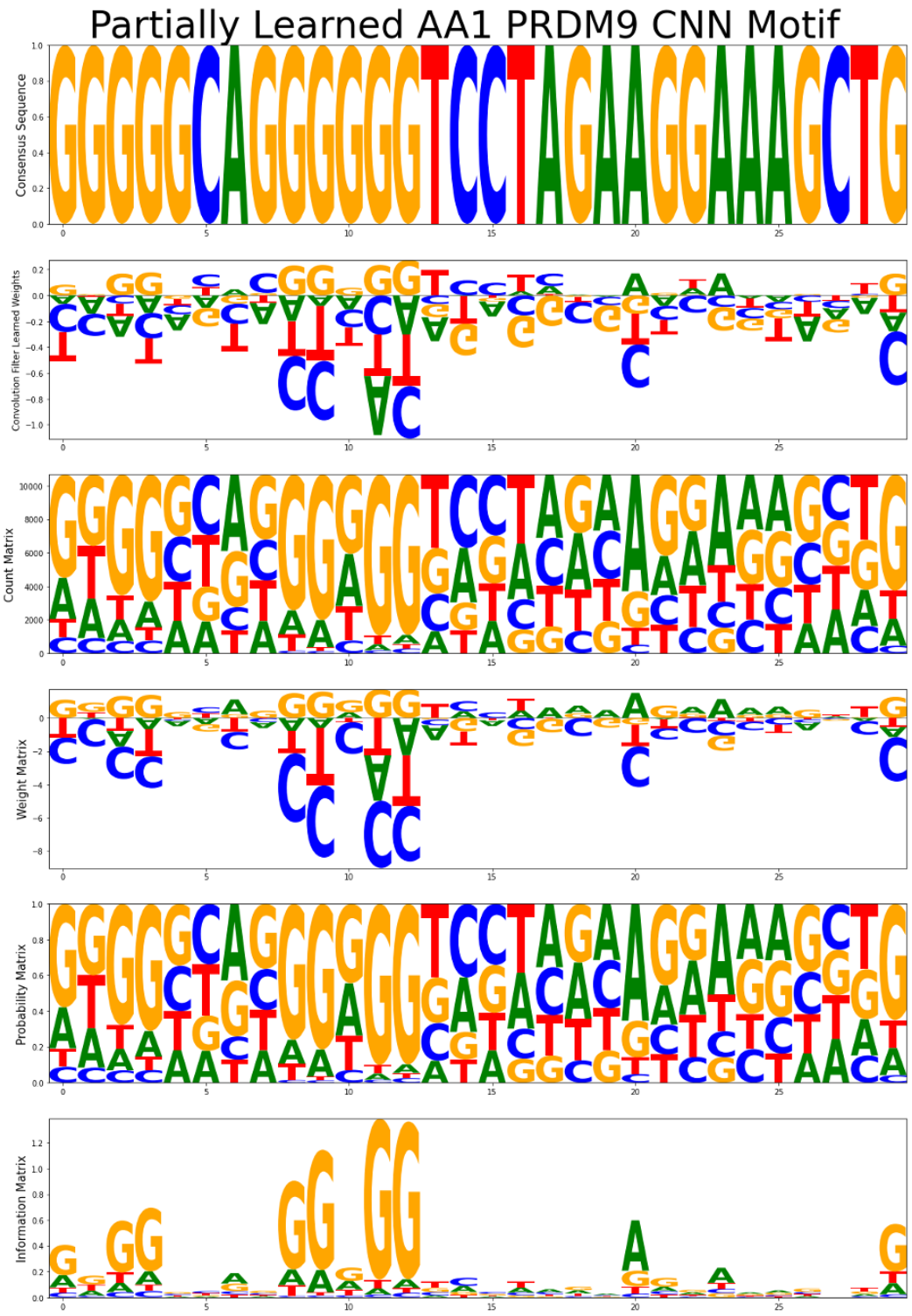

Figure 4.16 Reverse complement partial Myer motif (PRM9_A allele) learned by the model over the data coming from the AA1 individual

4.3.3.2 Other Genomic Features

The network was able to learn important features related to recombination as well. Interpreting filters that learnt such features is less obvious than inspecting the convolution filters, however they become clearer by looking at the information-matrix, but even then, it is open for interpretations. For example, in Figure 4.17, we can see that the motif is encoding some region that is rich in C, but it is less clear than the Myers motif convolution filters. However, the information-matrix hints to both C and G presence in those regions, despite having the prominent A at the 28th position. Figure 4.18 shows that the model has likely learned the same feature (because of the prominent A present), however its sequences are more G rich.

Figures 4.19 and 4.20 show that the model seem to be detecting A and T rich regions, which could be poly-A regions, and since T is the complement of A, in reverse complement sequences a T detector is also detecting a poly-A regions. The filter in Figure 4.19 seems to be more specialized in the poly-A while Figure 4.20 seem to have equally leaned both basepairs.

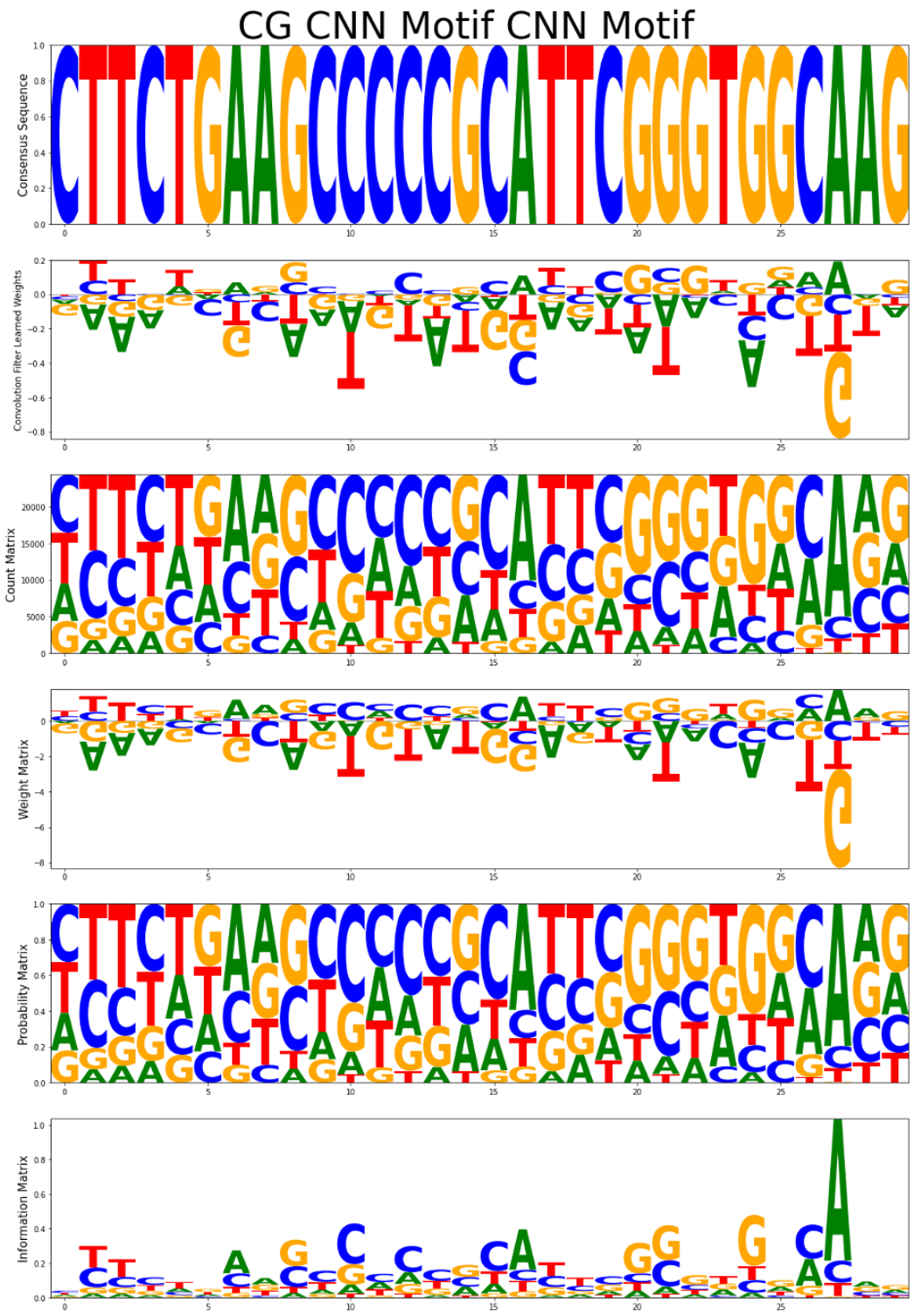

Figure 4.17 C and G detector learned by the model over the AA1 individual

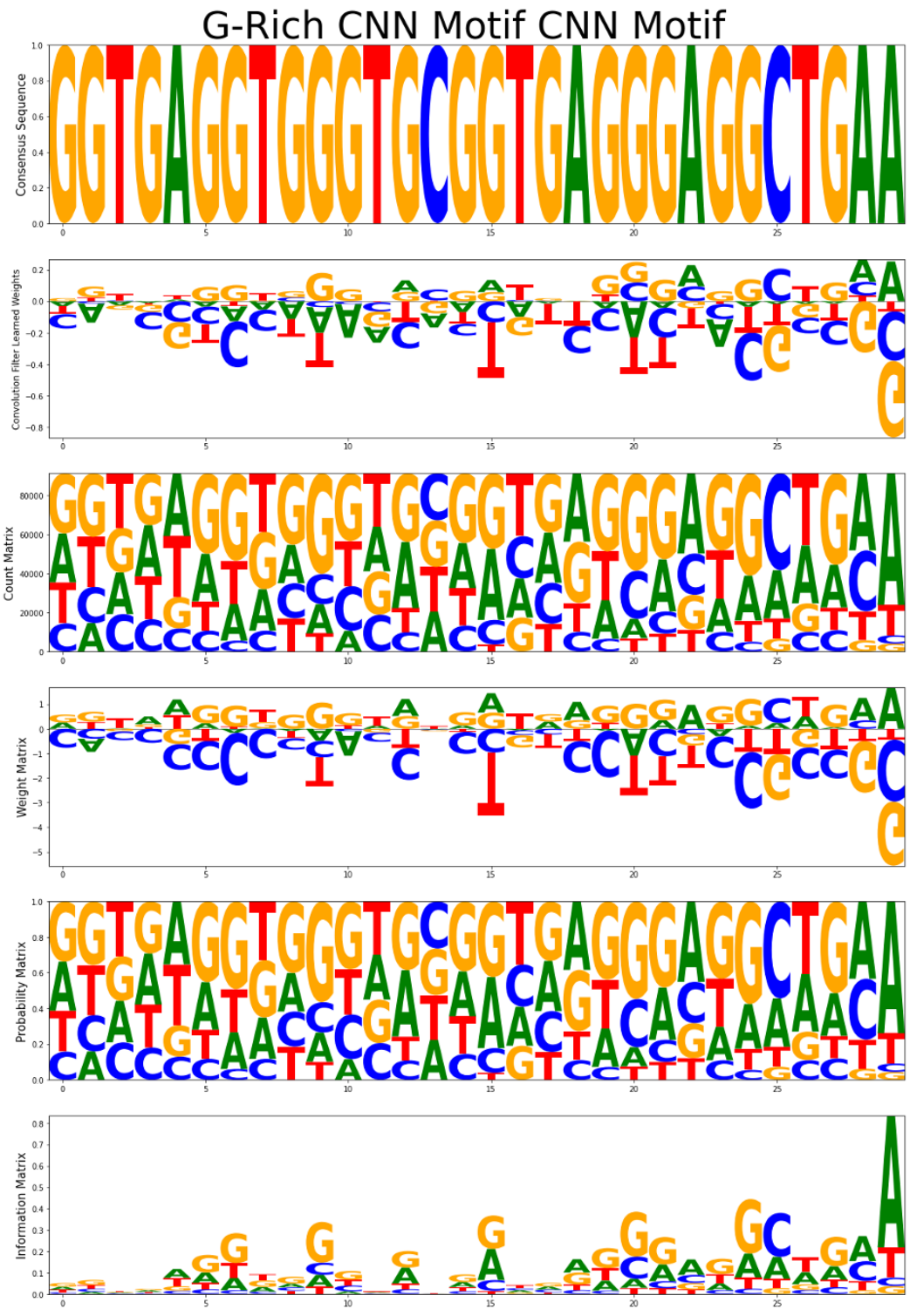

Figure 4.18 Another C and G detector learned by the model over the AA1 individual

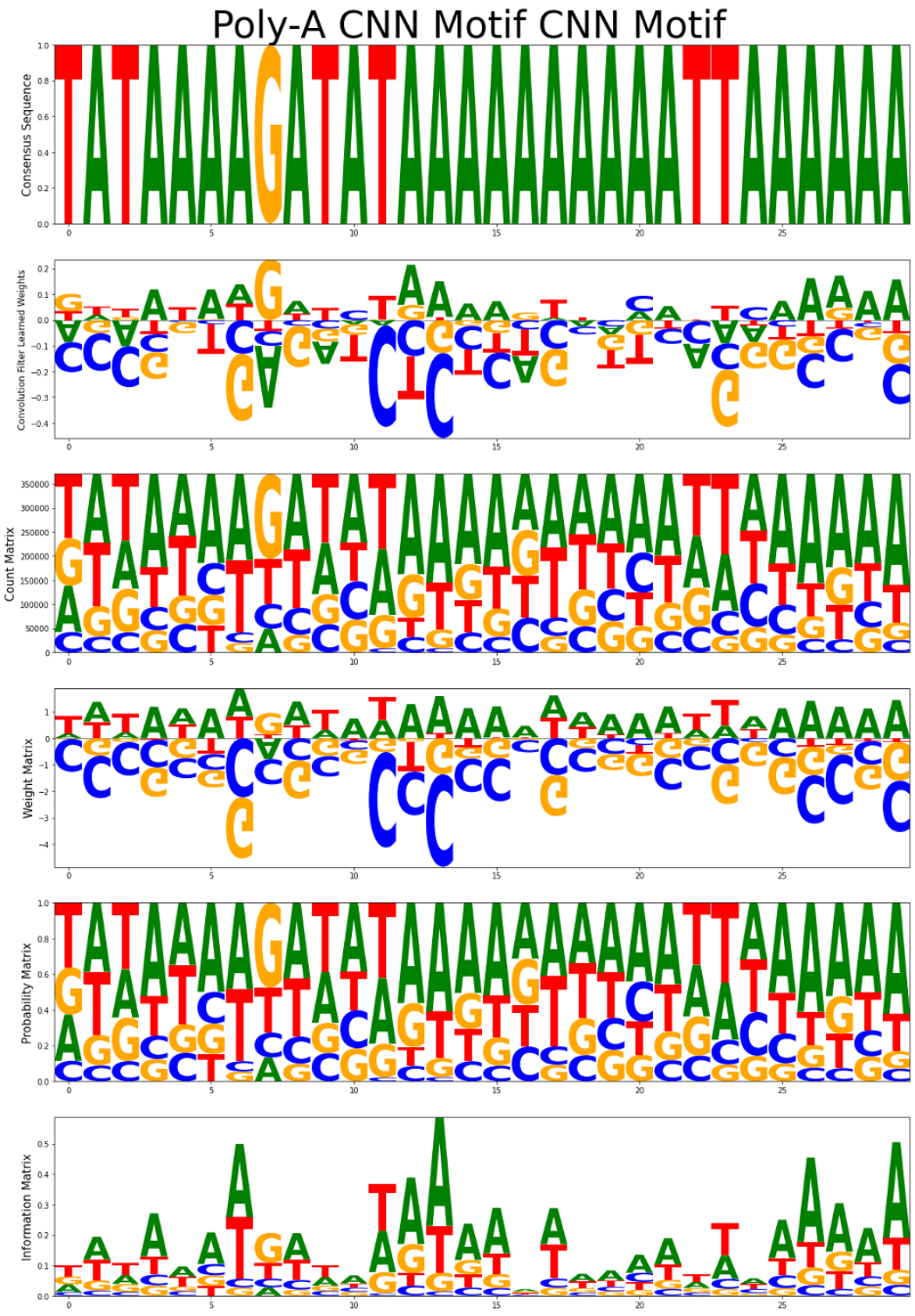

Figure 4.19 Number of hotspots per individual. Note the low count for individual AB2.

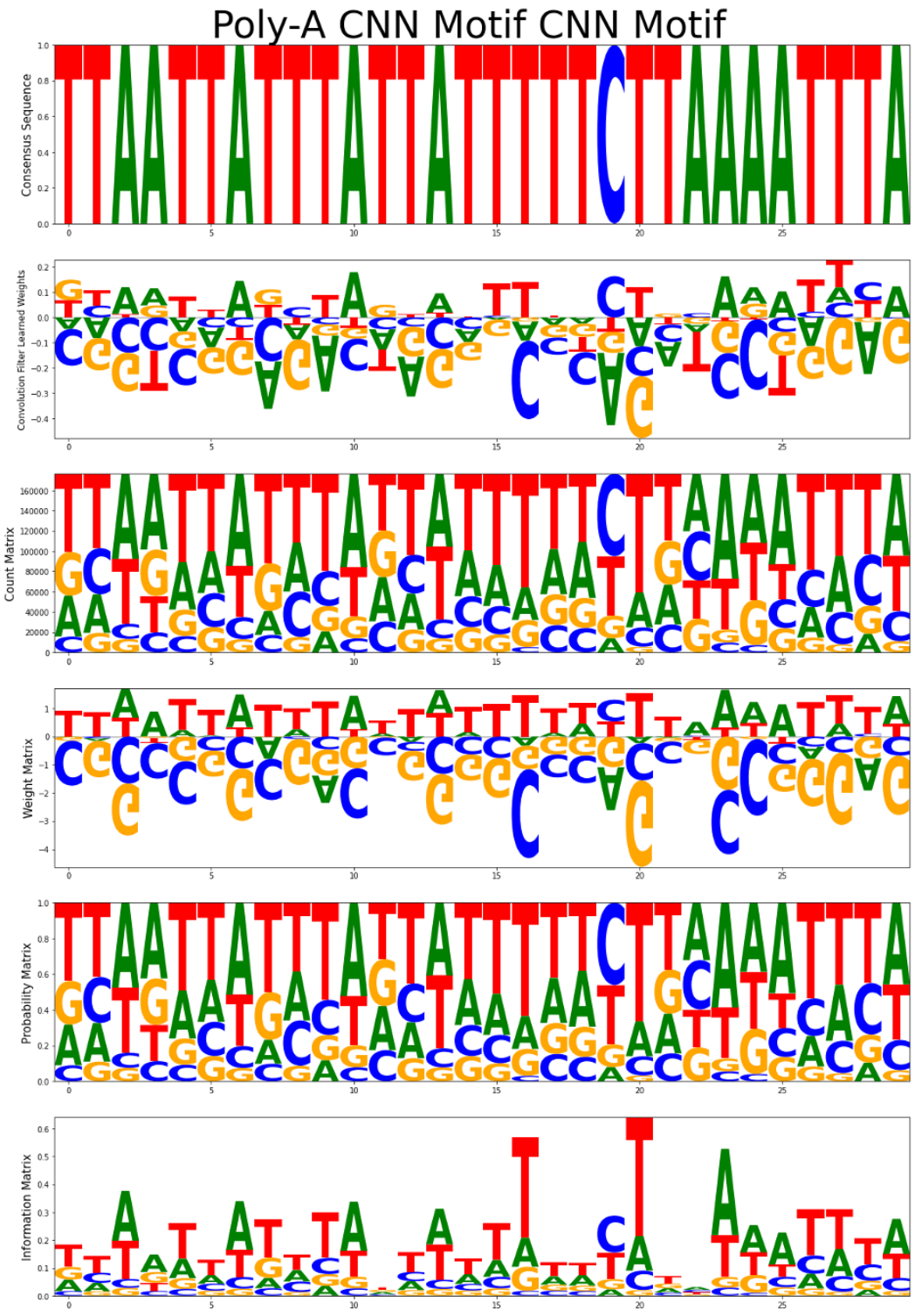

Figure 4.20 Number of hotspots per individual. Note the low count for individual AB2.

4.4 Model Predictions Explorer

Performance metrics and DNA motifs provide a lot of insights about the model's performance; however, they do not give a full picture about how the model performed across different parts of the genome. We wanted to be able to visualize such a relationship, which can help generate new hypothesis, like for example if there are certain regions on a specific chromosome that behaved in a peculiar manner. For this reason, we built a dashboard to plot different model predictions overlayed over all chromosomes.

Figure 4.21 contains a screenshot of the models' predictions exploration tool. The importance of such a tool is that it allows for visual encoding of the model's performance which can lead to further insights on top of knowing the performance metrics. The tool splits the results by chromosome and provides access to the global results across the whole dataset.

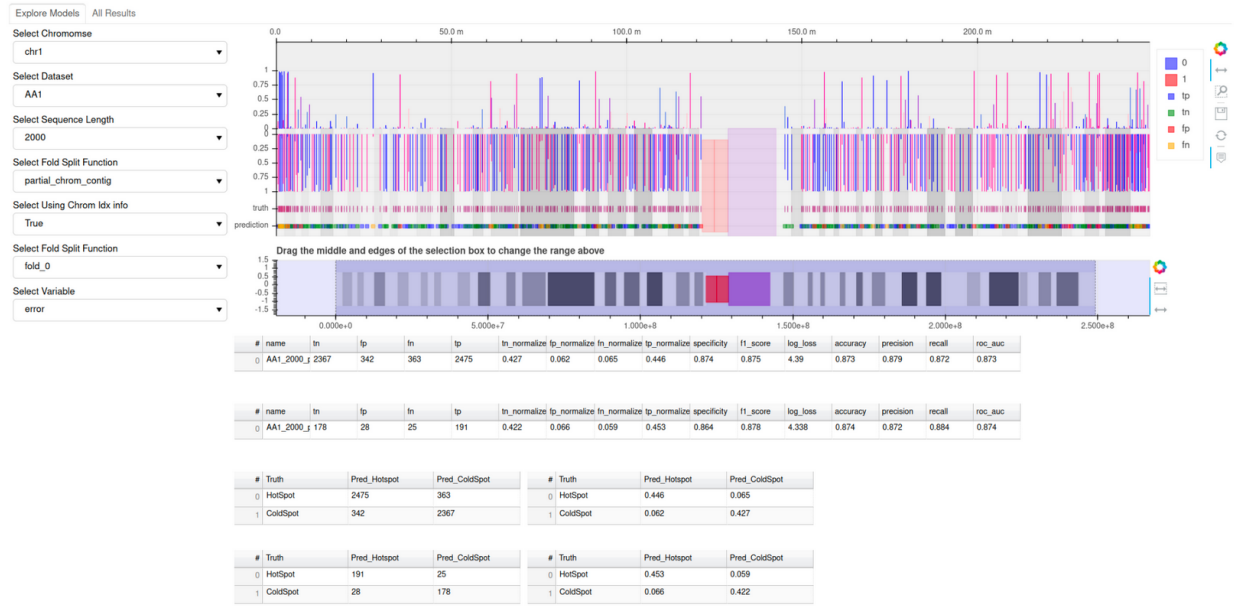

Figure 4.21 Screenshot of the model's predictions exploration tool.

The user interface (UI) is split into 5 subsections, annotated in Figure 4.22. Window 1 contains 7 dropdown menus to select which model/experiment to show. These dropdowns get the models by the chromosome number, dataset, input sequence length, folds splitting

function, if the model was provided the input sequences' genomic location and which of the 3 folds to display. Window 2 contains the main results of the examples at that specific chromosome, and these examples are displayed on the figure at their genomic coordinate. The x-axis of the result represents this genomic location.

Window 2 is split to 5 subsections annotated in Figure 4.23. Part A contains the x-axis values, which is the genomic location. Parts B and C are reciprocal to each other. Part B contains the error bar plot of the prediction at that location, while part C contains the (1 - error) value, which could be interpreted as the correctness of the prediction. The reason why we included both is due to two reasons. First, in B, extremely low values of an example cannot be distinguished from regions that do not contain examples at all. Second, since there are regions that have a high density of the examples relative to the plotting area (like the telomeres), it becomes difficult to spot good versus bad predictions. The bar plots in both sections are color encoded according to the true labels, where blue means it is a cold spot and red means a hotspot. Section D re-encodes the true labels, but in this section, we can clearly see the distribution of the examples along the chromosome, which is complementary to the bar plots. Section E contains a color encoding of the correctness of the model's prediction. The correctness is assessed according to the 4 categories of a confusion matrix, which are true positives (blue), true negatives (green), false positives (red) and false negatives (orange). The whole window has the chromosome cytogenetic bands plotted in the background with a high transparency to allow a rough estimation of the location of the example.

Back to Figure 4.22, pane 3 contains a navigation pane that allows zooming on certain locations in pane 2, which helps inspecting regions with high examples density. Pane 4 contains the numerical performance summary of the model on both across the whole dataset (top table) and on this specific chromosome (second table). Third and fourth tables isolate the confusion matrices across the whole dataset and this specific chromosome. Both confusion matrices are displayed twice, once with absolute numbers and another time normalized between 0 and 1. Finally, pane 5 contains an interactive legend, where clicking on a category toggles the visibility of its corresponding items in window 2. For example, in

Figure 4.23, the true positives and true negatives were turned off in pane E, showing only misclassified examples.

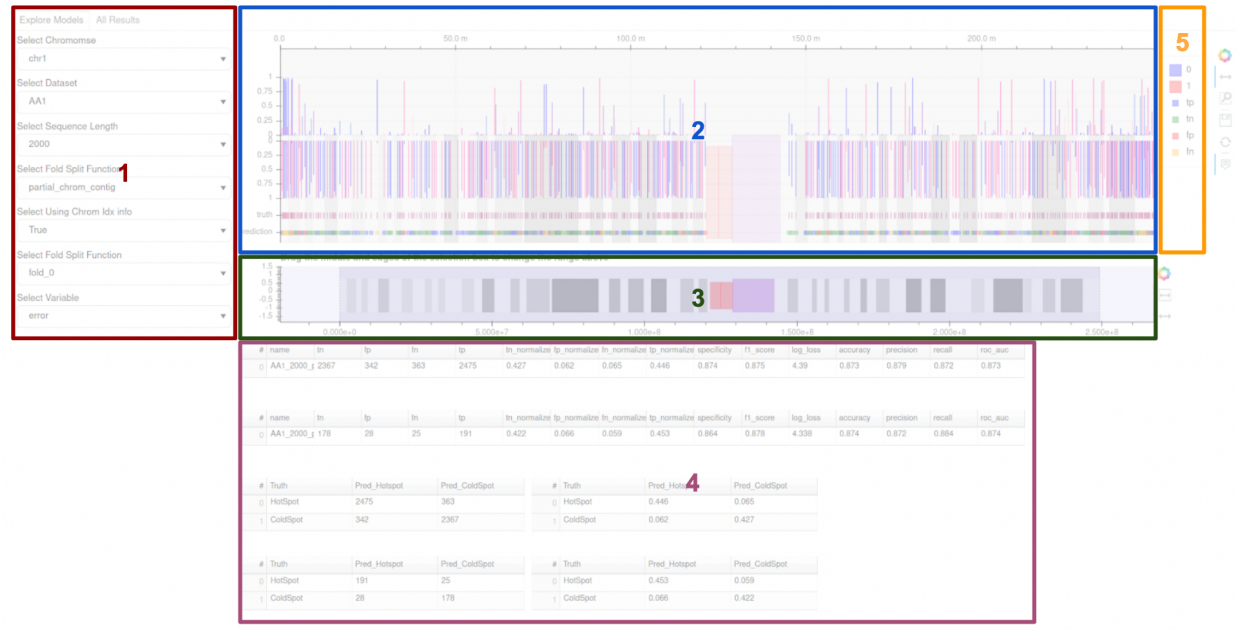

Figure 4.22 The UI is split into 5 regions. Region 1 contains the controls to select which model to display, region 2 display the results, region 3 allows zooming to specific regions on the chromosome, region 4 contains the performance metrics across the whole dataset and on that specific chromosome and region 5 contains an interactive legend to turn on and off the display of certain examples

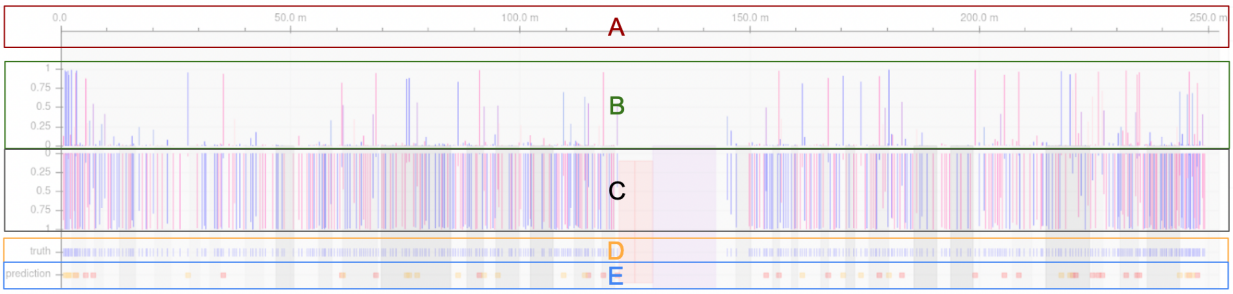

Figure 4.23 Details of pane 2. Region A contains the x-axis ticks, which are the genomic location. Regions B and C contain the bar plots *of the error and correctness of the prediction respectively, and these* bar plots *are color encoded according to the true label of the example. Region D contains the ground truth of the example. Region E color encodes the confusion matrix category of the example.*

The tool allowed to understand better the effect of the sequence length on predictions. It showed that the sequence lengths specifically helped with reducing the errors in specific regions, and it was especially helpful with reducing false negatives. In Figure 4.24, we overlayed the results of the same model across all 3 sequence lengths. In the region defined by the red box, we can see that as the sequence length increases, the number of errors in that region decreases, with practically no errors performed in this genomic region with an input sequence length of 3500 bps.

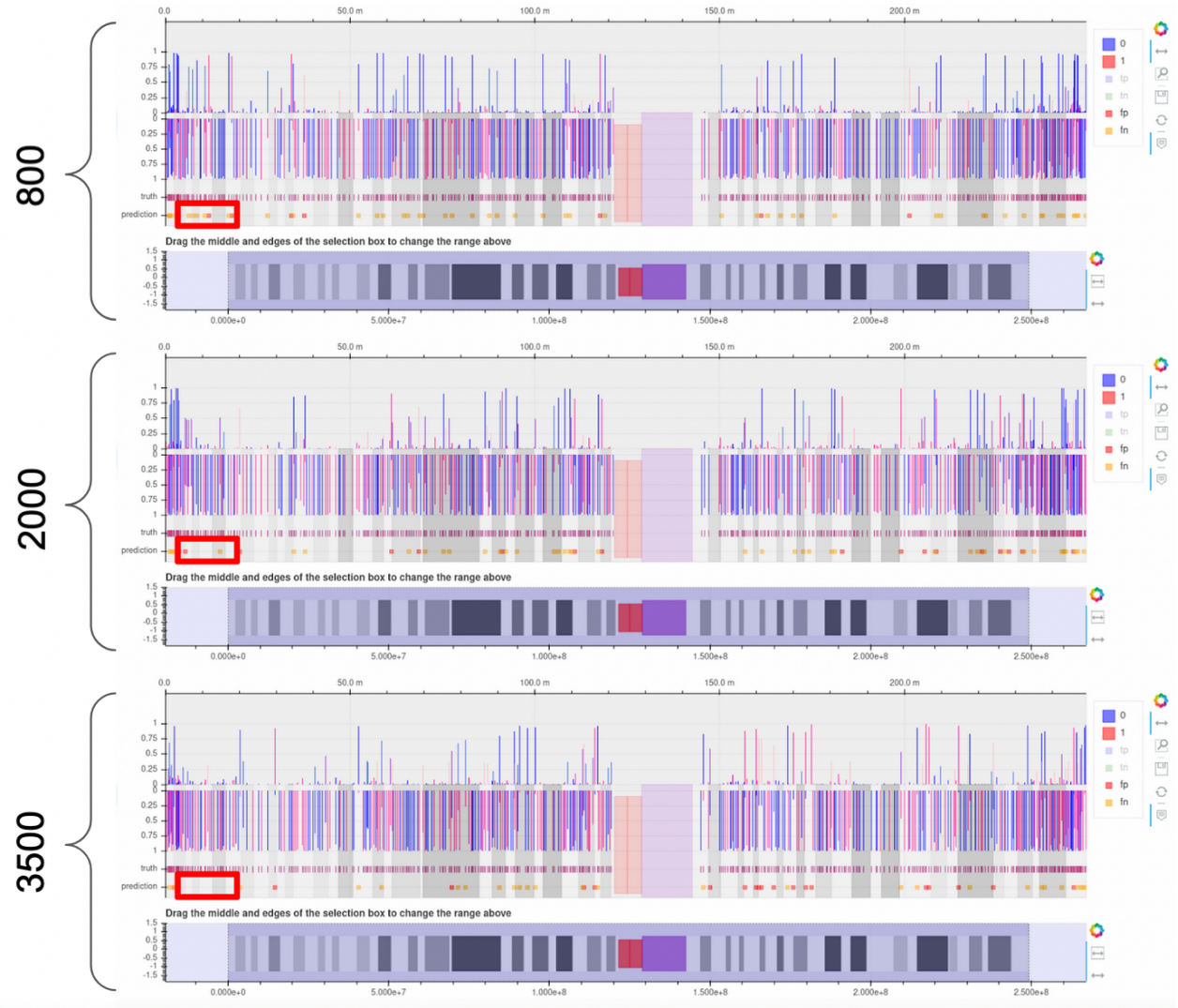

Figure 4.24 Preview of chromosome 1 on AA1 dataset across all used sequence lengths. We can notice inside the bold red box that as the sequence length increases, errors inside this region decrease and these errors disappear at sequence length 3500.

Chapter 5 Discussion, Conclusion and Future Work

In this project, we explored deep learning's capacity to model meiotic recombination hotspots from DMC1 ChIP-seq data. The results showed that this approach is suitable for the task. The model has learned to contrast DNA sequence classes and learned motifs that are known to be important for the DSB creation. We have explored new ways to construct our models and the effect of different experiment parameters we discuss in the next sections.

5.1 Input Sequence

The first important finding is that the input sequence lengths found in the literature are not the best ones for the task, and that it is better to base the input sequence length based on what we already know through biology. Despite that most of the examples were identifiable through showing only the middle 800 bps of the sequence, a significant increase in performance was obtained by using longer sequences. The shorter sequences still had their advantages though, as they had a higher precision and less computational requirements. Higher precision can indeed have its important use-cases, but if false negatives are less of an issue, then this advantage can be ignored. For the other computational benefit of using short sequences, with the ever-increasing computational performance of GPUs means that in the near future such a concern will never be a problem. Such an increase will allow also to test additional architectures. For example, in our early experiments, we have tried using the original transformer architecture (Vaswani et al., 2017). At the 3500 bps sequence, we could not use over two attention heads, and even then we had to reduce the batch size to two examples in order to avoid out of memory errors. The performance was not good, with an accuracy less than 60%, which is very close to a random baseline in balanced datasets (50% accuracy).

The second finding is that providing the model with the sequence position can improve the model's performance. However, it is important not to extrapolate what the model has learned, i.e., to never try to predict hotspots in chromosomal regions the model has never seen before. We also noticed that models trained solely on extremities of chromosomes had

better recall, and this needs more investigation to better understand the distinct predictive features near the centromeres.

5.2 Dataset

Our use-case was based over just 5 individuals, which is a very small sample size. Of course, the extremely high resolution at the bp level, and the usage of the reference genome make up for these and we can draw conclusions about the general population as proved by learning the correct PRDM9 motifs. However, integrating more knowledge will boost the findings much more. One example is integrating the genetic maps based on LD patterns. As discussed in the introduction, pedigree methods represent contemporary active hotspots, and therefore integrating them with the current reference human genome should still lead to good results.

The usage of the unprocessed ChIP-seq data should be explored. Avsec et al. (Avsec et al., 2020) have used these files and reported high performance with high sensitivity to individual single nucleotide polymorphism (SNP). However, these raw reads require more computational power than using the .bed files and will probably require more scrutiny to the trained model as these files contain unfiltered/noisy signals.

Integrating other data sources and implementing multitasking can be useful as well. For example, building a model that jointly learns to predict meiotic recombination hotspots and H3K4me3 methylation may prove interesting, as we know they are biologically related in meiosis. However, in our initial exploratory experiments, we found that extreme multitasking that was reported in the literature is harder to train and interpret. For example, the outputs become highly imbalanced (much more negative examples, with ratios over 9 to 1), and despite that we used custom loss functions to compensate for that, we could not match the performance of models learning each task alone across all tasks.

Another limitation is that, since the dataset was produced over a previous version of the reference genome, we do not make use of the full knowledge we have today. Remapping the experiment to the most recent version of the reference genome can provide better results, however such an operation needs a lot of care in choosing the pipeline parameters.

5.3 Input Features

A significant amount of design can be put into the input features. We used a one-hot-encoding for the DNA sequences and we filtered-out sequences that contained any ambiguous basepairs (N). This leads to losing some examples. Despite that for the overall task, we still had plenty of examples for the model to learn on, this strategy leads to the loss of some examples that may be interesting to study. A future work will be to study the impact of other approaches, such as one-hot-encode the N as a new category or encode such locations with different schemes such as setting all positions as 1, 0 or 0.25. There is other possible sequence encoding methods, for example byte-pair encoding (Gage, 1994) was already used for DNA sequences (Zaheer et al., 2021) , however we think we lose the advantage of using the convolution filter as a PWM and therefore we opted not to use it.

5.4 Motif Search and Interpretability

A lesson learned is that training new model changes the position of the learnt motif. This changes every time, which makes inspecting the filters laborious, and an automated search for motifs in known databases like JASPAR will further guide future steps.

We found the multiple-sequence-alignment of positive activation regions to give excellent results, especially with less well-defined regions that can have a variable length, such as poly-A regions. Out of the different representations that we tried; we preferred the information content matrix to be the most interpretable one. Directly inspecting the convolution weights was useful in cases where there is a pre-defined clear motif, and that the learnt weight bears a large resemblance to the PWM weight matrix calculation. We also visually noticed a relationship between the negative values of the learnt weights and the magnitude of the PWM information, and that may be a starting point for further investigation.

The major limitation of such a method is when there is no prior knowledge on a motif to be found. Our work acts as a positive control: the deep learning model has learnt the Myers motif. But, if for example, the biological problem at hand is affected by a motif that was partially learned by the model, inspecting the convolution filter will not yield an immediately obvious result. One suggestion could be to train the model several times over different

training and validation splits, and at each time to save the learnt motifs. Then, we can inspect the recurrent learnt features across different training rounds.

5.5 Deep Learning Limitations

Despite the unprecedented modeling powers that deep learning provided to the scientific communities, it is a far from perfect tool and its results should be treated with care. One reason is that the way we train the models only minimizes a loss function. However, this does not necessarily mean to learn the underlying causal process. We can see this as the drawback of Breiman's culture 2 (Breiman, 2001) approach, that in many cases the models will find "hacks" to minimize the loss function that are not in line with the biology. An elegant demonstration for this was performed by Geirhos et al. (Geirhos et al., 2019). In their work, they performed a style transfer (Gatys et al., 2015) of an elephant's skin to a cat image (Figure 5.1). Although that the resulting image is still easily recognizable by a human to contain a cat, the network classified the image to be that of an elephant, showing that the prediction is largely driven by the skin texture rather than elephant morphology. Such pitfalls should be addressed with great care when working with deep learning in genomics as well, especially when inferring de novo insights.

Figure 5.1 Gerihos et al. performed a style transfer of an image containing an elephant's skin (image A) to an image containing a cat (image B). The result is image C. Below each image, there are the top predictions of the neural network to this image

Another limitation of deep learning is the direct interpretation of the output as probability. For example, a model's output of 0.8 cannot be directly used as an accurate quantification of the model's uncertainty. Bayesian neural networks can provide a more accurate estimate for such uncertainty, and one way to compute these is using MC Dropouts (Gal & Ghahramani, 2015). In this approach, the network's dropouts are kept active at prediction time, knocking out a certain amount of the information flow inside the network. This leads to a certain variability in the prediction, and if we make a large enough number of predictions, we can use their average as an approximation to Bayesian network. When we applied this approach, we got some interesting but mixed results. In certain cases, the model's performance improved and in some other instances it metrics deteriorated. In Figure 5.2, we applied this approach where we turned off the dropouts and predicted the test2 dataset, then we turned them on and predicted the same dataset 100 times and plotted a histogram of the average of predictions. We noticed that in many instances, the model's certainty decreased for correct (this can be noticed by the flattening of the modes on the MC dropout predictions in the right column), and that also the incorrect predictions shifted towards the middle. However, for different hyperparameters, this was not the case, and therefore further investigation must be performed before we can report the final findings. This method should also be contrasted to directly training a full Bayesian neural network, which can be performed using the TensorFlow-probability library. This library uses variational inference to fit such a network.

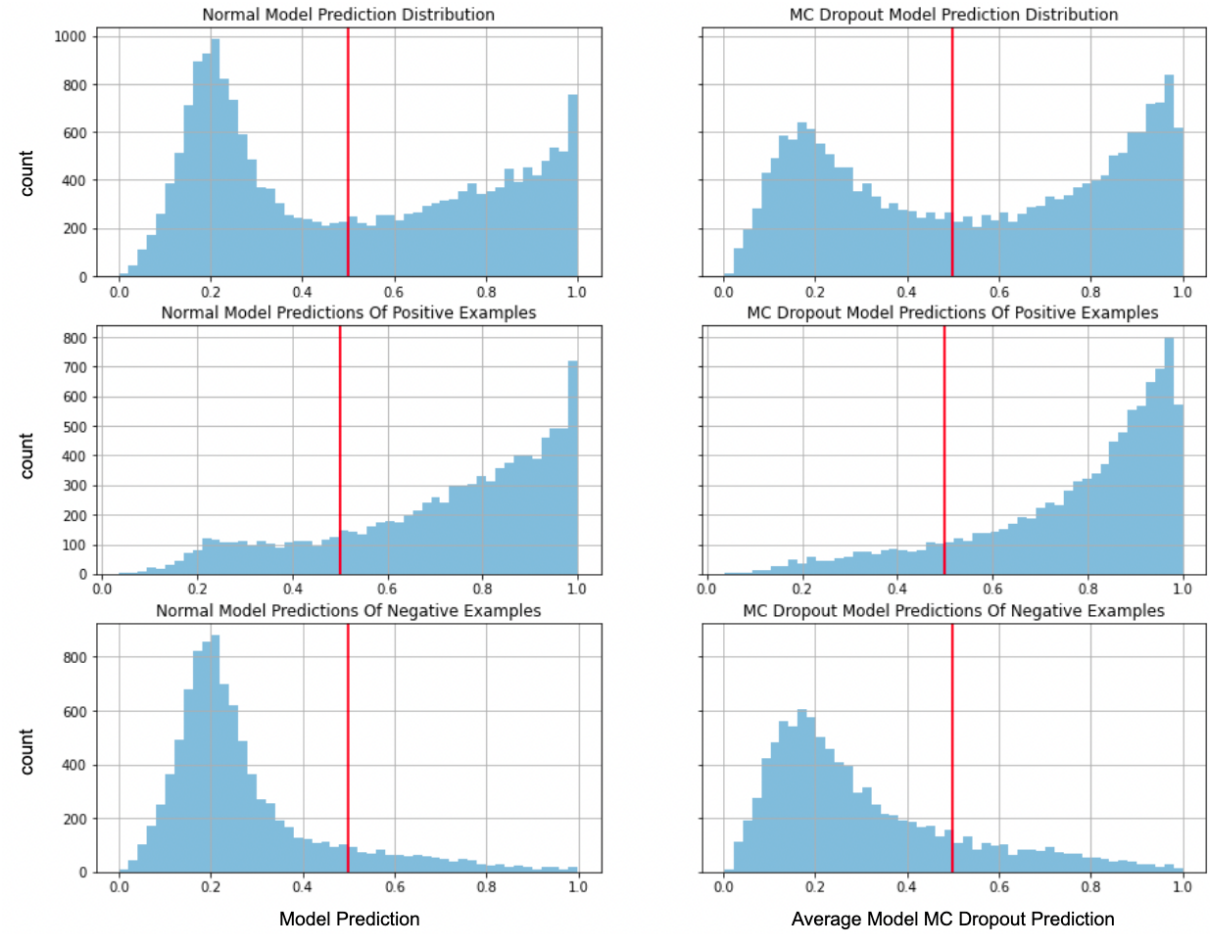

Figure 5.2 Early experiments of using MC Dropouts for estimating model uncertainty. The left column contains a histogram of the model's predictions while the right contains the average prediction of 100 predictions with dropouts left active (MC Dropout). The first row contains all predictions, the second row contains the prediction of the positive meiotic recombination hotspots examples and bottom row contains the predictions of the negative examples.

5.6 Software

We worked on exploring different best practices for developing deep learning models. The most important best practices are to separate the configuration files from the code, and to keep track of both separately in order to reproduce the results. The second important best practice is that, in case we plan on using early stopping, we need to save a fourth set, the one referred to as test2 in this project. This set provides a better measure for model performance. The result exploration tool is a very promising seed for a larger project where we can add

other covariates to the plots, such as the genomic annotations coming from the CHESS database (Pertea et al., 2018) . The UI design is by far the part that needs a lot of work, since the amount of data to be presented is not trivial. But such a tool can play an important role in hypothesis generation and refinement.

5.7 Future Work

Further investigations about the examples that gained performance by increasing the input sequence length would be interesting. We can do such investigations using gradient feature importance, such as the integrated gradients (Sundararajan et al., 2017). Another planned approach is to gradually remove the known important features such as mutating or shuffling the PRDM9 Myers motif across the examples and inspect the change in the model's prediction.

A next step to explore is the effect of sampling the negative examples of the dataset. We used only negative examples that are of close proximity to the hotspot (between 1000 and 5000 bps). So, investigating the effect of using further, far-out examples would have an effect over the model's performance would be an interesting question to explore.

Since showing the model the sequences' genomic location enhanced the predictive power, providing better positional information may be beneficial. For example, chromosomes 13, 14, 15, 21, 22 and Y are acrocentric, and therefore the q-arm in these chromosomes has little hotspots. One way to embed such an information is to show the model the distance of the sequence from the centromere instead of the normalized position, which has different properties across different chromosomes.

Software-wise, the current dataset is saved as an HDF5 file. Such a format is not thread-safe and therefore is not suitable for very large datasets. Restructuring the output dataset to use a thread-safe type such as parquet would make the dataset suitable for encoding much larger datasets. Also, incorporating a dataset version tracking tool such as DVC (https://dvc.org/) can be very useful for managing a large number of different but related datasets.

Finally, exploring further the best way to use the transformer architecture should be attempted. As of today, there are more efficient implementations such as the linformer (S. Wang et al., 2020), which can help with the computational cost of computing attention.

5.8 Conclusion

In this thesis, we introduced the biology of meiotic recombination and its importance for successive reproduction for different organisms. We also introduced deep learning's success on working with DNA sequences, and we went through our work on applying it to predict meiotic recombination hotspots. The results are encouraging as they showed that deep learning achieved high performance metrics and the problem, and that also the network learned biologically relevant features. Ultimately, we believe that the approach can be pushed far enough to uncover new insights, with the only caveat of respecting the development best practices and scrutinizing the model's predictions.

References

Abu-Mostafa, Y. S. (1990). Learning from hints in neural networks. *Journal of Complexity*, *6*(2), 192–198. https://doi.org/10.1016/0885-064X(90)90006-Y

Alipanahi, B., Delong, A., Weirauch, M. T., & Frey, B. J. (2015). Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning. *Nature Biotechnology*, *33*(8), 831–838. https://doi.org/10.1038/nbt.3300

Anderson, J. D., & Widom, J. (2001). Poly(dA-dT) Promoter Elements Increase the Equilibrium Accessibility of Nucleosomal DNA Target Sites. *Molecular and Cellular Biology*, *21*(11), 3830–3839. https://doi.org/10.1128/MCB.21.11.3830-3839.2001

Angermueller, C., Lee, H. J., Reik, W., & Stegle, O. (2017). DeepCpG: Accurate prediction of single-cell DNA methylation states using deep learning. *Genome Biology*, *18*(1), 67. https://doi.org/10.1186/s13059-017-1189-z

Arnheim, N., Calabrese, P., & Nordborg, M. (2003). Hot and Cold Spots of Recombination in the Human Genome: The Reason We Should Find Them and How This Can Be Achieved. *American Journal of Human Genetics*, *73*(1), 5–16.

Asgari, E., & Mofrad, M. R. K. (2015). Continuous Distributed Representation of Biological Sequences for Deep Proteomics and Genomics. *PLOS ONE*, *10*(11), e0141287. https://doi.org/10.1371/journal.pone.0141287

Avsec, Ž., Weilert, M., Shrikumar, A., Krueger, S., Alexandari, A., Dalal, K., Fropf, R., McAnany, C., Gagneur, J., Kundaje, A., & Zeitlinger, J. (2020). *Base-resolution models of transcription factor binding reveal soft motif syntax* (p. 737981). https://doi.org/10.1101/737981

Bagshaw, A. T., Pitt, J. P., & Gemmell, N. J. (2006). Association of poly-purine/poly-pyrimidine sequences with meiotic recombination hot spots. *BMC Genomics*, *7*, 179. https://doi.org/10.1186/1471-2164-7-179

Bahdanau, D., Cho, K., & Bengio, Y. (2016). Neural Machine Translation by Jointly Learning to Align and Translate. *ArXiv:1409.0473 [Cs, Stat]*. http://arxiv.org/abs/1409.0473

Baker, C. L., Petkova, P., Walker, M., Flachs, P., Mihola, O., Trachtulec, Z., Petkov, P. M., & Paigen, K. (2015). Multimer Formation Explains Allelic Suppression of PRDM9 Recombination Hotspots. *PLOS Genetics*, *11*(9), e1005512. https://doi.org/10.1371/journal.pgen.1005512

Baker, C. L., Walker, M., Kajita, S., Petkov, P. M., & Paigen, K. (2014). PRDM9 binding organizes hotspot nucleosomes and limits Holliday junction migration. *Genome Research*, *24*(5), 724–732. https://doi.org/10.1101/gr.170167.113

Base Pair. (n.d.). Genome.Gov. Retrieved December 16, 2021, from https://www.genome.gov/genetics-glossary/Base-Pair

Baudat, F., Buard, J., Grey, C., Fledel-Alon, A., Ober, C., Przeworski, M., Coop, G., & de Massy, B. (2010). PRDM9 is a Major Determinant of Meiotic Recombination Hotspots in humans and mice. *Science (New York, N.Y.)*, *327*(5967), 836–840. https://doi.org/10.1126/science.1183439

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(8), 1798–1828. https://doi.org/10.1109/TPAMI.2013.50

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, *5*(2), 157–166. https://doi.org/10.1109/72.279181

Berg, I. L., Neumann, R., Lam, K.-W. G., Sarbajna, S., Odenthal-Hesse, L., May, C. A., & Jeffreys, A. J. (2010). PRDM9 variation strongly influences recombination hot-spot activity and meiotic instability in humans. *Nature Genetics*, *42*(10), 859–863. https://doi.org/10.1038/ng.658

Beye, M., Gattermeier, I., Hasselmann, M., Gempe, T., Schioett, M., Baines, J. F., Schlipalius, D., Mougel, F., Emore, C., Rueppell, O., Sirviö, A., Guzmán-Novoa, E., Hunt, G., Solignac, M., & Page, R. E. (2006). Exceptionally high levels of recombination across the honey bee genome. *Genome Research*, *16*(11), 1339–1344. https://doi.org/10.1101/gr.5680406

Billings, T., Parvanov, E. D., Baker, C. L., Walker, M., Paigen, K., & Petkov, P. M. (2013). DNA binding specificities of the long zinc-finger recombination protein PRDM9. *Genome Biology*, *14*(4), R35. https://doi.org/10.1186/gb-2013-14-4-r35

Bird, A. (2002). DNA methylation patterns and epigenetic memory. *Genes & Development*, *16*(1), 6–21. https://doi.org/10.1101/gad.947102

Blum, C. F., & Kollmann, M. (2019). Neural networks with circular filters enable data efficient inference of sequence motifs. *Bioinformatics*, *35*(20), 3937–3943. https://doi.org/10.1093/bioinformatics/btz194

Boyle, A. P., Davis, S., Shulha, H. P., Meltzer, P., Margulies, E. H., Weng, Z., Furey, T. S., & Crawford, G. E. (2008). High-Resolution Mapping and Characterization of Open Chromatin across the Genome. *Cell*, *132*(2), 311–322. https://doi.org/10.1016/j.cell.2007.12.014

Breiman, L. (2001). Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statistical Science*, *16*(3), 199–231.

Brick, K., Smagulova, F., Khil, P., Camerini-Otero, R. D., & Petukhova, G. V. (2012). Genetic recombination is directed away from functional genomic elements in mice. *Nature*, *485*(7400), 642–645. https://doi.org/10.1038/nature11089

Brown, R. C., & Lunter, G. (2019). An equivariant Bayesian convolutional network predicts recombination hotspots and accurately resolves binding motifs. *Bioinformatics*, *35*(13), 2177–2184. https://doi.org/10.1093/bioinformatics/bty964

Buenrostro, J. D., Giresi, P. G., Zaba, L. C., Chang, H. Y., & Greenleaf, W. J. (2013). Transposition of native chromatin for fast and sensitive epigenomic profiling of open chromatin, DNA-binding proteins and nucleosome position. *Nature Methods*, *10*(12), 1213–1218. https://doi.org/10.1038/nmeth.2688

Caruana, R. (n.d.). *Multitask Learning*. 35.

Cell Division: Stages of Mitosis | Learn Science at Scitable. (n.d.). Retrieved December 25, 2021, from http://www.nature.com/scitable/topicpage/mitosis-and-cell-division-205

Chen, C., Hou, J., Shi, X., Yang, H., Birchler, J. A., & Cheng, J. (2019). *Interpretable attention model in transcription factor binding site prediction with deep neural networks* [Preprint]. Bioinformatics. https://doi.org/10.1101/648691

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1724–1734. https://doi.org/10.3115/v1/D14-1179

Cohn, D., Zuk, O., & Kaplan, T. (2018). *Enhancer Identification using Transfer and Adversarial Deep Learning of DNA Sequences* [Preprint]. Bioinformatics. https://doi.org/10.1101/264200

Crosland, M. W., & Crozier, R. H. (1986). Myrmecia pilosula, an Ant with Only One Pair of Chromosomes. *Science (New York, N.Y.)*, *231*(4743), 1278. https://doi.org/10.1126/science.231.4743.1278

Davies, B., Hatton, E., Altemose, N., Hussin, J. G., Pratto, F., Zhang, G., Hinch, A. G., Moralli, D., Biggs, D., Diaz, R., Preece, C., Li, R., Bitoun, E., Brick, K., Green, C. M., Camerini-Otero, R. D., Myers, S. R., & Donnelly, P. (2016). Re-engineering the zinc fingers of PRDM9 reverses hybrid sterility in mice. *Nature*, *530*(7589), 171–176. https://doi.org/10.1038/nature16931

De Bont, R., & van Larebeke, N. (2004). Endogenous DNA damage in humans: A review of quantitative data. *Mutagenesis*, *19*(3), 169–185. https://doi.org/10.1093/mutage/geh025

Diagouraga, B., Clément, J. A. J., Duret, L., Kadlec, J., de Massy, B., & Baudat, F. (2018). PRDM9 Methyltransferase Activity Is Essential for Meiotic DNA Double-Strand Break Formation at Its Binding Sites. *Molecular Cell*, *69*(5), 853-865.e6. https://doi.org/10.1016/j.molcel.2018.01.033

Douki, T., Reynaud-Angelin, A., Cadet, J., & Sage, E. (2003). Bipyrimidine photoproducts rather than oxidative lesions are the main type of DNA damage involved in the genotoxic effect of solar UVA radiation. *Biochemistry*, *42*(30), 9221–9226. https://doi.org/10.1021/bi034593c

Douven, I. (2021). Abduction. In E. N. Zalta (Ed.), *The Stanford Encyclopedia of Philosophy* (Summer 2021). Metaphysics Research Lab, Stanford University. https://plato.stanford.edu/archives/sum2021/entries/abduction/

Forejt, J., & Iványi, P. (1974). Genetic studies on male sterility of hybrids between laboratory and wild mice (Mus musculus L.). *Genetics Research*, *24*(2), 189–206. https://doi.org/10.1017/S0016672300015214

Fullwood, M. J., & Ruan, Y. (2009). ChIP-based methods for the identification of long-range chromatin interactions. *Journal of Cellular Biochemistry*, *107*(1), 30–39. https://doi.org/10.1002/jcb.22116

Gage, P. (1994). A new algorithm for data compression. *The C Users Journal*, *12*(2), 23–38.

Gal, Y., & Ghahramani, Z. (2015). *Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning*. https://arxiv.org/abs/1506.02142v6

Gartner Says Nearly Half of CIOs Are Planning to Deploy Artificial Intelligence. (n.d.). Gartner. Retrieved January 3, 2022, from https://www.gartner.com/en/newsroom/press-releases/2018-02-13-gartner-says-nearly-half-of-cios-are-planning-to-deploy-artificial-intelligence

Gatys, L. A., Ecker, A. S., & Bethge, M. (2015). A Neural Algorithm of Artistic Style. *ArXiv:1508.06576 [Cs, q-Bio]*. http://arxiv.org/abs/1508.06576

Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F. A., & Brendel, W. (2019). ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. *ArXiv:1811.12231 [Cs, q-Bio, Stat]*. http://arxiv.org/abs/1811.12231

Goldfarb, T., & Lichten, M. (2010). Frequent and Efficient Use of the Sister Chromatid for DNA Double-Strand Break Repair during Budding Yeast Meiosis. *PLoS Biology*, *8*(10), e1000520. https://doi.org/10.1371/journal.pbio.1000520

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. The MIT Press.

Habibi Aghdam, H., & Jahani Heravi, E. (2017). Visualizing Neural Networks. In H. Habibi Aghdam & E. Jahani Heravi, *Guide to Convolutional Neural Networks* (pp. 247–258). Springer International Publishing. https://doi.org/10.1007/978-3-319-57550-6_7

Hayashi, K., Yoshida, K., & Matsui, Y. (2005). A histone H3 methyltransferase controls epigenetic events required for meiotic prophase. *Nature*, *438*(7066), 374–378. https://doi.org/10.1038/nature04112

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778. https://doi.org/10.1109/CVPR.2016.90

Heissl, A., Betancourt, A. J., Hermann, P., Povysil, G., Arbeithuber, B., Futschik, A., Ebner, T., & Tiemann-Boege, I. (2019). The impact of poly-A microsatellite heterologies in meiotic recombination. *Life Science Alliance*, *2*(2), e201900364. https://doi.org/10.26508/lsa.201900364

Hicks, W. M., Yamaguchi, M., & Haber, J. E. (2011). Real-time analysis of double-strand DNA break repair by homologous recombination. *Proceedings of the National Academy of Sciences of the United States of America*, *108*(8), 3108–3115. https://doi.org/10.1073/pnas.1019660108

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hubel, D. H., & Wiesel, T. N. (1968). Receptive fields and functional architecture of monkey striate cortex. *The Journal of Physiology*, *195*(1), 215–243. https://doi.org/10.1113/jphysiol.1968.sp008455

Janin, J., & Wodak, S. J. (1983). Structural domains in proteins and their role in the dynamics of protein function. *Progress in Biophysics and Molecular Biology*, *42*, 21–78. https://doi.org/10.1016/0079-6107(83)90003-2

JASPAR An open-access database of transcription factor binding profiles. (n.d.). Retrieved December 29, 2021, from http://jaspar.genereg.net

Johnson, D. S., Mortazavi, A., Myers, R. M., & Wold, B. (2007). Genome-Wide Mapping of in Vivo Protein-DNA Interactions. *Science*. https://doi.org/10.1126/science.1141319

Kaiser, L., Gomez, A. N., Shazeer, N., Vaswani, A., Parmar, N., Jones, L., & Uszkoreit, J. (2017). *One Model To Learn Them All*. https://arxiv.org/abs/1706.05137v1

Kang, Z.-J., Liu, Y.-F., Xu, L.-Z., Long, Z.-J., Huang, D., Yang, Y., Liu, B., Feng, J.-X., Pan, Y.-J., Yan, J.-S., & Liu, Q. (2016). The Philadelphia chromosome in leukemogenesis. *Chinese Journal of Cancer*, *35*, 48. https://doi.org/10.1186/s40880-016-0108-0

Keeney, S., Lange, J., & Mohibullah, N. (2014). Self-Organization of Meiotic Recombination Initiation: General Principles and Molecular Pathways. *Annual Review of Genetics*, *48*, 187–214. https://doi.org/10.1146/annurev-genet-120213-092304

Kelley, D. R., Snoek, J., & Rinn, J. (2015). *Basset: Learning the regulatory code of the accessible genome with deep convolutional neural networks* [Preprint]. Genomics. https://doi.org/10.1101/028399

Khan, F., Khan, M., Iqbal, N., Khan, S., Muhammad Khan, D., Khan, A., & Wei, D.-Q. (2020). Prediction of Recombination Spots Using Novel Hybrid Feature Extraction Method via Deep Learning Approach. *Frontiers in Genetics*, *11*, 539227. https://doi.org/10.3389/fgene.2020.539227

Khil, P. P., Smagulova, F., Brick, K. M., Camerini-Otero, R. D., & Petukhova, G. V. (2012). Sensitive mapping of recombination hotspots using sequencing-based detection of ssDNA. *Genome Research*, *22*(5), 957–965. https://doi.org/10.1101/gr.130583.111

Kimothi, D., Soni, A., Biyani, P., & Hogan, J. M. (2016). Distributed Representations for Biological Sequence Analysis. *ArXiv:1608.05949 [Cs, q-Bio]*. http://arxiv.org/abs/1608.05949

Klug, A. (2010). The Discovery of Zinc Fingers and Their Applications in Gene Regulation and Genome Manipulation. *Annual Review of Biochemistry*, *79*(1), 213–231. https://doi.org/10.1146/annurev-biochem-010909-095056

Klug, A., & Rhodes, D. (1987). Zinc fingers: A novel protein fold for nucleic acid recognition. *Cold Spring Harbor Symposia on Quantitative Biology*, *52*, 473–482. https://doi.org/10.1101/sqb.1987.052.01.054

Lander, E. S., Linton, L. M., Birren, B., Nusbaum, C., Zody, M. C., Baldwin, J., Devon, K., Dewar, K., Doyle, M., FitzHugh, W., Funke, R., Gage, D., Harris, K., Heaford, A., Howland, J., Kann, L., Lehoczky, J., LeVine, R., McEwan, P., … The Wellcome Trust: (2001). Initial sequencing and analysis of the human genome. *Nature*, *409*(6822), 860–921. https://doi.org/10.1038/35057062

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, *1*(4), 541–551. https://doi.org/10.1162/neco.1989.1.4.541

Lee, C.-K., Shibata, Y., Rao, B., Strahl, B. D., & Lieb, J. D. (2004). Evidence for nucleosome depletion at active regulatory regions genome-wide. *Nature Genetics*, *36*(8), 900–905. https://doi.org/10.1038/ng1400

Li, Y., Chen, S., Rapakoulia, T., Kuwahara, H., Yip, K. Y., & Gao, X. (2021). *Deep learning identifies and quantifies recombination hotspot determinants* (p. 2021.07.29.454133). bioRxiv. https://doi.org/10.1101/2021.07.29.454133

Li, Y., Wu, F.-X., & Ngom, A. (2016). A review on machine learning principles for multi-view biological data integration. *Briefings in Bioinformatics*, bbw113. https://doi.org/10.1093/bib/bbw113

Lieberman-Aiden, E., van Berkum, N. L., Williams, L., Imakaev, M., Ragoczy, T., Telling, A., Amit, I., Lajoie, B. R., Sabo, P. J., Dorschner, M. O., Sandstrom, R., Bernstein, B., Bender, M. A., Groudine, M., Gnirke, A., Stamatoyannopoulos, J., Mirny, L. A., Lander, E. S., & Dekker, J.

(2009). Comprehensive Mapping of Long-Range Interactions Reveals Folding Principles of the Human Genome. *Science*, *326*(5950), 289–293. https://doi.org/10.1126/science.1181369

Lin, J. T., & Inigo, R. (1991). Hand written zip code recognition by back propagation neural network. *IEEE Proceedings of the SOUTHEASTCON '91*, 731–735. https://doi.org/10.1109/SECON.1991.147854

Lister, R., O'Malley, R. C., Tonti-Filippini, J., Gregory, B. D., Berry, C. C., Millar, A. H., & Ecker, J. R. (2008). Highly Integrated Single-Base Resolution Maps of the Epigenome in Arabidopsis. *Cell*, *133*(3), 523–536. https://doi.org/10.1016/j.cell.2008.03.029

Liu, F., Li, H., Ren, C., Bo, X., & Shu, W. (2016). *PEDLA: Predicting enhancers with a deep learning-based algorithmic framework* [Preprint]. Bioinformatics. https://doi.org/10.1101/036129

Luger, K., Mäder, A. W., Richmond, R. K., Sargent, D. F., & Richmond, T. J. (1997). Crystal structure of the nucleosome core particle at 2.8 A resolution. *Nature*, *389*(6648), 251–260. https://doi.org/10.1038/38444

Luscombe, N. M., Laskowski, R. A., & Thornton, J. M. (2001). Amino acid–base interactions: A three-dimensional analysis of protein–DNA interactions at an atomic level. *Nucleic Acids Research*, *29*(13), 2860–2874.

Malik, S.-B., Pightling, A. W., Stefaniak, L. M., Schurko, A. M., & Logsdon, J. M. (2007). An expanded inventory of conserved meiotic genes provides evidence for sex in Trichomonas vaginalis. *PloS One*, *3*(8), e2879. https://doi.org/10.1371/journal.pone.0002879

Mihola, O., Trachtulec, Z., Vlcek, C., Schimenti, J. C., & Forejt, J. (2009). A Mouse Speciation Gene Encodes a Meiotic Histone H3 Methyltransferase. *Science*, *323*(5912), 373–375. https://doi.org/10.1126/science.1163601

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *ArXiv:1301.3781 [Cs]*. http://arxiv.org/abs/1301.3781

Mill, J. S. (2011). *A System of Logic, Ratiocinative and Inductive: Being a Connected View of the Principles of Evidence, and the Methods of Scientific Investigation* (Vol. 1). Cambridge University Press. https://doi.org/10.1017/CBO9781139149839

Min, X., Zeng, W., Chen, S., Chen, N., Chen, T., & Jiang, R. (2017). Predicting enhancers with deep convolutional neural networks. *BMC Bioinformatics*, *18*(S13), 478. https://doi.org/10.1186/s12859-017-1878-3

Mitchell, T. (1997). *Machine Learning*. McGraw Hill. http://www.cs.cmu.edu/~tom/mlbook.html

Mumbach, M. R., Rubin, A. J., Flynn, R. A., Dai, C., Khavari, P. A., Greenleaf, W. J., & Chang, H. Y. (2016). HiChIP: Efficient and sensitive analysis of protein-directed genome architecture. *Nature Methods*, *13*(11), 919–922. https://doi.org/10.1038/nmeth.3999

Myers, S., Bottolo, L., Freeman, C., McVean, G., & Donnelly, P. (2005). A fine-scale map of recombination rates and hotspots across the human genome. *Science (New York, N.Y.)*, *310*(5746), 321–324. https://doi.org/10.1126/science.1117196

Myers, S., Bowden, R., Tumian, A., Bontrop, R. E., Freeman, C., MacFie, T. S., McVean, G., & Donnelly, P. (2010). Drive Against Hotspot Motifs in Primates Implicates the PRDM9 gene in Meiotic Recombination. *Science (New York, N.Y.)*, *327*(5967), 10.1126/science.1182363. https://doi.org/10.1126/science.1182363

Myers, S., Freeman, C., Auton, A., Donnelly, P., & McVean, G. (2008). A common sequence motif associated with recombination hot spots and genome instability in humans. *Nature Genetics*, *40*(9), 1124–1129. https://doi.org/10.1038/ng.213

Narasimhan, V. M., Hunt, K. A., Mason, D., Baker, C. L., Karczewski, K. J., Barnes, M. R., Barnett, A. H., Bates, C., Bellary, S., Bockett, N. A., Giorda, K., Griffiths, C. J., Hemingway, H., Jia, Z., Kelly, M. A., Khawaja, H. A., Lek, M., McCarthy, S., McEachan, R., … van Heel, D. A. (2016). Health and population effects of rare gene knockouts in adult humans with related parents. *Science (New York, N.Y.)*, *352*(6284), 474–477. https://doi.org/10.1126/science.aac8624

NASA Astrobiology. (n.d.). Retrieved December 8, 2021, from https://astrobiology.nasa.gov/research/life-detection/about/

Nath, A., & Karthikeyan, S. (2018). Enhanced prediction of recombination hotspots using input features extracted by class specific autoencoders. *Journal of Theoretical Biology*, *444*, 73–82. https://doi.org/10.1016/j.jtbi.2018.02.016

Needleman, S. B., & Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, *48*(3), 443–453. https://doi.org/10.1016/0022-2836(70)90057-4

Ng, P. (2017). *dna2vec: Consistent vector representations of variable-length k-mers*. https://arxiv.org/abs/1701.06279v1

Otto, S. P. (2007). The Evolutionary Consequences of Polyploidy. *Cell*, *131*(3), 452–462. https://doi.org/10.1016/j.cell.2007.10.022

Paggi, J. M., & Bejerano, G. (2017). *A sequence-based, deep learning model accurately predicts RNA splicing branchpoints* [Preprint]. Bioinformatics. https://doi.org/10.1101/185868

Pan, J., Sasaki, M., Kniewel, R., Murakami, H., Blitzblau, H. G., Tischfield, S. E., Zhu, X., Neale, M. J., Jasin, M., Socci, N. D., Hochwagen, A., & Keeney, S. (2011). A hierarchical combination of factors shapes the genome-wide topography of yeast meiotic recombination initiation. *Cell*, *144*(5), 719–731. https://doi.org/10.1016/j.cell.2011.02.009

Pascanu, R., Mikolov, T., & Bengio, Y. (2013). On the difficulty of training Recurrent Neural Networks. *ArXiv:1211.5063 [Cs]*. http://arxiv.org/abs/1211.5063

Patel, A., Horton, J. R., Wilson, G. G., Zhang, X., & Cheng, X. (2016). Structural basis for human PRDM9 action at recombination hot spots. *Genes & Development*, *30*(3), 257–265. https://doi.org/10.1101/gad.274928.115

Pertea, M., Shumate, A., Pertea, G., Varabyou, A., Breitwieser, F. P., Chang, Y.-C., Madugundu, A. K., Pandey, A., & Salzberg, S. L. (2018). CHESS: A new human gene catalog curated from thousands of large-scale RNA sequencing experiments reveals extensive transcriptional noise. *Genome Biology*, *19*(1), 208. https://doi.org/10.1186/s13059-018-1590-2

Powers, N. R., Parvanov, E. D., Baker, C. L., Walker, M., Petkov, P. M., & Paigen, K. (2016). The Meiotic Recombination Activator PRDM9 Trimethylates Both H3K36 and H3K4 at Recombination Hotspots In Vivo. *PLoS Genetics*, *12*(6), e1006146. https://doi.org/10.1371/journal.pgen.1006146

Pratto, F., Brick, K., Khil, P., Smagulova, F., Petukhova, G. V., & Camerini-Otero, R. D. (2014). Recombination initiation maps of individual human genomes. *Science (New York, N.Y.)*, *346*(6211), 1256442. https://doi.org/10.1126/science.1256442

Quang, D., & Xie, X. (2015). *DanQ: A hybrid convolutional and recurrent deep neural network for quantifying the function of DNA sequences* [Preprint]. Genomics. https://doi.org/10.1101/032821

Quang, D., & Xie, X. (2017). *FactorNet: A deep learning framework for predicting cell type specific transcription factor binding from nucleotide-resolution sequential data* [Preprint]. Genomics. https://doi.org/10.1101/151274

Reisz, J. A., Bansal, N., Qian, J., Zhao, W., & Furdui, C. M. (2014). Effects of Ionizing Radiation on Biological Molecules—Mechanisms of Damage and Emerging Methods of Detection. *Antioxidants & Redox Signaling*, *21*(2), 260–292. https://doi.org/10.1089/ars.2013.5489

Robinson, J. A. (1971). Computational Logic: The Unification Computation. *Machine Intelligence*, *6*, 63–72.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536. https://doi.org/10.1038/323533a0

Sakoe, H., Isotani, R., Yoshida, K., Iso, K.-I., & Watanabe, T. (1989). Speaker-independent word recognition using dynamic programming neural networks. *International Conference on Acoustics, Speech, and Signal Processing,* 29–32 vol.1. https://doi.org/10.1109/ICASSP.1989.266355

Saxonov, S., Berg, P., & Brutlag, D. L. (2006). A genome-wide analysis of CpG dinucleotides in the human genome distinguishes two distinct classes of promoters. *Proceedings of the National Academy of Sciences of the United States of America*, *103*(5), 1412–1417. https://doi.org/10.1073/pnas.0510310103

Schneider, T. D., & Stephens, R. M. (1990). Sequence logos: A new way to display consensus sequences. *Nucleic Acids Research*, *18*(20), 6097–6100.

Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, *45*(11), 2673–2681. https://doi.org/10.1109/78.650093

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. *2017 IEEE International Conference on Computer Vision (ICCV)*, 618–626. https://doi.org/10.1109/ICCV.2017.74

Shrikumar, A., Greenside, P., & Kundaje, A. (2017). *Reverse-complement parameter sharing improves deep learning models for genomics* [Preprint]. Bioinformatics. https://doi.org/10.1101/103663

Shrivastav, M., De Haro, L. P., & Nickoloff, J. A. (2008). Regulation of DNA double-strand break repair pathway choice. *Cell Research*, *18*(1), 134–147. https://doi.org/10.1038/cr.2007.111

Simonyan, K., Vedaldi, A., & Zisserman, A. (2014). Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. *ArXiv:1312.6034 [Cs]*. http://arxiv.org/abs/1312.6034

Simonyan, K., & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. *ArXiv:1409.1556 [Cs]*. http://arxiv.org/abs/1409.1556

Smallwood, S. A., Lee, H. J., Angermueller, C., Krueger, F., Saadeh, H., Peat, J., Andrews, S. R., Stegle, O., Reik, W., & Kelsey, G. (2014). Single-Cell Genome-Wide Bisulfite Sequencing

for Assessing Epigenetic Heterogeneity. *Nature Methods*, *11*(8), 817–820. https://doi.org/10.1038/nmeth.3035

Smit, A. F. A. (1993). Identification of a new, abundant superfamily of mammalian LTR-transposons. *Nucleic Acids Research*, *21*(8), 1863–1872. https://doi.org/10.1093/nar/21.8.1863

Smith, T. F., & Waterman, M. S. (1981). Identification of common molecular subsequences. *Journal of Molecular Biology*, *147*(1), 195–197. https://doi.org/10.1016/0022-2836(81)90087-5

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, *15*(56), 1929–1958.

Sundararajan, M., Taly, A., & Yan, Q. (2017). Axiomatic Attribution for Deep Networks. *Proceedings of the 34th International Conference on Machine Learning*, 3319–3328. https://proceedings.mlr.press/v70/sundararajan17a.html

Sung, P., & Klein, H. (2006). Mechanism of homologous recombination: Mediators and helicases take on regulatory functions. *Nature Reviews Molecular Cell Biology*, *7*(10), 739–750. https://doi.org/10.1038/nrm2008

Thibault-Sennett, S., Yu, Q., Smagulova, F., Cloutier, J., Brick, K., Camerini-Otero, R. D., & Petukhova, G. V. (2018). Interrogating the Functions of PRDM9 Domains in Meiosis. *Genetics*, *209*(2), 475–487. https://doi.org/10.1534/genetics.118.300565

Úbeda, F., & Wilkins, J. F. (2011). The Red Queen theory of recombination hotspots. *Journal of Evolutionary Biology*, *24*(3), 541–553. https://doi.org/10.1111/j.1420-9101.2010.02187.x

Umarov, R. Kh., & Solovyev, V. V. (2017). Recognition of prokaryotic and eukaryotic promoters using convolutional deep learning neural networks. *PLOS ONE*, *12*(2), e0171410. https://doi.org/10.1371/journal.pone.0171410

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention Is All You Need. *ArXiv:1706.03762 [Cs]*. http://arxiv.org/abs/1706.03762

Vylomova, E., Rimell, L., Cohn, T., & Baldwin, T. (2016). Take and Took, Gaggle and Goose, Book and Read: Evaluating the Utility of Vector Differences for Lexical Relation Learning. *ArXiv:1509.01692 [Cs]*. http://arxiv.org/abs/1509.01692

Wang, M., Tai, C., E, W., & Wei, L. (2018). DeFine: Deep convolutional neural networks accurately quantify intensities of transcription factor-DNA binding and facilitate evaluation of functional non-coding variants. *Nucleic Acids Research*, *46*(11), e69–e69. https://doi.org/10.1093/nar/gky215

Wang, S., Li, B. Z., Khabsa, M., Fang, H., & Ma, H. (2020). Linformer: Self-Attention with Linear Complexity. *ArXiv:2006.04768 [Cs, Stat]*. http://arxiv.org/abs/2006.04768

Watson, J. D., & Crick, F. H. C. (1953). Molecular Structure of Nucleic Acids: A Structure for Deoxyribose Nucleic Acid. *Nature*, *171*(4356), 737–738. https://doi.org/10.1038/171737a0

Weitzman, M. D., & Fradet-Turcotte, A. (2018). Virus DNA Replication and the Host DNA Damage Response. *Annual Review of Virology*, *5*(1), 141–164. https://doi.org/10.1146/annurev-virology-092917-043534

Whole-Genome Chromatin IP Sequencing (ChIP-Seq). (n.d.). 3.

Xiong, H. Y., Alipanahi, B., Lee, L. J., Bretschneider, H., Merico, D., Yuen, R. K. C., Hua, Y., Gueroussov, S., Najafabadi, H. S., Hughes, T. R., Morris, Q., Barash, Y., Krainer, A. R., Jojic, N., Scherer, S. W., Blencowe, B. J., & Frey, B. J. (2015). The human splicing code reveals new insights into the genetic determinants of disease. *Science*, *347*(6218), 1254806. https://doi.org/10.1126/science.1254806

Zaheer, M., Guruganesh, G., Dubey, A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., & Ahmed, A. (2021). Big Bird: Transformers for Longer Sequences. *ArXiv:2007.14062 [Cs, Stat]*. http://arxiv.org/abs/2007.14062

Zeng, H., Edwards, M. D., Liu, G., & Gifford, D. K. (2016). Convolutional neural network architectures for predicting DNA–protein binding. *Bioinformatics*, *32*(12), i121–i127. https://doi.org/10.1093/bioinformatics/btw255

Zeng, J., & Yi, S. V. (2014). Specific Modifications of Histone Tails, but Not DNA Methylation, Mirror the Temporal Variation of Mammalian Recombination Hotspots. *Genome Biology and Evolution*, *6*(10), 2918–2929. https://doi.org/10.1093/gbe/evu230

Zhou, J., & Troyanskaya, O. G. (2015). Predicting effects of noncoding variants with deep learning–based sequence model. *Nature Methods*, *12*(10), 931–934. https://doi.org/10.1038/nmeth.3547

Zitnik, M., Nguyen, F., Wang, B., Leskovec, J., Goldenberg, A., & Hoffman, M. M. (2019). Machine learning for integrating data in biology and medicine: Principles, practice, and opportunities. *Information Fusion*, *50*, 71–91. https://doi.org/10.1016/j.inffus.2018.09.012