RADBOUD UNIVERSITY NIJMEGEN

MASTER THESIS

Predicting chromatin accessibility with convolutional neural networks

Author:
Cas VAN DEN BOGAARD

Supervisors:
Dr. Kees ALBERS
Prof. dr. Tom HESKES

July 10, 2017

Radboud University Nijmegen

Abstract

Predicting chromatin accessibility with convolutional neural networks

by Cas VAN DEN BOGAARD

We have trained convolutional neural networks to predict chromatin accessibility, hoping to study the role of different transcription factors across a variety cell types. To learn in which way these networks represent transcription factor motifs, we have generated our own data, providing a ground truth for the patterns and interactions that are relevant for the prediction. By training several networks on this self-generated data, we learned that the network uses some of its convolution filters to detect transcription factor motifs, while other filters are used to detect a combination of transcription factor motifs. Factorization machines, which we implemented as a neural network layer, are used to model non-linear interactions between transcription factors in self-generated data. By comparing networks with and without factorization machines, we showed that interactions between transcription factors might be relevant when predicting chromatin accessibility. We have also created the motif building network, which appears to be able to handle positional dependencies in transcription factor binding patterns better than more traditional architectures. However, more research is necessary to show its effectiveness on real-world data.

Chapter 1

Introduction

Genetics research has become more and more important over the past decades, following advances in the techniques that allow us to sequence DNA. The Human Genome Project, a decade long, multi-billion project, concluded in 2003, having completed the first full mapping of the DNA of a human. Since then, costs of sequencing have dropped dramatically, allowing anyone these days to have their DNA sequenced for less than $1000. This cheaper sequencing of the DNA allows for genome-wide association studies (GWAS) to be done, in which differences in the genome of individuals can be linked to various traits. In 2006, Shinya Yamanaka produced his Nobel prize-winning paper [1] in which he shows how mature cells can be converted to stem cells, allowing for new ways to study cells and paving the way for research into therapies to cure genetic diseases.

Even though huge steps have been taken in understanding the human genome, there is still much work to be done. Uncovering the hugely complex systems responsible for translating the DNA into a functional organism is a daunting task. While the mechanisms of transcription and translation, through which proteins are produced, are well understood, the way that these processes are regulated is not. Transcription factors, proteins with specific binding sequences, can prevent or encourage the start of transcription. The functions of many of these transcription factors are not yet well understood, and neither are the interactions between them. Furthermore, the complex three dimensional structure of the DNA can cause different regions to be inaccessible for transcription factors, or it can bring regions that are many basepairs apart closer, allowing for interactions that would seem distant when simply looking at the DNA sequence.

The subject of this thesis will be that of trying to uncover the influence that different transcription factors have on the accessibility of the chromatin, which transcription factors are indicative of different cell types and what role interactions between transcription factors play. We will use convolutional neural networks (CNNs) to model the accessibility of various regions of the chromatin in different cell types and attempt to interpret the relations that these models have learned. In the last decade, CNNs have become some of the most popular tools in the field of machine learning. They are at the basis of many state-of-the-art models in computer vision, but are also applied in a wide array of other fields, such as speech recognition/synthesis and bioinformatics. CNNs have recently been used to predict chromatin accessibility in different cell types by directly looking at DNA sequences, showing state-of-the-art performance. They are, however, notorious for their difficulty to interpret. This is due to the large number of parameters that are tuned and due to operations that improve model performance but hurt interpretability, such as drop-out. This results in models where, even though one can follow all individual steps in the network and realize how the output value is constructed, the “reasoning” behind the parameters

of the model is unclear.

In this thesis, there will often be mentions of the interpretability of trained networks or the convolution filters that these networks have learned. In those cases, interpretability refers to the ease with which patterns that are relevant in biology can be extracted from the network. When a convolution filter corresponds to a known pattern, it is very interpretable. A filter which is a combination of multiple biologically relevant patterns is more difficult to interpret, and a filter which is used by the network in some non-linear way to interact with the output of other filters is the least interpretable. We will attempt to construct our networks in such a way that the interpretability becomes high, hopefully allowing us to learn the biological reasoning behind the relations that the networks learn.

Chapter 2

Background

2.1 Biological background

2.1.1 DNA and genetics

Deoxyribonucleic acid (DNA) is the carrier of evolutionary information for all life on earth. It is made up of a long chain of paired nucleotides, of which there are 4 types, each characterized by a single nucleobase. Adenine and Thymine form a pair, as do Cytosine and Guanine. With the choice of four different nucleotides, each position in the DNA sequence can hold two bits. For humans, the total length of the DNA sequence is around 3 billion base pairs, which contain sections that code for certain proteins, sites where regulatory elements bind and parts with a variety of other functions.

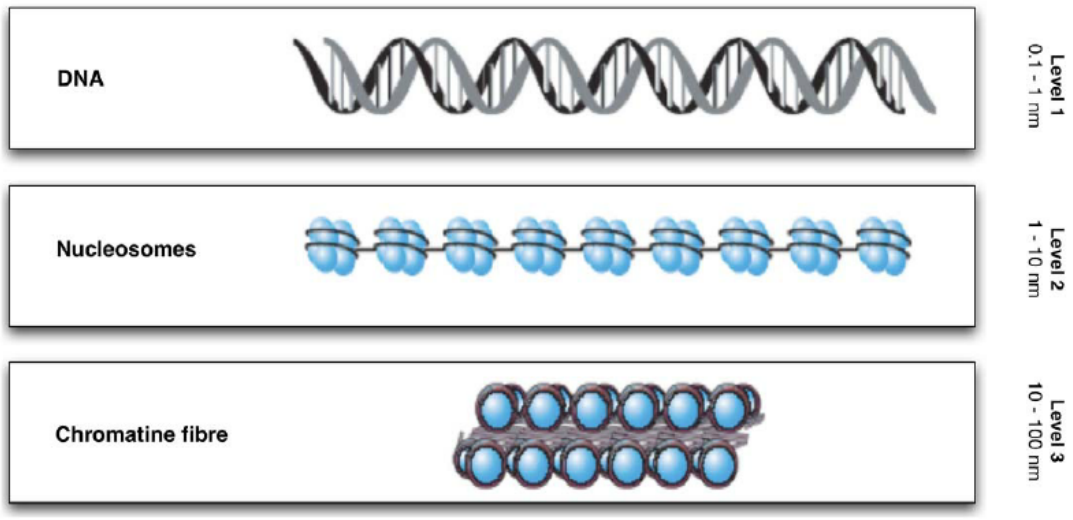

FIGURE 2.1: Chromatin structure, highlighting three states in which DNA can be present in the cell. Image produced by Santoni et al. [2]

DNA is a very long molecule and, because it is to be copied and read in a structured fashion, needs to exist in a compacted form. The DNA wraps around proteins called histones, forming nucleosomes. In this relatively open structure seen in Figure 2.1, also called "beads on a string", proteins can easily bind to the DNA sequence. The DNA can be further compacted by wrapping into a tightly nit fiber, which prevents most proteins from binding to the DNA. These different chromatin stages, each with a different accessibility, can be present at the same time.

2.1.2 Gene expression

The first step of gene expression is transcription, in which various proteins interact with the DNA sequence to create RNA. RNA is a single-strand molecule that contains the same information as the part of the DNA from which it is copied. Some of this RNA, called messenger RNA, is then used in the creation of proteins, as it codes for the sequence of amino-acids in the protein. These proteins will then go on to perform a variety of tasks that are essential to life.

This process of transcription requires many complicated interactions between the DNA and various proteins. The enzyme RNA polymerase can bind to a specific sequence next to a gene, called the promoter. If bound to the promoter, the RNA polymerase can start copying the information in the DNA into RNA. This binding, however, is influenced by other proteins, which can bind to specific sequences in the DNA. These proteins with sequence-specific binding sites are called transcription factors (TFs) and perform a variety of functions. They can function as activator, binding to an enhancer-site and promoting the binding of RNA polymerase to the promoter region by bending the enhancer to which it is bound towards the promoter. They can also function as repressor, preventing transcription by physically blocking RNA polymerase from moving across the gene, or they can up- or down-regulate transcription by altering the shape of the chromatin or by recruiting other proteins to perform any of the aforementioned tasks. While the effects of these transcription factors are important, little is known about the exact function each of the hundreds of transcription factors in the human body has. Interactions between the TFs could increase the complexity of this problem even further.

Transcription factors have sequence-specific binding sites, meaning that they bind more easily to some sequences than to others. In this paper we refer to the sequences that these transcription factors prefer to bind to are referred to as transcription factor motifs.

2.1.3 Transdifferentiation

Even though every cell (excluding reproductive cells) contains the same DNA sequence, their behavior varies dramatically. Thanks to the regulatory processes explained in the previous section, different genes can be switched on or off, allowing each type of cell to be tailored to their own function. This behavior, however, is not fixed at the start of a cell's life. Some cells have the ability to transform into other cells in a process called *differentiation*, an example of which would be stem cells, which can differentiate into all cell types of the organ from which they originate. One can imagine that pluripotent cells, which have the ability to differentiate into any type of adult cell, can have many uses. This is the reason that in 2012 the Nobel Prize in Physiology and Medicine was awarded to John Gurdon and Shinja Yamanaka, who found a way to turn fibroblasts into pluripotent stem cells [1]. One of these induced pluripotent stem cells (iPSCs), can be differentiated to any other cell, thus allowing experimentation on different cell types by only taking a sample of skin cells. To do this, the genes for 4 specific transcription factors are inserted into the DNA of the cells, forcing the cells to produce these transcription factors. The processes that are a result of these transcription factors being produced are then responsible for reversal of the process of differentiation. After this process, a similar procedure can be employed, using a different set of transcription factors to force the iPSC to differentiate to the desired cell type.

A more efficient way of going from a mature cell type to another mature cell type would be to bypass the pluripotent cell type entirely, since their production takes a long time. This direct differentiation between mature cells is called transdifferentiation. The problem is that for each different transdifferentiation, one needs to know which set of transcription factors is best suited to produce the final cell. With hundreds of transcription factors and many cell types one would like to create, finding the correct set of transcription factors can be a tiresome process.

2.1.4 Sequencing methods

Whether a protein can bind to the DNA depends on several factors. Some proteins can only bind to specific sequences, thus only binding in places where this sequence is present. It then also depends on the structure of the DNA, because parts of the chromatin that are not accessible cannot be bound, even if the sequence matches the binding site of the protein. DNase-Seq is a technique which maps this accessibility, making use of the DNase I enzyme. This enzyme can bind to any part of the DNA that is accessible and as such, the binding sites of this enzyme, called DNase I hypersensitive sites (DHSs), correspond to the parts of the DNA that are in an open chromatin region. By mapping the DNase-Seq signal to the DNA sequence, one has a way of measuring chromatin accessibility for all positions in the DNA sequence. The DNase-Seq data used in this thesis is provided by two consortia: ENCODE [3] and Roadmap Epigenomics [4]. Both of these consortia aim to create a comprehensive set of data on the topics of gene regulation and epigenomics, allowing researchers to study the complex processes that play a role in genetics. Combining both these resources provides us with a dataset that contains chromatin accessibility data for 164 different cell types.

An alternative for DNase-seq is ATAC-seq. Just like DNase-Seq, it can be used to study the accessibility of different chromatin regions. One of the advantages of ATAC-seq over DNase-Seq, is that the ATAC-seq protocol does not require many cells for an accurate sequencing. Whereas DNase-Seq is typically done on millions of cells, one only needs 50.000 cells to run an ATAC-seq experiment [5]. Although for the purpose of this thesis it is enough to known that both methods are acceptable ways of mapping chromatin accessibility, an overview of the differences between DNase-Seq, ATAC-seq and other similar methods can be found in a paper by Meyer and Liu [6]. The research group op Kees Albers has collected ATAC-seq data for several different cell types, aiming to study the transdifferentiation of fibroblasts to neuronal cells. This has resulted in a dataset containing information on chromatin accessibility in fibroblasts, iPSCs and iNeurons. These iNeurons are neuronal cells derived from iPSCs. Along with the data for these three cell types, there is also ATAC-seq data for three different transdifferentiation products. These cells are fibroblasts in which different transcription factors are overexpressed, resulting in cells that longer behave like fibroblasts, but are also not yet fully neuronal.

2.2 Machine Learning

2.2.1 Artificial Neural Networks

Artificial neural networks (ANNs) are models that derive their name from their similarity to the brain. Where the brain consists of neurons that are interconnected by their synapses, an ANN consists of interconnected *units*. The connections between these units are weighted, allowing the network to learn complex interactions by tuning these weights. This tuning is referred to as *training* and is usually done iteratively, using gradient-based methods to update the weights.

In feed-forward networks, the units are grouped in layers. In each layer, starting in the input-layer, the output of each unit is calculated based on the outputs of the units in the previous layer and the weights between the layers. This way, information is propagated forward through the network. During training, the output of the network for a certain input can be compared to the known true label corresponding to this input. To obtain a good model, the weights need to be updated so that the difference between the network output and the true labels is minimized. The function used to determine this difference is called the *loss function*. To train the network, input samples for which the true labels are used as input for the network and the outputs are calculated (the forward propagation). Then, the gradients of the loss function with regards to the weights are calculated through *backpropagation*. This backpropagation allows for a computation of the gradients in linear time w.r.t. the number of weights. Using a gradient descent procedure, the weights are then updated for the next iteration. The procedure used to train the networks for this thesis is RMSprop [7], in which the learning rate is adapted during training.

There are many different ways to connect the units between layers in a feed-forward neural network. The most straightforward way is to simply have every unit in a layer be connected to every unit in the next layer. This type of layer is called fully-connected or *dense*(see Figure 2.2). The output of a unit will then be calculated as the weighted sum of all its inputs. That is:

$$x_i^{(l+1)} = \sum_{j=1}^{N} w_{ij}^{(l)} x_j^{(l)} \tag{2.1}$$

where $x_i^{(l)}$ is the output value of a unit i in layer l, N is the total number of units in layer l and $w^{(l)}$ are the weights between layers l and $l+1$.

Dense layers allow neural networks to learn very complex models, since each unit can interact with every other unit. However, this is also where its problems are: the number of connections, and thus the number of weights that need to be learned increase quickly with increasingly large networks, which makes training more difficult. Another issue, which can easily be fixed, is that simply stacking dense layers does not increase model complexity. A dense layer can be seen as a simple matrix multiplication of the inputs with their corresponding weights. This means that two stacked dense layers can always be replaced by a single dense layer. To combat this, a non-linear *activation function* is applied to the output of each unit. This introduces the non-linearity that is necessary to build a model that can handle complex problems. The function used to introduce this non-linearity is the Rectified Linear Unit (ReLU), which sets all negative output values to zero, while doing nothing with positive output values.

Convolutional Neural Networks

In recent years, convolutional neural networks (CNNs) have proven to be very effective across a variety of classification and regression tasks, including image recognition [8][9] and natural language processing [10]. A CNN is a feed-forward neural network which employs special convolutional layers to reduce the number of weights to be learned. It does this by exploiting the translational invariance that is present in many problems.

Figure 2.3 shows how a convolution layer works. Instead of connecting every unit in a layer to every unit in the next layer, as is the case for dense layers, a convolutional layer uses several kernels of weights that are moved across the input. Each time the kernel is moved, the weighted sum is calculated for the part of the input that is being overlapped by the kernel. This set of sums forms the output for the kernel. Due to the kernels being much smaller in size than the input, the number of parameters that must be learned is smaller. Because the problem was assumed to be translationally invariant, this does not reduce the amount of complexity that the network can learn.

A slightly different type of convolution is the dilated convolution, the filters of which contain gaps in regular intervals. For each input value that the filter looks at, it skips the next n, where n is the dilation rate. An example of this can be seen in Figure 2.4. One could use dilated convolutions to enlarge the total receptive field without increasing the number of weights have to be learned, sacrificing resolution in the process. In this thesis, however, dilated convolutions will be used in an architecture in which each input value represents a small subsequence, resulting in adjacent input values sharing information about adjacent positions in the DNA sequence. Due to the skips in the dilated convolutions the network does not look at overlapping subsequences.

2.2.2 Factorization Machines

Factorization machines, first introduced by Steffen Rendle [11], are an effective method for modeling feature interactions. Where a naïve model would include a parameter for each feature interaction, leading to a model complexity quadratic in the number of features n, factorization machines allow for k parameters per feature. The interaction between features happens solely through these weight vectors, thus leading to a model with $\mathcal{O}(nk)$ parameters. This type of interaction has been shown to work well for sparse data, because two features can interact indirectly, without the need for a datapoint in which the interaction is learned directly.

The model is defined as follows:

$$y(\mathbf{x}) := w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j \tag{2.2}$$

Here, y is the prediction for a given input $\mathbf{x}$. w_0 is the bias, w_i are the weights for the linear part of the model and $\mathbf{v}_i$ are the interaction vectors corresponding to the ith feature. This formula can be rewritten, as shown in [11], to not contain a double sum over the number of features:

$$y(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + \frac{1}{2} \sum_{f=1}^{k} \left(\left(\sum_{i=1}^{n} v_{i,f} x_i \right)^2 - \sum_{i=1}^{n} v_{i,f}^2 x_i^2 \right) \tag{2.3}$$

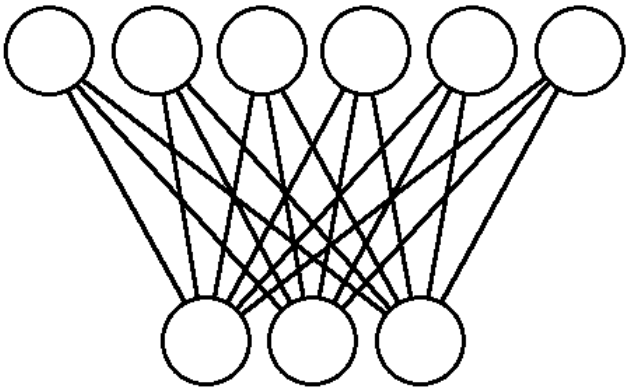

FIGURE 2.2: An example of a dense layer. Each node in one layer is connected to every node in the next. Every connection, represented by the lines, corresponds to a weight that the neural network can learn.

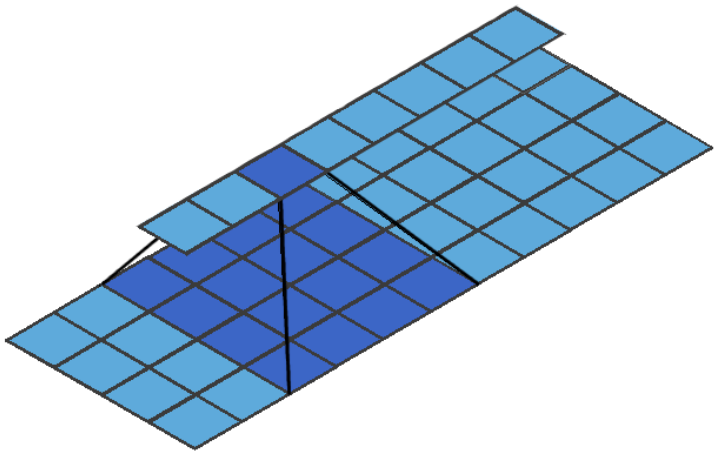

FIGURE 2.3: An example of a convolution. The input is a 4 by 11 matrix, which is "scanned" by a kernel of size 4 by 4. Typically, a convolution layer has multiple kernels.

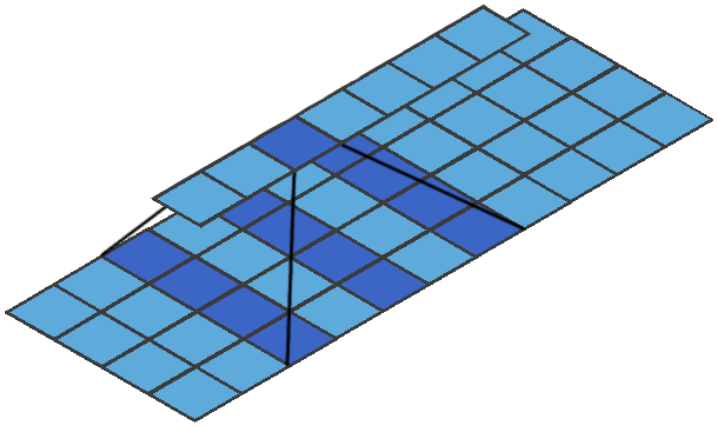

FIGURE 2.4: In a dilated convolution, the kernel has regular gaps. The example has a dilated kernel of size 4 by 3 with a dilation of 1

In this form, it is clear that $y(\mathbf{x})$ can be computed in linear time w.r.t. n and k. Thus, the output of factorization machines can be efficiently

One could also use a factorization machine as a layer in a neural network. In that case, the input $\mathbf{x}$ is given by all output values from the previous layer, whereas $y(\mathbf{x})$ is then the output of a single FM node. Using multiple factorization machines, each trained independently from the others, allows for an arbitrarily large number of output nodes for the FM layer.

Chapter 3

Methods

3.1 Models for generating sequence data

To achieve our goal of better understanding the relations that the network learns, we need to make sure that our interpretation of the relationships within the model is correct. By generating our own dataset and varying the underlying models, we can get a better understanding of the way that the network learns. It provides a "ground truth" to the relations that we are trying to uncover and perhaps the knowledge we gain from this enables us to force the network into a more interpretable representation of the model.

The data is generated in the following manner:

1. **Generate a background sequence**
   Not all positions within a DNA sequence are informative. As we are modeling the effects of transcription factor binding sites, all positions that do not contain a TF motif are non-informative and, as such, belong to the background. This background is generated first.

2. **Select motifs to insert**
   Each sequence should contain a different set of motifs, representing the sites that transcription factors can bind to. Each sequence might contain the same number of motifs, or this number could vary between sequences within the dataset. All sequences in the same dataset contain transcription factor motifs from the same pool of possible motifs.

3. **Insert selected motifs**
   Each motif is inserted into the sequence in such a way that none of them overlap. Since the motifs are represented by position probability matrices, there are two obvious ways of inserting a motif into a sequence. One would be to use the "consensus sequence", meaning that for every position we select the nucleotide with the highest probability at that position. The other option is to sample from the probabilities such that the position frequencies across the simulated dataset resemble those of the motifs.

4. **Compute sequence accessibility**
   With the full sequence generated, we can determine if it is accessible according to one of the models shown in Section 3.1.2. These accessibility models can take into account which motifs are present, how they are positioned relative to each other and if there are interactions between them. Multiple accessibility models can be used to calculate accessibility for the same sequence, allowing training to more than one target class at the same time.

3.1.1 Background models

The most straight-forward method of generating a background is to assume that each position is independent of the others. For every position, each nucleotide can be chosen with a fixed probability. One can also think of more complicated ways of generating the background sequence, in which positions are not independent of each other, or with varying nucleotide probabilities between sequences. These more complicated methods, however, are out of scope for this thesis, since they introduce extra parameters to the experiments, which will only make our analyses more difficult. All experiments instead use the simple method in which, in every position, each nucleotide has an equal probability of being chosen.

3.1.2 Chromatin accessibility models

The exact mechanisms behind chromatin accessibility are still not uncovered and as such, it is difficult to describe a model for the effects that transcription factors have on the accessibility of a given sequence. There are several models that try to explain how the occurrences and positions of motifs could influence the accessibility of a sequence. Instead of attempting the daunting task of trying to find out which of these models best matches reality, we have implemented several of these models. By studying different accessibility models, we try to get a better understanding of the behavior of the neural networks used. At the very least it should possible to find out if the networks used can learn the different accessibility models.

The additive model

In the additive model, the assumption is that each occurring motif has a linear contribution to the accessibility of a sequence. The contributions of all occurring motifs are summed and, when a specific threshold is met, the sequence is said to be accessible. In the additive model, one assumes that there are no motif interactions and the position of each motif in the sequence is irrelevant. This is most likely an oversimplification of reality, but it is the easiest model to work with.

The paired motifs model

A type of non-linear interaction that could occur, is that two transcription factors only influence accessibility when they can both bind to the sequence. To model this type of interaction, each motif in the simulation set is matched to another motif. When two of these paired motifs are present in the same generated sequence, the sequence is said to be accessible. If non of the motifs in a sequence belong to the same pair, the sequence is said to be inaccessible.

3.2 Neural network architectures

TABLE 3.1: Different network architectures used during this thesis

Baseline network	
Convolution	Kernel width equal to that of motifs
ReLU	
AveragePooling	Pooling across the whole input
Dense	One unit for each output class

Factorization machines network	
Convolution	Kernel width equal to that of motifs
ReLU	
AveragePooling	Average pooling across the whole input
Factorization machines	One FM for each output class

Motif building network	
Convolution	Small kernel of width n, initialized with all 4^n blocks
Threshold	For each output value of the convolution this returns 0, unless it is the maximum activation possible
Dilated convolution	Dilation rate n, receptive field should be approx. equal to width of motifs
ReLU	
MaxPooling	Pooling across the whole input
Dense	One unit for each output class

In the experiments described in this paper, three different network architectures are used with varying parameters. Table 3.1 shows the structure of these networks, highlighting the properties of some of the layers. The baseline network has a simple structure, with a single convolutional layer followed by a pooling operation and a single dense layer. Experiments that are designed to help us better understand neural network behavior will therefore use this network architecture. The factorization machines network is made to be comparable to the baseline network, allowing us to study the effects that the added interaction term has on the generalization performance and on its interpretability. Finally, the motif building network is designed to capture positional dependencies, as will be shown in Section 3.6. None of the networks described use drop-out, as we expect that this would hurt the interpretability of the networks.

3.3 Extracting motifs from convolution filters

Given that transcription factors play an important role in shaping the chromatin, we expect their sequence-specific binding patterns to be prevalent in the networks that we train. Given that a convolutional layer scans sections of the input sequence for short patterns, we hope that the filters learned by this layer indeed correspond to the important transcription factors. The patterns in these filters could then be compared to a database of known transcription factor motifs, providing insight into which transcription factors are relevant. The motifs of these known transcription

factors are represented by position-probability matrices (PPMs), which describe the probability for each nucleotide to be present at a given position in the binding site. Thus, a procedure is necessary to retrieve a PPM-like representation of the learned filters, so that matching to this known set of motifs can be done. To do this matching, we would like to find the "average binding sequence" for each filter and use that as PPM to match against the known PPMs. To this extent, the Basset framework[12] has implemented a procedure that passes a set of test sequences through the network, averaging over the sequences that activate the filter more than some threshold. This height of this threshold is based on the sequence with the highest activation. While this approach can work, it is computationally expensive, due to the need of running a test set through the network. The resulting PPM will also be biased towards motifs that appear often in the set of test sequences, since their high frequency increases their importance when determining the average sequence. As such, this method will not result in a PPM that best represents the information in the filter. Even if the filter activates heavily on a rare motif, the PPM will not look like this motif when there is a frequently occurring motif that also matches the filter relatively well.

Because of the aforementioned problems, we use a procedure to directly extract the information we need from the weights. This removes the reliance on a test set, thus circumventing the problem of introducing a bias towards frequently occurring motifs and removing the need to run test sequences through the network. The goal of this procedure to extract the patterns that are learned by the filter and format them such that they can be easily compared to the transcription factor PPMs. While we want a "matchable" result, the resulting PPM should still represent what the filter has learned. First, we first set all negative values to zero. This is done because we only want to select nucleotides which have a positive contribution to the binding of a transcription factor. Then there is a column-wise normalization to turn the positive weights into a PPM. This means that the probabilities for each position in the binding sequence sum to 1. The resulting PPM can then be fed to TomTom [13], which is a tool for comparing PPMs to a database of known transcription factor motifs.

The same procedure can be used to match the filters of networks trained on self-generated data to the motifs that were included. However, because there are relatively few motifs relevant for the prediction and because those motifs are all consensus sequences that are known beforehand, we can improve on the matching done by TomTom. In this improved procedure, for each position in the filter it is determined which nucleotides the filter is looking for in that position. A nucleotide is said to be look for a nucleotide, when its contribution to the information content of that position is larger than some threshold. This means that a nucleotide needs to have a relatively high value in the PPM and the position needs to have a high information content. This results in a list of nucleotides present for each position, as can be seen in Figure 3.1, which is compared to the list of relevant motifs. This comparison is done by sliding the motifs across the filter to find the best match, counting the number of nucleotides that are both present in the filter and the motif at each position. This count is then normalized by dividing through the maximum count a filter could get, which is either the filter length or the motif length, depending on which one is smaller. This results in a matching score for each combination of a filter and a motif. If this score is higher than some threshold, the filter and motif are said to match.

The thresholds for contribution to the information content and matching score are chosen after visually comparing learned filters to the motifs present in the data. The information content threshold is chosen to be 0.1, so as to filter out positions in which all nucleotide probabilities fluctuate around 0.25. The threshold for matching

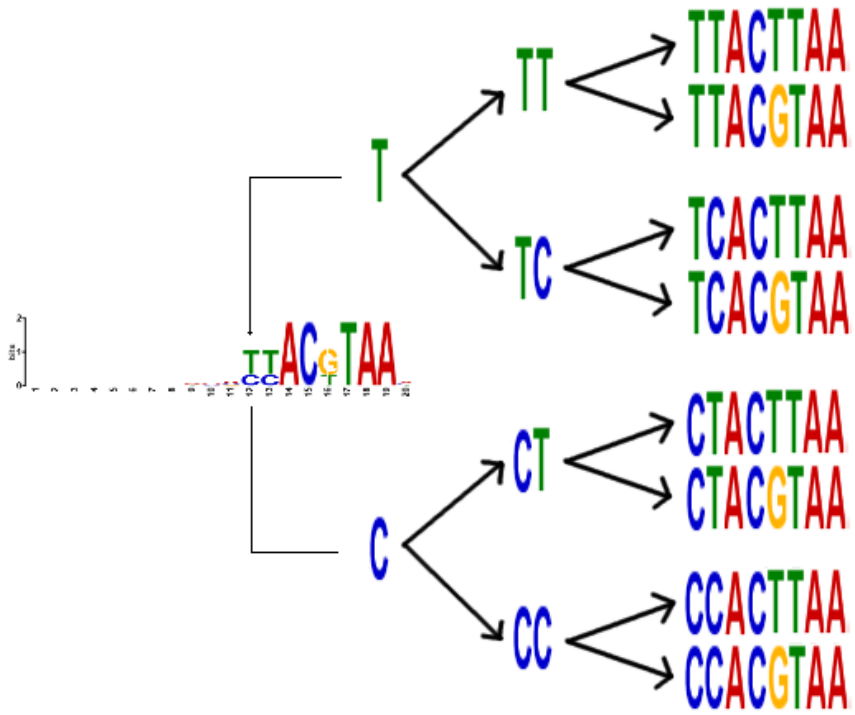

FIGURE 3.1: Example of the method with which a list of possible motifs is obtained from a learned filter. The positions in the source motif are scaled to information content, meaning that empty positions correspond to those positions having an equal probability for each nucleotide.

score is set to 0.8, so as to not exclude the matches in which a filter has learned a significant part of a long motif. Choosing these thresholds remains fairly arbitrary, since one has to make a trade-off between precision and recall. This is made even more difficult by the fact that it is very difficult to determine whether the network is really using the matched filter to detect the matching transcription factor motif, or whether the filter is used in a different way in a later part of the network, meaning that we are never sure if a match is correct or not.

3.4 Experiments on the baseline network

3.4.1 Varying the number of convolution filters

To study the influence that the number of filters has on the interpretability of these filters, the baseline network is trained on self-generated data for a varying number of convolution filters in the first layer. Choosing this number to be smaller than, greater than, or equal to the number of motifs in the data, allows us to see what happens when the network has respectively too few, too many or just enough filters to learn a single filter for every motif. In the ideal case, the network would have learned a single motif in each of its convolution filters, allowing for easy interpretation. We expect these experiments to increase our understanding of the way in which the network represents the relevant motifs, possibly providing clues as to how to improve the interpretability of networks trained on real-world data.

3.4.2 Training on multiple targets

Because some transcription factors have approximately the same effect on the accessibility for a given accessibility model, we expect that the network has little incentive to learn the motifs for those transcription factors separate from each other. This corresponds to the real-world case where certain transcription factors can behave in a similar way in a certain cell type. However, their existence suggests that there are cases in which they have a different function. By extending the dataset with multiple target classes, we attempt to force the network to distinguish between transcription factors that have a similar function in one target class. The expectation is then that this will lead to models that are easier to interpret, because more motifs end up in their own convolutional filter. With self-generated data, one can simulate more "cell types" by calculating the accessibility score several times for each sequence, using a different accessibility model for each score. This reduces similarities between the function of motifs, because they might have roughly the same contribution in one target class, while having different contributions in another. While this is an efficient way of decorrelating motif functions, it does assume that the similarity between transcription factor function is independent of the similarity of their motifs, which is not necessarily the case.

3.5 Using factorization machines to discover interactions

3.5.1 Experiments on the factorization machines network

We hope to discover transcription factor interactions by utilizing the capability of the factorization machines to capture non-linear interactions. To study how certain interactions are represented in the network, we will look at the interactions learned after training the factorization machines network on self-generated data. Accessibility is determined with the paired motifs model, meaning all interactions present in the data are known. With this ground truth, the interactions found from the factorization machines can be checked, providing validation for the methods of retrieving these interactions.

We will see that, in comparison to the previous experiment, the convolution filters that the network has learned are harder to interpret. This would hinder the search for interactions in the factorization machine, because we are looking at the interactions between these filters. Not knowing what patterns these filters are looking for makes it hard to determine what the interactions in the factorization machines represent. Even though the network is adept at using these unintuitive filters, we prefer filters that are more easily interpretable. To counteract this problem of interpretability, training of the network is repeated after replacing the convolution filters with those from another network, in which the convolution filters are more easily interpretable. Fixing these filter weights ensures that they are not changed and therefore remain interpretable. One can expect a drop in performance with this method, since the network is no longer allowed to tune a part of its weights. However, because these weights originate from a network trained on a similar dataset, this drop might be small enough that using this method is worth it for the increased interpretability.

To study which interactions the network finds and whether the same interactions are learned consistently, the factorization machines network is trained in 10-fold cross-validation. Each fold uses the same convolution filter weights, originating from one of the baseline networks. This results in several factorization machines

that are trained on the same inputs and outputs. Having several of these factorization machines trained using the same convolution filters, we are able to study the consistency with which the network represents important interactions. We expect these factorization machines to value interactions between paired motifs highly, because those are the only interactions that are relevant for the prediction.

3.5.2 Proof of non-linear interactions

The baseline and factorization machines networks are designed to be comparable, the only difference being the interaction term that is introduced in the factorization machine. The baseline network, containing just a dense layer after the average pooling operation, can only combine the filter activations in a linear fashion to create the predictions and, as such, should not be able to capture the non-linear interactions between transcription factors that might be present. The factorization machines network, however, should be able to learn those interactions. To verify this reasoning, a baseline network is trained on a self-generated dataset in which the accessibility for each sequence is determined with the paired-motifs model. This network can then be compared to the factorization machines networks trained in the previous experiment. The expectation is that the baseline network will not be able to make a prediction that is much better than a random guess, whereas the factorization machine network should be able to predict accessibility relatively well.

If the transcription factor network is shown to capture non-linear interactions in the data better than the baseline network, we can test whether transcription factor interactions play a role in determining chromatin accessibility. To this extent, both networks are trained on real-world data. The factorization machines network having a generalization performance that is higher than that of the baseline network would give a strong indication that non-linear interactions between transcription factors play a role in determining chromatin accessibility.

3.5.3 Attempting to find transcription factor interactions

Using the knowledge gained from the preceding experiments, we can attempt to find potential transcription factor interactions that are relevant to chromatin accessibility in the humans. One would expect that the factorization machines network that has been trained on the real-world data has learned some of these interactions. First, we must know what patterns the convolution filters are looking for. By applying the extraction procedure explained in Section 3.3, a PPM is obtained for each of the filters. Using TomTom, those filter PPMs are then matched to the database of known transcription factors to find potential matches. Once the filters are matched, the factorization machine vectors can be studied to see which filter interactions appear to be the most important.

3.6 Capturing positional dependencies

In all previous experiments we attempt to directly learn the PPMs of important transcription factor motifs in a convolutional layer, extracting these learned patterns using the method described in Section 3.3. This approach makes sense, seeing that the databases of known motifs also contain PPMs for each of the motifs. When describing a motif with a PPM, the assumption is made that each position is independent of the others. This means that the contribution of a given nucleotide to the binding strength of a transcription factor does not depend on the other nucleotides in

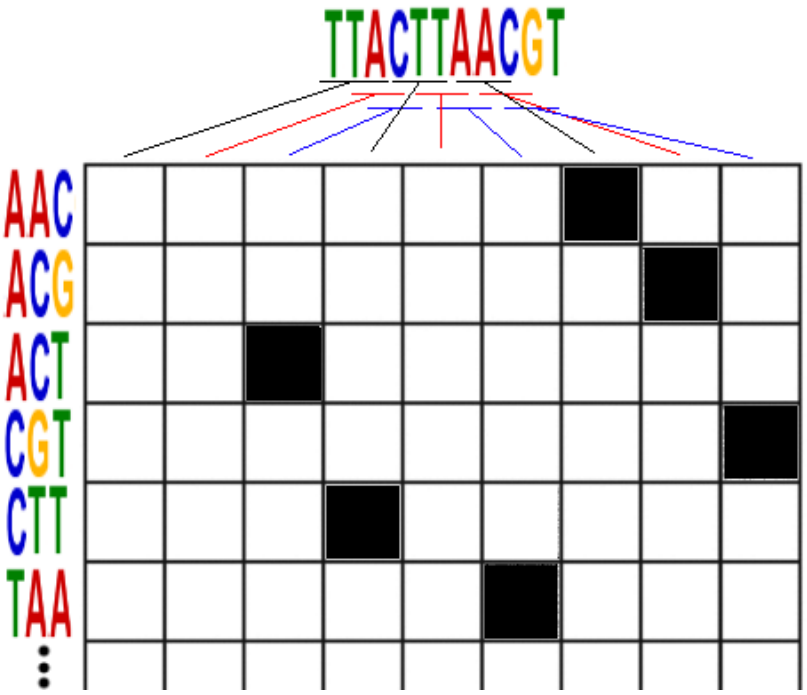

FIGURE 3.2: The transformation of the input sequence (top) to the new representation (bottom). The bottom matrix actually consists $4^3 = 64$ values for every position in the sequence, each corresponding to one of the possible 3-letter subsequence.

the motif. One can think of cases, however, where this assumption does not hold. For example, a transcription factor might bind to a sequence with either the subsequence AA or the subsequence CC in a given position. In a PPM, this can only be represented by giving the A and C equal probabilities in both positions. From this PPM, there is no way to determine that the transcription factor only binds to the AA and CC subsequences, while not binding to the subsequences CA and AC. To accurately capture this higher-order information, a different way of representing motifs is necessary, which in turn means that the network architectures that aim to directly learn motifs in a PPM-like representation also need modification. That these positional relationships are relevant for predicting chromatin accessibility is shown by Omidi and Van Nimwegen [14]. Their paper shows that models that incorporate higher-order sequence information provide a better representation transcription factor motifs than PPMs.

Table 3.1 shows the architecture with which we will attempt to tackle this problem of positional dependence in transcription factor motifs. The basic idea is to build up the motifs so that the network ends up with a PPM-like representation of each motif, but in such a way that not all sequences matching that PPM-like representation need to have a similar activation. The first two layers of this architecture transform the input sequence, which consists of a 4-dimensional one-hot vector for each position, to a sequence where each position is a 4^n-dimensional one-hot vector, as can be seen in Figure 3.2. In this transformed input, each position no longer represents a single nucleotide, but instead represents a block of n adjacent nucleotides. These blocks will serve as the "building blocks", from which the motifs will be constructed. The next layer in the network is a dilated convolution, which has a similar function to the convolutional layer in previous networks. The dilation is necessary

to prevent the filters from looking at overlapping subsequences, and results in this dilated convolution having a similar receptive field as the convolutional layers in previous models. This architecture allows us to interpret the filters as PPMs, while granting the network the ability to incorporate positional dependencies.

As a proof of concept, small versions of both this network and the baseline network will be trained on self-generated data, containing motifs with a variety of consensus sequences. These different consensus sequences are derived from the same PPM, but have a different contribution to the accessibility. This forces the networks to be able to discriminate between different versions of the same motif. The hypothesis is that the baseline network will be outperformed by the motif building network. The baseline network has a single filter available for each motif, but it also needs to be able to separate the different versions of each motif, which it should not be able to do with only a single filter. Of course, the baseline network should still be able to do better than random guessing, possibly by combining motifs in each of the filters or by only trying to recognize some of the motifs. The motif building network, on the other hand, should be able to differentiate between the different versions of each motif, leading to a better prediction than that of the baseline network.

Having shown that the motif building network can outperform the baseline network in the case of motifs with positional dependencies, we can use it to predict chromatin accessibility on the real-world data. By training a network with an amount of dilated convolution filters that is similar to the number of convolution filters used by the networks of the experiment in Section 3.5.2, we can compare the performance of the motif building network with that of the other architectures. Because the motif building network receives more information than the baseline and factorization machines networks and is otherwise fairly similar, the network is expected to perform at least as good as those networks. Even when there are no positional dependencies within the motifs, one would expect that the motif building network has a similar performance as the baseline network, because it should be able to learn the same relationships linear relations as the baseline network.

Chapter 4

Experimental setup

4.1 Data

The data that will be used in the experiments in this thesis is split into two categories: self-generated data and real-world data. The real-world data consists of two separate datasets, retrieved using different sequencing methods. One is the in-house dataset consisting of ATAC-seq data for 6 different types of cells, as previously highlighted in the biological background: fibroblasts, iNeurons, iPS cells, fibroblasts and 3 transdifferention products. ATAC-seq is a genome-wide assay, meaning that ATAC-seq signal provides a measurement of chromatin accessibility for each nucleotide in the DNA sequence in each of these cell types. For each cell type, the ATAC-seq signal is thresholded, resulting in a binary classification of accessibility with single nucleotide resolution. Open regions are merged into the same peak when they are located within 200 nucleotides of each other. Overlapping peaks between cell types are also merged into peaks that are said to be open in both cell types. Then, to provide us with input sequences of consistent length, a 600 nucleotide sequence around each peak in the joint peak set is taken. These 600 nucleotide sequences are used as input when training the neural networks. If the peak was present in a certain cell type, the corresponding sequence is said to be accessible in this cell type. For the other cell types, the sequence is said to be inaccessible. In total, this results in slightly over 640.000 sequences of 600 nucleotides, each of which corresponds to an open chromatin region in at least one of the cell types.

The second dataset used is the combination of data from the ENCODE and Roadmap Epigenomics consortia. The sequences and their accessibilities are obtained in a similar way as those for the in-house dataset. Following the same procedure of peak-calling and merging peaks, there is chromatin accessibility data for 164 different cell types, resulting in over 2 million input sequences that are accessible in at least one of the cell types.

The datasets for those networks that are trained with self-generated data are, unless otherwise specified, sets containing 100.000 sequences with a length of 600 nucleotides. Each sequence contains exactly three motifs of up to 20 nucleotides, selected from a set of 20 different transcription factor motifs. The size of this dataset is chosen to not be overbearing when training, while providing enough data to be able to separate the relevant motifs from the background. Of these 100.000 sequences, 10.000 sequences are held out for testing and 9000 are held out for validation, leaving 81.000 sequences to be used as training data.

4.2 Experiments on the baseline network

4.2.1 Varying the number of filters

Self-generated data, using the additive model for chromatin accessibility, is used to train the baseline network in 10-fold cross-validation. This dataset contains a total of 20 transcription factor motifs on which the accessibility is based. A varying number of convolution filters is used: 10, 20 and 30. These values represent the case where there is a number of convolution filters less than, equal to and more than, the number of relevant motifs in the data respectively. Training is done in batches of 2048 sequences, with RMSProp as optimizer and a learning rate of 0.002. Training is stopped after 250 epochs without improvement on the validation performance, after which the weights with the best validation performance are saved.

4.2.2 Training on multiple targets

Again, self-generated data is used to train the baseline network in 10-fold cross-validation, this time using 20 convolution filters. Instead of training on a single target class, the additive model is used to generate 10 independent target classes for which the network has to predict accessibility. The procedure for training, including the parameters, is the same as in Section 4.2.1.

4.3 Finding relevant motifs

4.3.1 Experiments on factorization machines networks

The factorization machines network is trained in 10-fold cross-validation on a set of self-generated data. To introduce non-linear interactions between transcription factors, accessibility for each sequence is determined through the paired motifs model. The convolutional layer has 20 filters and the factorization machines have an interaction vector of dimension $k = 20$, both matching the number of motifs in the data. Training is done in batches of 2048 sequences, with RMSProp as optimizer and a learning rate of 0.001. Training is stopped after 250 epochs without improvement on the validation performance, after which the weights with the best validation performance are saved. This process is repeated twice, once training on a single target class and once training on 10 separate target classes. After training, the motifs are extracted from the convolution filters and matched to the motifs that are present in the data.

When the training of this network does not result in filters that are easy to interpret, training is repeated. Instead of randomly initializing the convolution weights as one would normally, they are set to be the same as the convolution weights from the networks in the experiments in Section 4.2.1. We are looking to study the interactions between filter activations that the factorization machines have learned. To study the consistency with which interactions are learned, the network is trained in 10-fold cross-validation, every network using the same convolution filters. The filters used for this weight transfer are the filters of the baseline network that has learned the most single-motif filters.

4.3.2 Proof of non-linear interactions

A baseline network network is trained on the same paired-motifs dataset that is used to train the previous factorization machines networks. This network has 20 filters in

its convolutional layer, matching the number of motifs in the data, and is trained to a single target class. Training is done in batches of 2048 sequences, with RMSProp as optimizer and a learning rate of 0.001. This baseline network is compared to the previous factorization machines, looking at their AUC on the test set as a measure of performance.

To study the interactions in real-world data, a baseline network and a factorization machines network are both trained until convergence on the ENCODE+Roadmap data. The factorization machines have interaction vectors with dimension $k = 5$. Training is done in batches of 1024 sequences, with RMSProp as optimizer and using a learning rate of 0.001. For both networks, the weights from the epoch with the lowest loss on the validation set are used to calculate the AUC on the test set. This experiment is repeated with the in-house ATAC-seq dataset.

4.3.3 Attempting to find transcription factor interactions

The learned patterns are extracted from the convolution filters of the factorization machines network trained in the previous experiment. All these extracted motifs are matched to the CisBP database [15] using TomTom, which results in a list of potential matches. For each potential match, an E-value is given, which represents the expected number of times one expects a random match, given the size of the database. We say that a filter has learned a transcription factor motif when its top match has an E-value of at least 0.01. This is a fairly high threshold, given that there are 300 filters for which we apply this test. However, filters that are not matched provide us with no information at all. We would rather incorrectly match some filters than match no filters at all, as the latter would mean that any study of the interactions between these filters is impossible. This does mean that all interactions found need to be considered with the knowledge that some of the matches are expected to be due to chance.

4.4 Capturing positional dependencies

TABLE 4.1: The three motifs used in the proof-of-concept experiment. Each motif has 4 different versions. The differences between these versions are highlighted in bold.

Motifs	Variants	Value
	TAA**CC**TAGCTGC TAA**GG**TAGCTGC	1
	TAA**CG**TAGCTGC TAA**GC**TAGCTGC	0
	CCTA**CT**CGGATA CCTA**GA**CGGATA	1
	CCTA**CA**CGGATA CCTA**GT**CGGATA	0
	**AT**GCAGTGGACT **TA**GCAGTGGACT	1
	AAGCAGTGGACT **TT**GCAGTGGACT	0

As proof-of-concept, both the baseline network and the motif building network are trained on a small set of self-generated data, consisting of 25.000 sequences of length 300. Each sequence contains exactly one variant of each the motifs shown in Table 4.1. These three motifs each have four different versions, of which two add to the accessibility, while the other two do not. Accessibility is determined using the additive model with a threshold of 1.5, meaning that a sequence is said to be accessible when at least 2 out of 3 motifs have a non-zero contribution. The motif building network uses blocks of width 4, resulting in 256 first-layer filters. The dilated convolution layer has three filters, each of width 3 and a dilation matching the width of the blocks. This leads to a filter that has a total receptive field of 12 positions. L2 regularization with $\lambda = 0.01$ is added to this second convolution. The baseline network has three first-layer filters of width 12, matching the receptive field of the motif building network.

After showing that this alternative architecture can work, a motif building network is trained on the ATAC-seq data. This network uses blocks of length 4, resulting in a total of 256 filters in the first convolution layer. The dilated convolution layer has 300 filters with a width of 5, matching both the number of filters and receptive field of the convolution layer in the baseline and factorization machines networks from the experiment in Section 3.5.2. Again, L2 regularization with $\lambda = 0.01$ is added to the dilated convolution layer.

Chapter 5

Results

5.1 Experiments on the baseline network

5.1.1 Varying the number of first layer filters

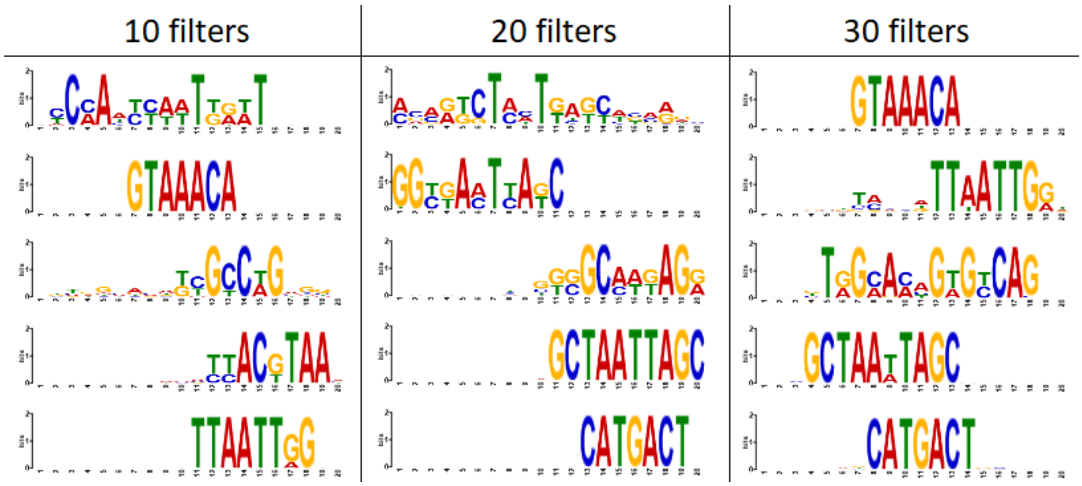

FIGURE 5.1: This figure shows a small sample of the motifs extracted from the first layer of the networks with resp. 10, 20 and 30 convolutional filters. Each position is scaled to its information content, which ranges from 0, when all 4 nucleotides have equal probability, to 2, when a single nucleotide has probability 1. Chromatin accessibility was determined with the additive model.

Three baseline networks have been trained in 10-fold cross-validation, with either 10, 20 or 30 convolution filters. The networks with 10 convolution filters reached an average AUC of 0.966 ± 0.010, while the networks with 20 and 30 filters both reached an average AUC of 0.984 ± 0.002.

Figure 5.1 shows a sample of what the motifs extracted from the learned filters look like for the three networks. The figure contains examples of filters that are easy to interpret and of filters that are hard to interpret. Some filters, such as the second filter in the 10-filter network, or the first and fifth filters for the 30-filter network, are easy to interpret. For each informative position, only a single nucleotide is learned, indicating that the filters have learned exactly one motif, which is easy to match to known motifs. Slightly more difficult to interpret are filters such as the fourth filter of the 10-filter network or the second and third filters of the 20-filter network. While it is clear that the filter does not correspond to a single motif, a combination of two of the input motifs explains the learned pattern. Filters like the third filter of the 10-filter network or the first filter of the 20-filter network are filters which do not simply correspond to one or two input motifs. Several explanations for these

filters can be thought up, such as the filter not having converged or the filter being necessary or helpful as modulation to other filters, but it is hard to determine which of these explanations is correct.

Figure 5.2 shows how well the filters from the trained networks can be matched to the input motifs. From the figure, it is clear that the interpretation of the network benefits from having enough filters to cover the range of input motifs. Increasing the number of convolution filters from 10 to 20 roughly doubles the number of interpretable filters, especially increasing the number of filters that match to a single motif. Further increasing the number of filters to 30 does lead to an increase in matchable filters, although this increase is due to an increased number of combination filters, as opposed to the more easily interpretable filters that can be matched to a single motif. As such, increasing the number of convolution filters past the number of relevant transcription factor motifs can only be helpful if one can interpret the combination motifs. Otherwise, this extra increase provides little benefit.

5.1.2 Predicting on multiple targets

Another baseline network, containing 20 convolution filters, is trained in 10-fold cross-validation. This time, there are 10 target classes for which accessibility has been predicted, each determined from a different additive accessibility model. In Figure 5.3 the number of filters matching to the relevant motifs is compared to the filters that were learned when training the 20-filter baseline network on a single target class. Whereas the average number of single-motif filters is comparable between these cases, the filters of the networks trained on 10 target classes vary more heavily. Where the networks trained on 1 target class all have between 5 and 8 single-motif filters, the networks trained on 10 classes have between 3 and 10.

5.2 Using factorization machines to discover interactions

5.2.1 Experiments on the factorization machines network

The factorization machines network is trained in 10-fold cross-validation on a simulated dataset of 100.000 sequences of 100 nucleotides long, each of which contains 3 motifs. These motifs have been randomly selected from a set of 20 motifs. Accessibility is determined with the paired-motif model. This experiment was done with both 1 target class and 10 target classes, resulting in a total of 20 networks being trained. Not a single one of the filters that are learned by these networks was a single-motif filter, as can be seen in Figure 5.4. For the networks trained on a single target class, an average of 18.9 filters can be matched to multiple motifs, whereas the networks trained on 10 target classes had an average of 16.3 combination filters. On average, these networks have reached an AUC of 0.890 ± 0.02. Because none of these filters behave in an ideal way, training is repeated, but with the first layer weights initiated by the weights from one of the baseline networks, as learned in the experiments on varying the number of convolution filters. During this repeated training, the transfered weights are fixed to prevent them from changing.

Appendix A contains a table with paired motifs and 10 tables showing the top-10 interactions for each of the factorization machine networks that have been trained in the 10-fold cross-validation. Each table contains the 10 sets of filters with the largest inner product between their interaction vectors. These tables show that the network consistently learns the same interactions, many of which can be said to be correct. On average, 7.4 top-10 interactions are correct. The remaining interactions

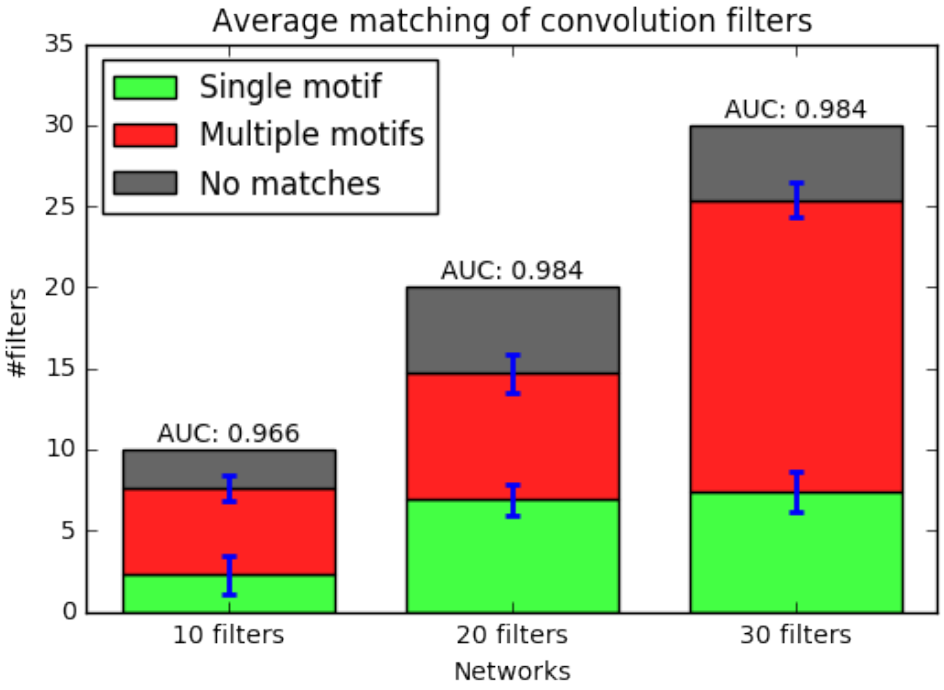

FIGURE 5.2: This figure shows how many of the filters in the baseline network can be matched on average for the three different numbers of filters. Chromatin accessibility was decided using the additive model. Top error bar is for combination filters, bottom error bar for single-motif filters.

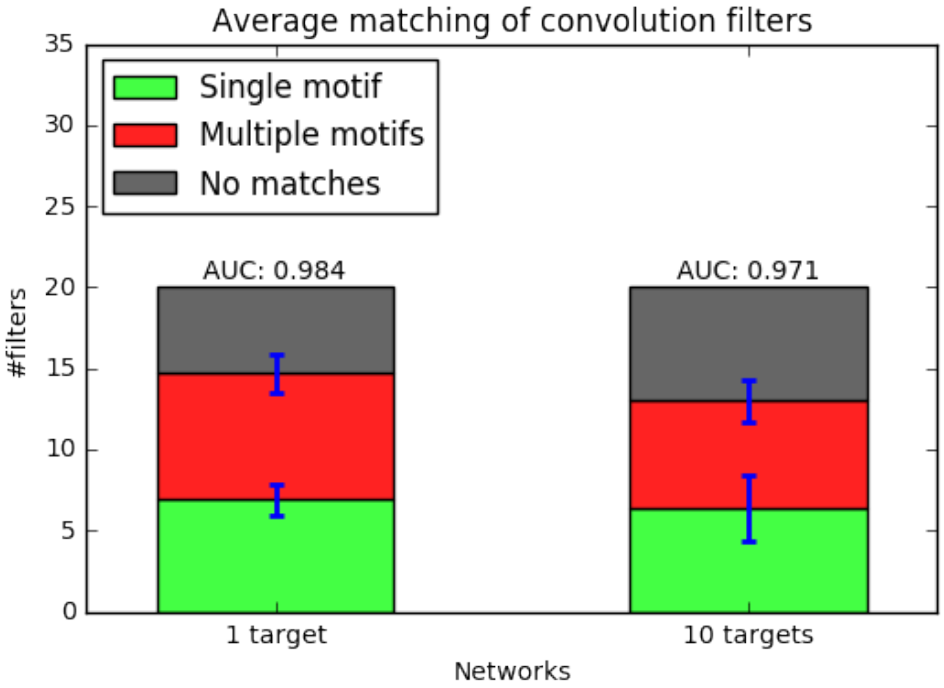

FIGURE 5.3: This figure shows how many of the convolution filters can be matched on average, when training the 20-filter baseline network on either 1 or 10 targets. Chromatin accessibility was decided using the paired-motifs model. Top error bar is for combination filters, bottom error bar for single-motif filters.

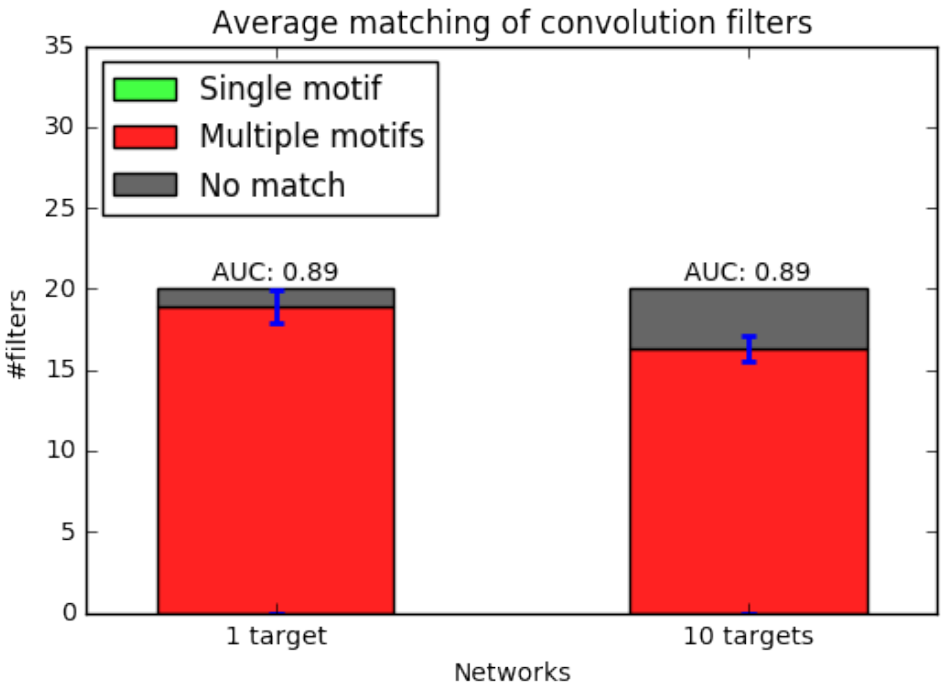

FIGURE 5.4: This figure shows how many of the first-layer filters can be matched, when training the 20-filter FM network on either 1 or 10 targets. Top error bar is for combination filters, bottom error bar for single-motif filters. Chromatin accessibility was decided using the paired-motifs model

are always interactions between filters that both match against the same motif. The learned interactions seem very stable, with the interaction between the HOXD8 and TFAP2E filters always being the most important and most of the interactions being top-10 interactions in 9 or 10 folds. Figure 5.5 shows the difference in interaction importance for those interactions that are correct and those that are not. The average weight for a correct interaction is 1284 ± 4773, whereas the average weight for an interaction that is between filters that do not correspond to a motif pair is -607 ± 1371. We apply Welch's unequal variances t-test [16] to test whether this difference is significant. The null hypothesis is that both sets are equal, which we test against the alternative hypothesis that the set of correct interactions has a higher mean than the incorrect interactions. The p-value for this test is 4.18×10^{-14}, which is low enough for us to reject the null hypothesis in favor of the alternative hypothesis.

5.2.2 Proof of non-linear interactions

A baseline network was trained on the paired motifs dataset. This network reached an AUC of 0.52, which is only a slight improvement over random guessing. The average AUC of 0.89 that the factorization machines networks managed to achieve is much higher, showing that a factorization machines network is indeed able to learn non-linear interactions in a way that the baseline network is not.

To show that non-linear interactions between transcription factor motifs are relevant when predicting chromatin accessibility, the baseline network is compared with the factorization machines network after training them on the ENCODE+Roadmap and ATAC-seq data. Both networks are trained to convergence. Figure 5.6 shows the loss during training of the networks on the ATAC-seq data. The loss-curve for the

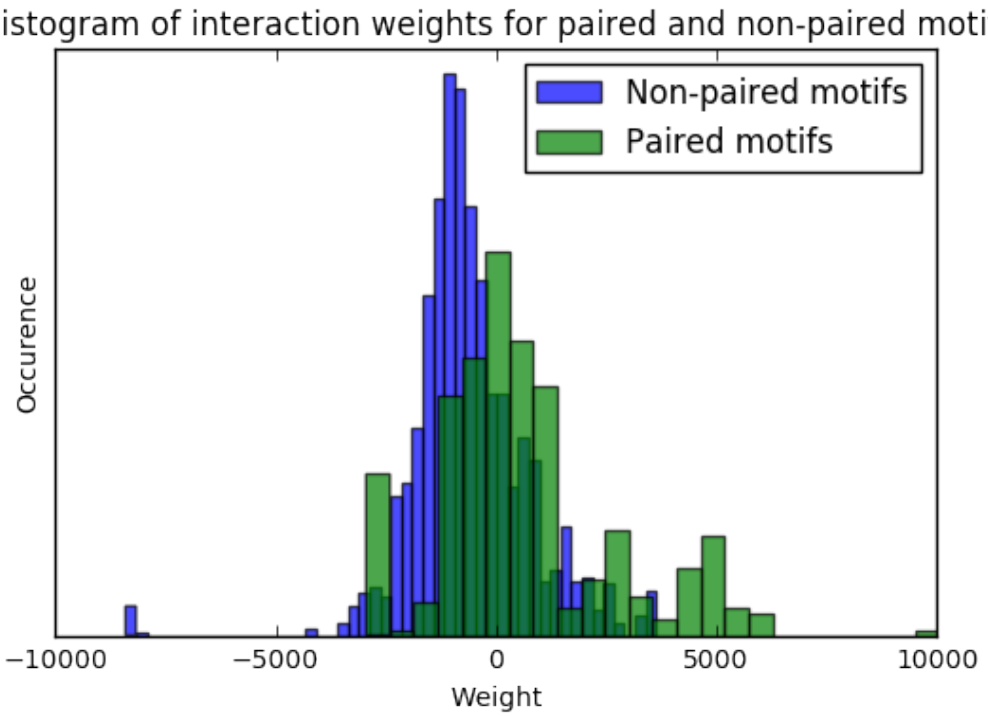

FIGURE 5.5: Histograms showing the difference in interaction weights between the interactions that are between paired motifs, and those that are not. Both histograms are normalized to correct for the difference in data points for each set. The weights shown are the inner product of the interaction vectors learned by the factorization machines across the 10 folds.

networks trained on the ENCODE+Roadmap data is similar. The baseline network reaches an average AUC of 0.834 on the ENCODE+Roadmap data and an average AUC of 0.804 on the ATAC-seq data, while the network with the factorization machines layer has an average AUC of 0.855 on the ENCODE+Roadmap data and an average AUC of 0.823 on the ATAC-seq data. The mean ROC-curves have been plotted for both of these networks and can be seen in Figures 5.7 and 5.8. To test whether the difference between AUCs is significant, we calculate the difference in AUC between the networks for each of the cell types. This results in a set of 164 differences with a mean of $(2.03 \pm 0.78) \times 10^{-2}$ (biased towards a higher AUC for the FM network). Applying Wilcoxon signed-rank test [17], we find a p-value of 5.75×10^{-29} for the null hypothesis as opposed to the alternative hypothesis that the mean is larger than zero (in the direction of a larger mean for the FM network). This p-value is very low, meaning that we can reject the null hypothesis.

5.2.3 Attempting to find transcription factor interactions

Using TomTom, we match the convolution filters in the factorization machines network trained on the ECODE+Roadmap data to a database of transcription factor motifs. Of the 300 filters, 53 can be matched to an existing transcription factor motif with a an E-value of at most 0.01. This E-value is the number of times that one expects a random match against the target database. This threshold is chosen to not be very strict, as we would rather have more matches for which we can look at interactions. Of these matches, the first four are displayed in Figure 5.9. Many of the

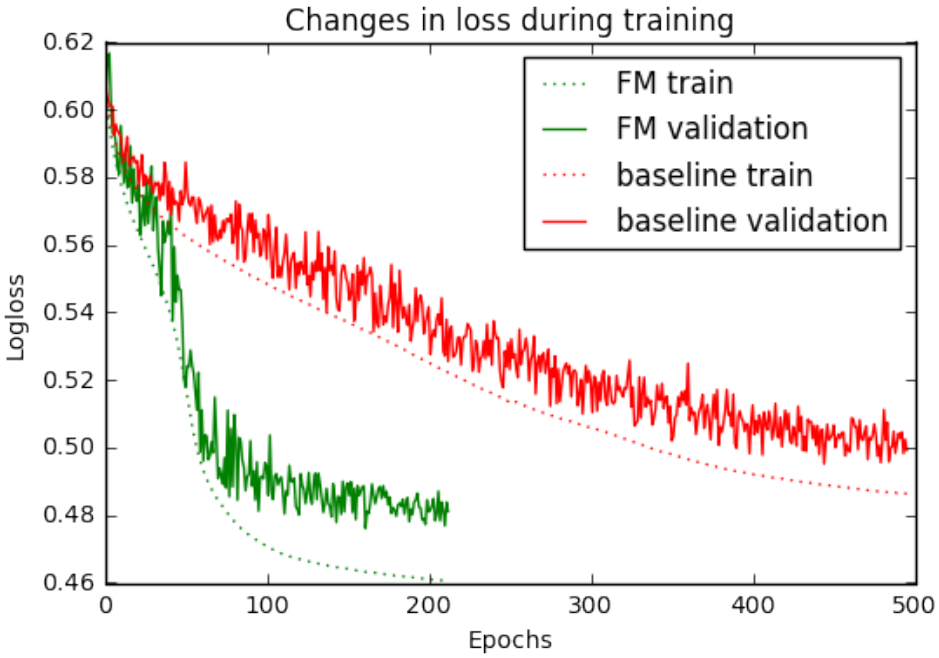

FIGURE 5.6: Training and validation loss for both the baseline and factorization machines networks, during training on the in-house ATAC-Seq data.

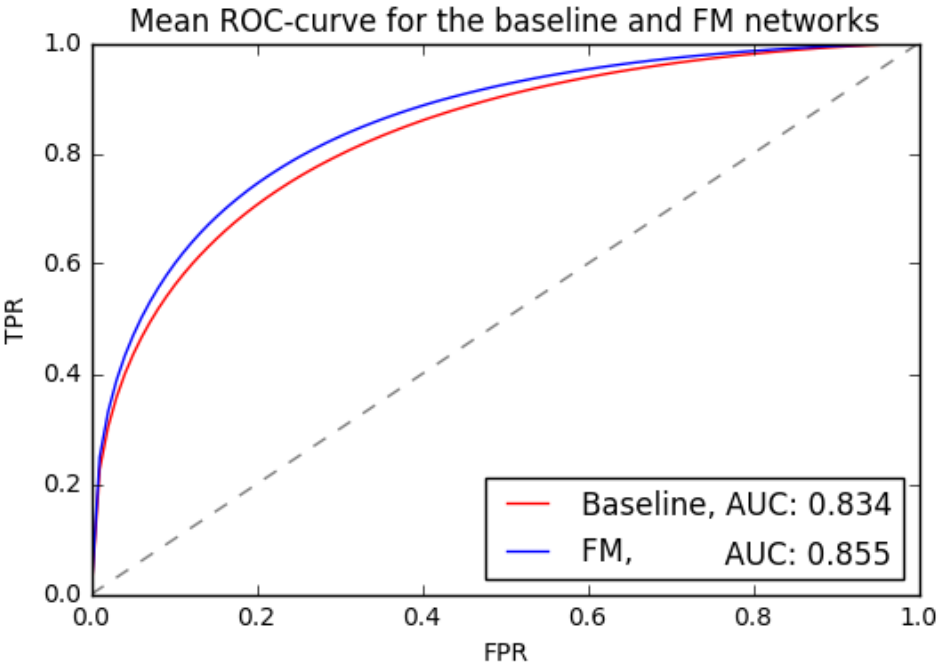

FIGURE 5.7: The mean ROC-curve for both the baseline and factorization machines networks, after being trained on the ENCODE+Roadmap data. The ROC-curve is the mean of the curves of all 164 cell types.

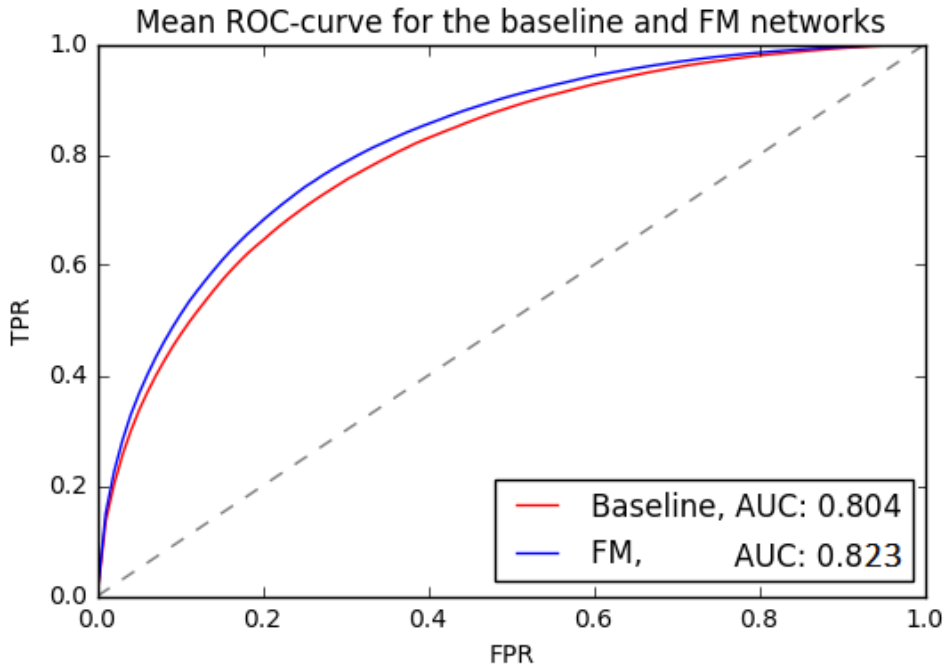

FIGURE 5.8: The mean ROC-curve for both the baseline and factorization machines networks, after being trained on the in-house ATAC-seq data. The ROC-curve is the mean of the curves of all 6 cell types.

matching filters seem to have learned a pattern that alternates either A's and T's or C's and G's. While these do match to a transcription factor motif, it is not hard to imagine that the network could have learned these patterns for another reason than the detection of the matching transcription factors.

In an attempt to find which interactions the network has learned, the interactions with the highest contribution are ranked. However, of the 100 highest ranking interactions, not a single interaction is between two filters that can be matched to a transcription factor. Because not a single interaction top-100 interactions is between filters of which the contents can be interpreted, studying of the learned interactions is not possible.

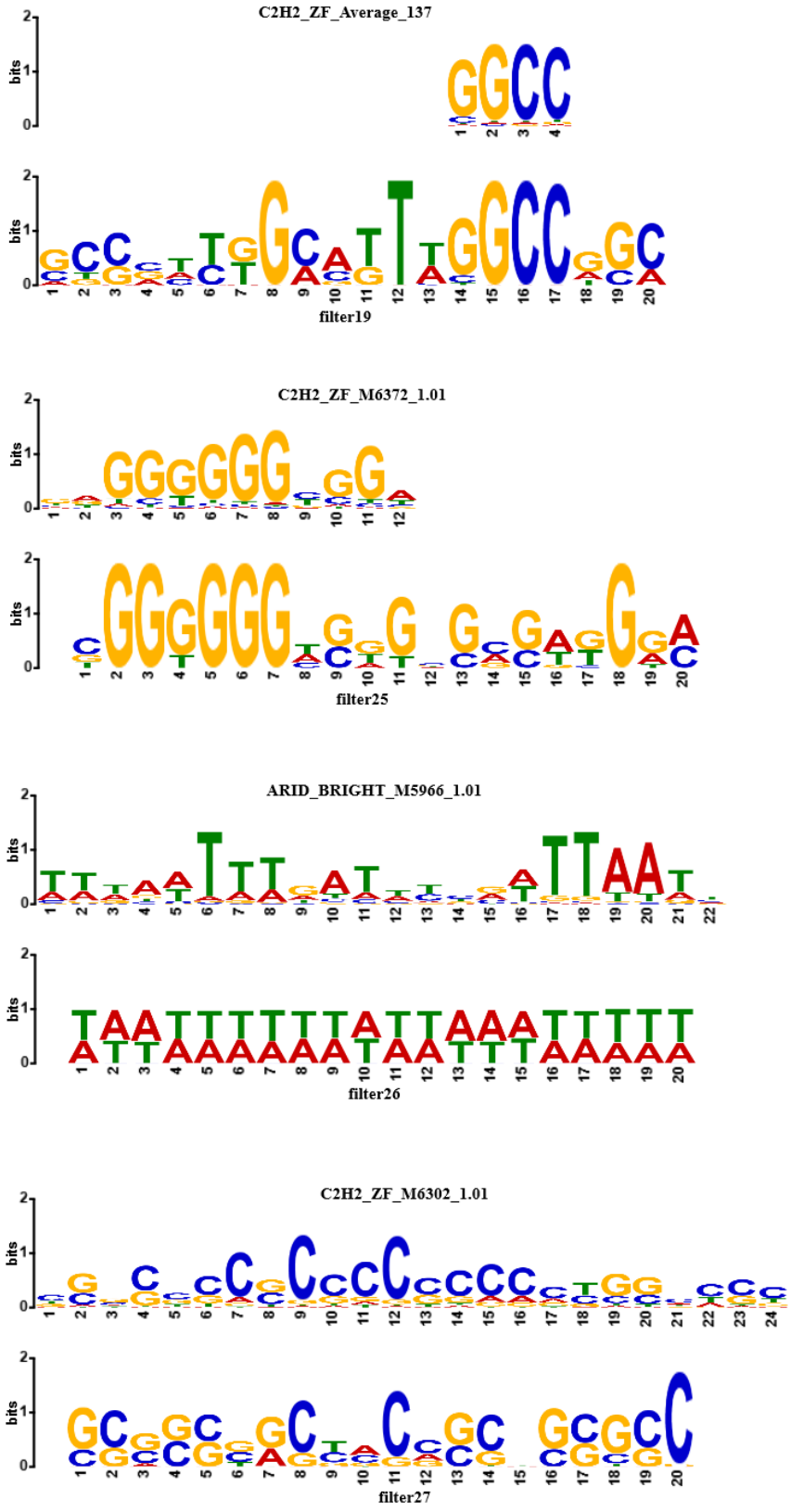

FIGURE 5.9: Four matches of the convolution filters to a transcription factor motif.

5.3 Capturing positional dependencies

Both the baseline network and the motif builder network are trained on a small self-generated dataset, as explained in Section 4.4. On this dataset, the motif building network reached an AUC on the test set of 0.87, whereas the baseline network only reached 0.62. The ROC-curve for both these networks can be seen in Figure 5.11. This shows that the motif building network is better at modeling the different effects of various versions of the same motifs. The shape of the ROC-curve is fairly unexpected. A normal ROC-curve would approximate a TPR of 1.0, but would not reach it until the FPR is also 1.0. This ROC-curve, however, suggests that the network never makes a mistake on the predictions about which it is very sure. Figure 5.12 shows the predictions that the network has made on the test set. There are peaks close to 0 and 1, and a wider distribution between 0.25 and 0.6. The shape of the ROC-curve suggests that the predictions about which the network is very sure (the 0's and 1's) are all correct.

When training the motif building network on the ATAC-seq data, it only reached an AUC of 0.707. This performance is worse than that of the baseline architecture on the same dataset, as can be seen in Section 5.2.2. Figure 5.13 shows the loss during training for the motif building network, as compared to that of the baseline network. It shows that the network is not improving its performance, neither on the training data nor on the validation data.

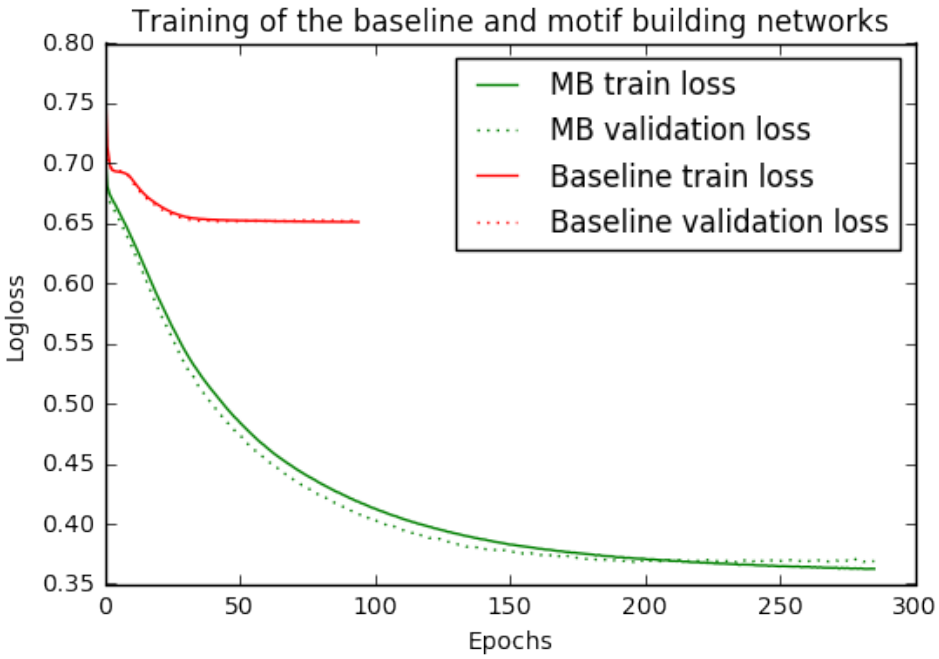

FIGURE 5.10: The loss during training for both the motif building (MB) network and the baseline network. Training is stopped after 50 iterations with no improvement on validation loss.

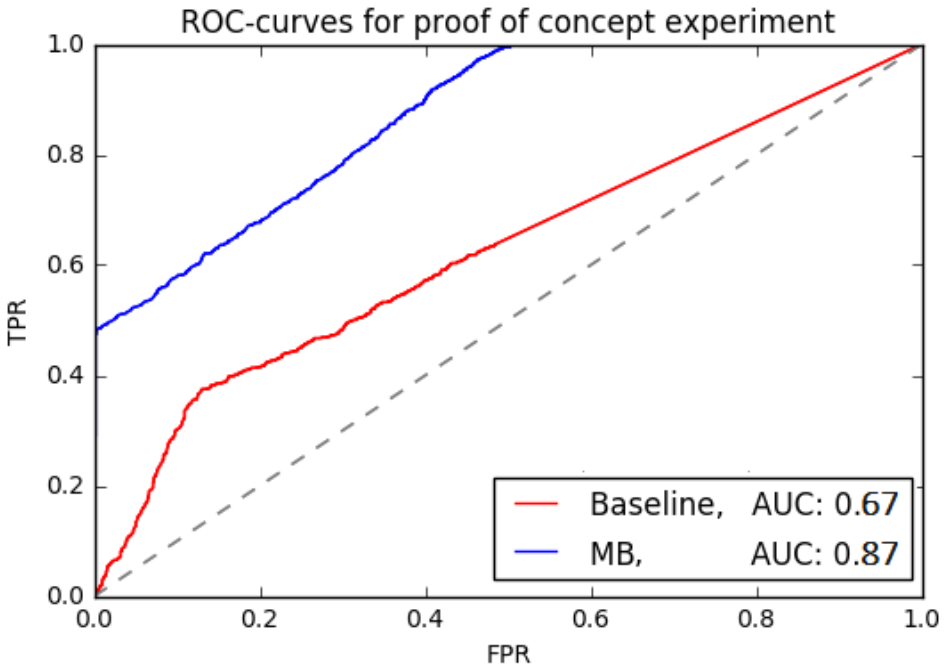

FIGURE 5.11: The ROC-curve for the baseline and motif building networks after being trained on a small set of sequences with positional dependencies.

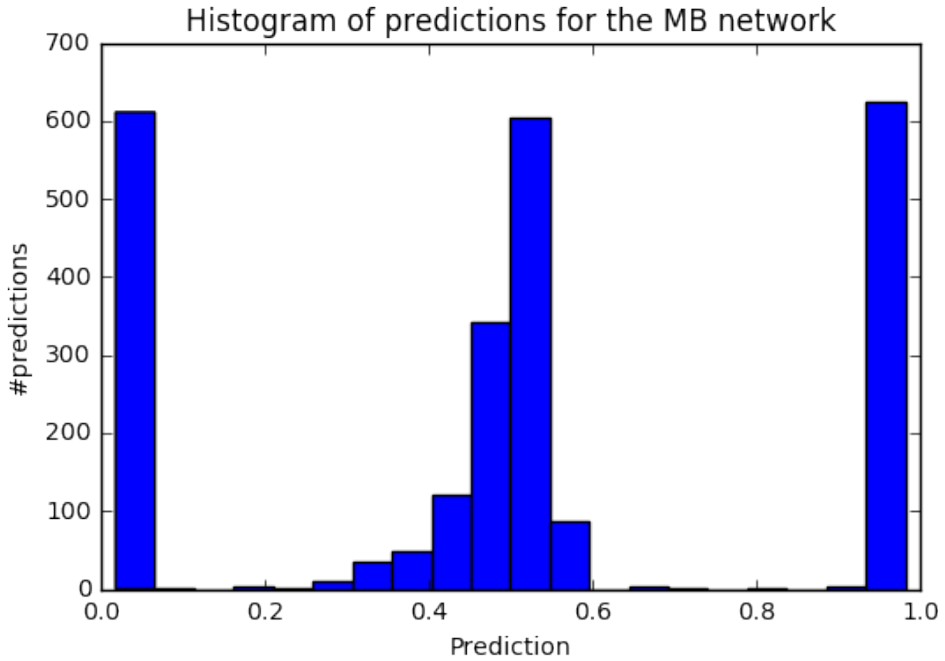

FIGURE 5.12: A histogram of the predictions that the motif building network has made on the test set of the proof-of-concept experiment.

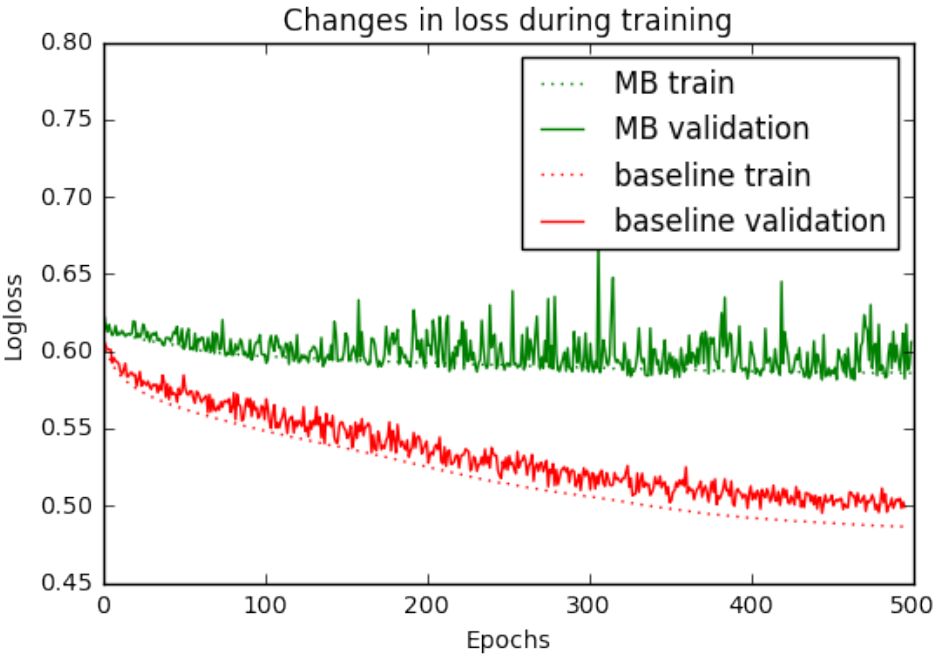

FIGURE 5.13: The loss during training for both the motif building (MB) network and the baseline network, during training on the ATAC-seq dataset.

Chapter 6

Conclusion

6.1 Experiments on the baseline network

The experiments on the baseline network were done to better understand in which way the network represents important transcription factor motifs. Using the procedure explained in Section 3.3, learned motifs were extracted from the trained networks and compared to those motifs that were included in the self-generated data. This way, we learned that even though some convolution filters correspond to a single motif, an even larger part of the filters is used to detect a combination of motifs. By training the baseline network with a varying number of convolution filters, we have studied the effect that this parameter has on the interpretability of the learned filters. We found that having less filters than the number of relevant motifs in the data is bad for interpretability, because this reduces the number of single-motif filters that the network learns. While increasing the number of filters past the number of relevant motifs does lead to more combination filters, the number of single-motif filters stays the same. As such, when an estimated range of relevant transcription factor motifs is given, this experiment seems to suggest that it is better to use a number of filters at the higher end of this range, as underestimation hurts interpretability, whereas overestimation does not.

We hypothesized that the network might group some transcription factors into a single filter, because they have roughly the same contribution to the accessibility. In those cases, the need for the network to be able to differentiate these motifs is small and as such, the network might simply combine them into a single filter. Following through with this reasoning, we trained a 20-filter baseline network on 10 target classes and compared the results to the networks trained on a single target class. This experiment showed that our hypothesis was wrong, as the networks trained on 10 target classes did not learn more single-motif filters than those trained on a single target class. This shows that decorellating transcription factor functions by training to multiple targets does not lead to more interpretable results. This also suggests that one does not require many cell types to interpret a CNN trained on DNase-Seq or ATAC-seq data, as interpretability is not improved when one trains the network to predict accessibility in multiple cell types at the same time.

6.2 Using factorization machines to discover interactions

6.2.1 Experiments on the factorization machines network

We trained the factorization machines network on a set of self-generated sequences for which chromatin accessibility is determined with the paired motifs model. This resulted in networks that were much harder to interpret than the baseline networks. None of the convolution filters learned by the network were single-motif filters,

making matching to the relevant motifs difficult. Assuming the same happens when the factorization machines networks trained on real-world data, this would make matching to actual transcription factor motifs nearly impossible. This was also the reason that we trained the network again, using the weights of the more interpretable filters of previous experiments. While this helped increasing interpretability of the network, its predictive performance decreased.

The factorization machines seemed to find a stable solution, in which the same interactions between filters were deemed important across all folds. Most of the interactions with the highest importance appear to be interactions between paired motifs, showing that we can extract the correct transcription factor interactions from the trained networks. This is the result we were expecting and it shows that the factorization machines can represent non-linear interactions in the data in an intuitive way.

6.2.2 Proof of non-linear interactions

First, we have shown that the factorization machines network are better than the baseline network at predicting chromatin accessibility for sequences where non-linear interactions between transcription factor motifs are relevant. This means that, in real-world data, factorization machines should have a superior performance if these non-linear interactions are indeed relevant.

We then showed that non-linear interactions between transcription factors appear to be relevant in the determination of chromatin accessibility in both the ENCODE+Roadmap data and the in-house ATAC-seq data. We did this by training the baseline network and the factorization machines network, which differ only by the factorization machine's interaction term, on these datasets and comparing their generalization performance. The factorization machines network consistently reaches a higher AUC across cell types than the baseline network, leading to a significantly better generalization performance. This means that the non-linear relations between average filter activations are provide a benefit when predicting chromatin accessibility. Since these filters seem to detect transcription factor motifs, this result suggests that there are some interactions between transcription factors that influence the chromatin state. Even though little is known about transcription factor interactions, the high complexity that nearly all biological processes exhibit would suggest that those interactions are not just linear, which is consistent with the obtained results.

6.2.3 Attempting to find transcription factor interactions

Studying the interactions learned by the factorization machines did not allow us to find transcription factor interactions. The reason for this approach not working, is that only a small part of the convolution filters in the network could be matched against known transcription factor motifs. For most of the filters, no matches could be found, meaning that we do not know which transcription factor motifs these filters are looking for. This means that any interaction that includes these unmatchable filters provides us with no information. Because all important interactions were of this kind, we were unable to discover any interaction between transcription factors.

6.3 Capturing positional dependencies

Through a proof-of-concept experiment, we have shown that the motif building network can indeed perform better than the baseline network on a dataset that includes positional dependencies within motifs. The predictions made by the baseline network were slightly better than random guessing, while the motif building network managed to reach an AUC of 0.87. This showed that this novel architecture can indeed capture positional dependencies more effectively than the baseline network.

After the motif building network was proven to be effective at capturing positional dependencies in self-generated data, it was trained on the ATAC-seq dataset to test its effectiveness on real-world data. However, comparing it to the baseline network trained on the same data revealed that the motif building network did not perform well. This was unexpected, as the network has the capacity to learn the same relations as the baseline network. Apparently, transforming the input to be a sequence of overlapping subsequences has made the problem of predicting chromatin accessibility more difficult to learn, resulting in the network not training properly. In the future, change the training parameters could show whether this network holds promise.

Chapter 7

Discussion

7.1 Usage of self-generated data

One of the large issues with the interpretation of the networks, lies in the fact that there is no ground truth for transcription factor motifs. It is known that transcription factors play a role in chromatin accessibility, and since the network can accurately predict accessibility, it is likely that it can recognize transcription factor binding patterns. We have shown that the patterns extracted from the convolution filters of networks trained on accessibility data can be matched to transcription factor motifs. The list of known motifs, however, is far from complete. If a learned pattern does not match a known motif, very little can be said about it. It might be a new transcription factor motif, but it might also be a noisy combination of motifs or a pattern that is useful to the network for any other reason. The same issue exists with interactions between those transcription factors, not many of which are known. Therefore, none of the relations that one finds from the network can be tested, meaning that they could be interactions that were not known before, or relations that have no biological meaning. We have tried to remedy this issue through the use of self-generated data, which provides a way of testing whether the techniques used come up with the correct results. This approach is far from perfect. Any model used to generate the data is bound to be incorrect in some way, since the exact mechanisms that determine chromatin accessibility are unknown. Therefore, there is no guarantee that techniques that work for the self-generated data will also work on the real-world data. By varying the parameters for generation of the data, we can improve our understanding of the behavior of the networks, thereby gaining confidence that the techniques we use will still find the correct relations when they are applied to real-world data.

There is still a lot of work that can be done using self-generated data to further increase understanding of the networks. Once could think of different models for the background to determine whether some of the filters that cannot be matched are used to filter out the background. Another possibility is to incorporate more complex accessibility models, to end up with a model that closer matches the real-world scenario. These experiments are out of scope for this thesis, since the goal was not just to understand how the networks represent data, but also apply that understanding to the real-world data.

7.2 Difficulties in matching combination filters

In the experiments conducted with self-generated data, a distinction was made between single-motif filters, combination filters and other filters. Matching these combination filters to the motifs they represent was possible for two reasons: accessibility only depends on 20 different motifs and the fact that those motifs used consensus sequences instead of having different weight values for all nucleotides for every position. This made it relatively simple to find the motifs to which a filter could correspond. For the real-world data, however, transcription factor are not the simple consensus sequences. This, combined with the fact that more than 500 of them are known, makes it a much more difficult task to match filters to the corresponding motifs. TomTom is built to match motifs to the known transcription factors, which works fairly well for the single-motif filters. However, TomTom is not well suited for finding combination filters. It interprets the combination filter as a single motif and looks for a matching transcription factor. Since the combination filter will look fairly different from the motifs it is a combination of, chances that single transcription factors match with the filter will be low.

If it were possible to match combination filters to the correct transcription factors, then much more information could be taken from the networks. We would no longer be limited to just looking at interactions between single-motif filters. The experiment in Section 5.1.1 suggests that these combination filters make up a majority of the filters, so being able to including them in analyses could make a huge difference.

One of the things that can be improved upon, is the extraction of motifs from the filters in a network. The current procedure normalizes every position separately, which means that the resulting motif only depends on the ratio between different nucleotides for each position. It completely disregards the differences in size of the weights between positions. When one assumes that the weights across relevant positions of the filter are within the same magnitude, this is not an issue, but there is no guarantee that this is the case. When a position in the filter is non-informative, the network can either make the weights very small or assign the same weight to each nucleotide. In the latter case, any input will lead to the same constant output for that position. If the former case occurs, not all nucleotides might receive an equally small value. While this is irrelevant for the network, since the contribution of any of the nucleotides is insignificant, the motif extraction procedure could be fooled by this small absolute difference between nucleotides. Looking at the experiments with self-generated data, however, shows that the current approach actually works fairly well.

Even though the procedure works fairly well when the network is trained on self-generated data, there is no guarantee that the same is true when training on real-world data. To improve on the method, one would like to take the actual weights across positions into account. One way to do this, would be to smooth out the ratio between nucleotides when weights are low. Since having equal values for every nucleotide in a position is equivalent to having low weights for the whole position, one could combine the two. By adding a constant to each nucleotide value, their relative importances are reduced. By inversely scaling this constant with the weight size, this approach makes the differences between nucleotides in positions with high weights relatively more pronounced, thus making those positions more important in the extracted motif. While an approach based on this principle might work, there are issues. For example, when the fluctuations in weights between relevant positions

are large, this approach would make it harder to match the extracted motif to real transcription factor motifs, because some of the relevant positions are smoothed as well. Due to issues such as these, we have used the simple method described in the paper in favor of one that is more complicated.

7.3 Proof of non-linear interactions

The experiment in which the factorization machines network is shown to outperform the baseline network is set up in such a way, that the only difference between the networks is the added interaction term in the factorization machines. Comparing the performance of both networks shows whether or not it is useful to combine average filter activations in a non-linear way. The reasoning behind this, is that these filter activations solely look at transcription factors, meaning that non-linear interactions between these filters correspond to non-linear interactions between transcription factors. While this reasoning is sound when these convolution filters indeed only look at transcription factors, it breaks down when convolution filters are being used for reasons other than transcription factor detection. It is not hard to imagine a case in which the network uses a filter to detect nucleotide frequency in the background, using this information in a non-linear way to improve prediction. The baseline network cannot learn non-linear relationships such as this background correction, meaning that the difference in performance could be caused by effects other than that of the non-linear interactions between transcription factors. While this experiment still proves that interactions between filter activations provide a benefit, it can not be seen as a proof of interactions between transcription factors.

7.4 The role of positional information

The baseline and factorization machines networks both contain an pooling layer that averages the outputs of a convolution filter. This means that all positional information that these networks can learn lies within these convolution filters. As such, the network can not base its predictions on the locations of the filter activations, meaning that distances between transcription factor binding sites, or the sequence in which they occur are not factors that the network can take into account. Despite these limiting factors, the factorization machines network still managed to reach an average AUC of 0.855 on the combined Encode and Roadmap data. While this is not as good as the network suggested in the Basset paper [12], which reaches an average AUC of 0.895 on the same data, these results do suggest that positional information between transcription factor binding sites might not be of great importance for the chromatin state.

7.5 Use of MaxPooling and AveragePooling

In the motif building architecture, a MaxPooling layer is used as opposed to the AveragePooling layers in previous networks. These layers have a similar function, both reducing the output to a single value for each filter. The advantage of the AveragePooling layer is that there will be a difference in activation when a sequence contains multiple motifs for which the filter has a large activation as opposed to containing a single motif. This is not the case for the MaxPooling layer, for which motif repetition does not lead to a different output. On the other hand, an AveragePooling layer

can result in a sub-optimal filter having the same output signal as a filter that better matches the target motif, because all output values are taken by the pooling layer. By using a MaxPooling layer this effect can be avoided, because only information about the highest activation is passed on by the pooling layer, forcing the network to learn the full motif if it wants to detect a motif. This effect is the reason that a MaxPooling layer is necessary in the motif building architecture, for without it, the network will simply learn the "most optimal" block and repeat it several times, leading to filters that do not represent actual transcription factor motifs. With a MaxPooling layer replacing the AveragePooling, this effect is eliminated.

7.6 Sources of loss

Ideally, one would like to produce a model that perfectly predicts the chromatin accessibility for all new sequences. In practice, this is hard to achieve due to a variety of reasons. Even on the self-generated data set, some issues could prevent us from reaching a perfect prediction. One of these is the occurrence of motifs in the background pattern. In the self-generated data sets, the sequences have a randomly generated background pattern, parts of which are replaced with the transcription factor motifs. Which of these motifs are inserted in the sequences determines whether the sequence is accessible or not. This process, however, does not actually check for the presence of randomly occurring motifs in the background sequence, which could lead to wrong accessibility scores. A rough estimate of this happening: given a motif of width 15 and approx. 500 background positions in which a motif could occur, one expects a motif to randomly occur once every $\frac{4^{15}}{500} \approx 2 \times 10^6$ sequences. As such, with only 100.000 sequences and 20 motifs in our generated data sets, this effect is not something to worry about. However, for small motifs this effect becomes larger. During development of the motif building architecture one of the networks would not train properly. The reason for this was that the relevant motifs were only of length 6, thus greatly increasing the chance that the background sequence randomly contained one of these motifs. Increasing the length of these motifs to 12 solved this problem.

A different issue that could cause incorrect predictions,is that the network simply doesn't have enough power to learn the relations necessary to make a correct prediction. This is the case when we train the baseline network on ENCODE+Roadmap data. Even when the network is properly trained, the restrictions imposed by the single dense layer following the pooling layer simply do not allow it to learn more complex models.

Finally, the data used to train the models might not allow a better prediction. In our case, we use sequences that are 600 nucleotides long and force the network to learn the chromatin state for these sequences. However, we might not be providing the network with all information necessary to make the right prediction. The three-dimensional shape of the chromatin might be such that regions of the DNA that are millions of nucleotides away play an important role, causing the same 600 nucleotide sequence to have a different chromatin state in different regions. To increase the model performance in cases such as these, one needs to enrich the input data with more information that is relevant to the prediction you want to make.

Appendix A

Appendix A

A.1 Experiments on the paired motif model

TABLE A.1: List of motif pairs as used in the paired-motifs model in the experiments in section 4.3.1.

Motif 1	Motif 2
HHEX	IRF4
ZBTB12	NFIB
JUND	SCRT1
ELK1	FOXL1
HOXA7	NKX2-2
TFAP2E	HOXD8
ZHX1	BSX
MAX	MAFG
JUNB	GATA4
ZNF282	BHLHE41

TABLE A.2: Inner product between the interaction vectors learned by the factorization machines network, as trained in Section 4.3.1. This network was trained on fold 1 out of 10, resulting in a total of 7 correctly matching filters.

Weights	Filter 1	Filter 2
23970	HOXD8	TFAP2E
10052	BSX	ZHX1
5367	HHEX BSX	IRF4 MAFG NKX2-2
4638	TFAP2E NFIB	ZBTB12
4569	HOXA7 MAFG	FOXL1 NKX2-2
3673	MAFG	MAX NKX2-2
3613	TFAP2E ZHX1	ZHX1
3457	TFAP2E ZHX1	TFAP2E
2893	HOXD8	TFAP2E ZHX1
2780	HHEX BSX	HHEX

TABLE A.3: Inner product between the interaction vectors learned by the factorization machines network, as trained in Section 4.3.1. This network was trained on fold 2 out of 10, resulting in a total of 7 correctly matching filters.

Weights	**Filter 1**	**Filter 2**
22696	HOXD8	TFAP2E
5926	BSX	ZHX1
5058	TFAP2E NFIB	ZBTB12
4978	HHEX BSX	IRF4 MAFG NKX2-2
4258	HOXA7 MAFG	FOXL1 NKX2-2
3558	MAFG	MAX NKX2-2
3531	TFAP2E ZHX1	ZHX1
3327	TFAP2E ZHX1	TFAP2E
2911	HOXD8	TFAP2E ZHX1
2646	HOXD8	HOXD8 BHLHE41 ELK1 MAFG

TABLE A.4: Inner product between the interaction vectors learned by the factorization machines network, as trained in Section 4.3.1. This network was trained on fold 3 out of 10, resulting in a total of 8 correctly matching filters.

Weights	**Filter 1**	**Filter 2**
30942	HOXD8	TFAP2E
4978	TFAP2E NFIB	ZBTB12
4972	HHEX BSX	IRF4 MAFG NKX2-2
4740	BSX	ZHX1
4689	HOXA7 MAFG	FOXL1 NKX2-2
3461	TFAP2E ZHX1	ZHX1
3358	TFAP2E ZHX1	TFAP2E
3217	MAFG	MAX NKX2-2
2819	HOXD8	TFAP2E ZHX1
2576	HOXD8	HOXD8 BHLHE41 ELK1 MAFG

TABLE A.5: Inner product between the interaction vectors learned by the factorization machines network, as trained in Section 4.3.1. This network was trained on fold 4 out of 10, resulting in a total of 8 correctly matching filters.

Weights	**Filter 1**	**Filter 2**
37214	HOXD8	TFAP2E
6228	BSX	ZHX1
5383	TFAP2E NFIB	ZBTB12
4863	HHEX BSX	IRF4 MAFG NKX2-2
4514	HOXA7 MAFG	FOXL1 NKX2-2
3557	TFAP2E ZHX1	ZHX1
3403	TFAP2E ZHX1	TFAP2E
3245	MAFG	MAX NKX2-2
2725	HOXD8	TFAP2E ZHX1
2710	IRF4 MAFG NKX2-2	HHEX

TABLE A.6: Inner product between the interaction vectors learned by the factorization machines network, as trained in Section 4.3.1. This network was trained on fold 5 out of 10, resulting in a total of 8 correctly matching filters.

Weights	**Filter 1**	**Filter 2**
15479	HOXD8	TFAP2E
5325	BSX	ZHX1
4857	TFAP2E NFIB	ZBTB12
4588	HHEX BSX	IRF4 MAFG NKX2-2
4328	HOXA7 MAFG	FOXL1 NKX2-2
3527	TFAP2E ZHX1	ZHX1
3309	TFAP2E ZHX1	TFAP2E
3208	MAFG	MAX NKX2-2
2770	IRF4 MAFG NKX2-2	HHEX
2698	HOXD8	TFAP2E ZHX1

TABLE A.7: Inner product between the interaction vectors learned by the factorization machines network, as trained in Section 4.3.1. This network was trained on fold 6 out of 10, resulting in a total of 7 correctly matching filters.

Weights	**Filter 1**	**Filter 2**
15730	HOXD8	TFAP2E
5863	BSX	ZHX1
4861	TFAP2E NFIB	ZBTB12
4766	HOXA7 MAFG	FOXL1 NKX2-2
4138	HHEX BSX	IRF4 MAFG NKX2-2
3754	MAFG	MAX NKX2-2
3550	TFAP2E ZHX1	ZHX1
3366	TFAP2E ZHX1	TFAP2E
2798	HOXD8	TFAP2E ZHX1
2705	HOXD8	HOXD8 BHLHE41 ELK1 MAFG

TABLE A.8: Inner product between the interaction vectors learned by the factorization machines network, as trained in Section 4.3.1. This network was trained on fold 7 out of 10, resulting in a total of 8 correctly matching filters.

Weights	**Filter 1**	**Filter 2**
40428	HOXD8	TFAP2E
5051	TFAP2E NFIB	ZBTB12
4796	HHEX BSX	IRF4 MAFG NKX2-2
4677	BSX	ZHX1
4276	HOXA7 MAFG	FOXL1 NKX2-2
3630	TFAP2E ZHX1	ZHX1
3431	MAFG	MAX NKX2-2
3409	TFAP2E ZHX1	TFAP2E
2901	HOXD8	TFAP2E ZHX1
2620	IRF4 MAFG NKX2-2	HHEX

TABLE A.9: Inner product between the interaction vectors learned by the factorization machines network, as trained in Section 4.3.1. This network was trained on fold 8 out of 10, resulting in a total of 7 correctly matching filters.

Weights	**Filter 1**	**Filter 2**
23441	HOXD8	TFAP2E
5765	BSX	ZHX1
4866	TFAP2E NFIB	ZBTB12
4752	HHEX BSX	IRF4 MAFG NKX2-2
4567	HOXA7 MAFG	FOXL1 NKX2-2
3517	TFAP2E ZHX1	ZHX1
3313	TFAP2E ZHX1	TFAP2E
3201	MAFG	MAX NKX2-2
2963	HOXD8	TFAP2E ZHX1
2570	HOXD8	HOXD8 BHLHE41 ELK1 MAFG

TABLE A.10: Inner product between the interaction vectors learned by the factorization machines network, as trained in Section 4.3.1. This network was trained on fold 9 out of 10, resulting in a total of 7 correctly matching filters.

Weights	**Filter 1**	**Filter 2**
22936	HOXD8	TFAP2E
5458	HHEX BSX	IRF4 MAFG NKX2-2
4824	TFAP2E NFIB	ZBTB12
4590	BSX	ZHX1
4571	HOXA7 MAFG	FOXL1 NKX2-2
3413	TFAP2E ZHX1	ZHX1
3314	TFAP2E ZHX1	TFAP2E
3236	MAFG	MAX NKX2-2
2752	HOXD8	HOXD8 BHLHE41 ELK1 MAFG
2732	HOXD8	TFAP2E ZHX1

TABLE A.11: Inner product between the interaction vectors learned by the factorization machines network, as trained in Section 4.3.1. This network was trained on fold 10 out of 10, resulting in a total of 8 correctly matching filters.

Weights	**Filter 1**	**Filter 2**
33950	HOXD8	TFAP2E
5374	HOXA7 MAFG	FOXL1 NKX2-2
4862	TFAP2E NFIB	ZBTB12
4815	BSX	ZHX1
4179	HHEX BSX	IRF4 MAFG NKX2-2
3534	TFAP2E ZHX1	ZHX1
3447	TFAP2E ZHX1	TFAP2E
3396	MAFG	MAX NKX2-2
3003	IRF4 MAFG NKX2-2	HHEX
2976	HOXD8	TFAP2E ZHX1

Bibliography

[1] K. Takahashi and S. Yamanaka, "Induction of pluripotent stem cells from mouse embryonic and adult fibroblast cultures by defined factors", *Cell*, vol. 126, no. 4, pp. 663–676, 2006.

[2] D. Santoni, F. Castiglione, and P. Paci, "Identifying correlations between chromosomal proximity of genes and distance of their products in protein-protein interaction networks of yeast", *PloS one*, vol. 8, no. 3, e57707, 2013.

[3] E. P. Consortium *et al.*, "The encode (encyclopedia of dna elements) project", *Science*, vol. 306, no. 5696, pp. 636–640, 2004.

[4] B. E. Bernstein, J. A. Stamatoyannopoulos, J. F. Costello, B. Ren, A. Milosavljevic, A. Meissner, M. Kellis, M. A. Marra, A. L. Beaudet, J. R. Ecker, *et al.*, "The nih roadmap epigenomics mapping consortium", *Nature biotechnology*, vol. 28, no. 10, pp. 1045–1048, 2010.

[5] J. D. Buenrostro, P. G. Giresi, L. C. Zaba, H. Y. Chang, and W. J. Greenleaf, "Transposition of native chromatin for fast and sensitive epigenomic profiling of open chromatin, dna-binding proteins and nucleosome position", *Nature methods*, vol. 10, no. 12, pp. 1213–1218, 2013.

[6] C. A. Meyer and X. S. Liu, "Identifying and mitigating bias in next-generation sequencing methods for chromatin biology", *Nature Reviews Genetics*, vol. 15, no. 11, pp. 709–721, 2014.

[7] G. Hinton, "Overview of mini-batch gradient descent", [Online]. Available: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks", in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification", in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[10] C. N. Dos Santos and M. Gatti, "Deep convolutional neural networks for sentiment analysis of short texts.", in *COLING*, 2014, pp. 69–78.

[11] S. Rendle, "Factorization machines", in *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, IEEE, 2010, pp. 995–1000.

[12] D. R. Kelley, J. Snoek, and J. L. Rinn, "Basset: Learning the regulatory code of the accessible genome with deep convolutional neural networks", *Genome research*, vol. 26, no. 7, pp. 990–999, 2016.

[13] S. Gupta, J. A. Stamatoyannopoulos, T. L. Bailey, and W. S. Noble, "Quantifying similarity between motifs", *Genome biology*, vol. 8, no. 2, R24, 2007.

[14] S. Omidi and E. van Nimwegen, "Automated incorporation of pairwise dependency in transcription factor binding site prediction using dinucleotide weight tensors", *BioRxiv*, 2016. DOI: 10.1101/078212. [Online]. Available: http://www.biorxiv.org/content/early/2016/09/28/078212.

[15] M. T. Weirauch, A. Yang, M. Albu, A. G. Cote, A. Montenegro-Montero, P. Drewe, H. S. Najafabadi, S. A. Lambert, I. Mann, K. Cook, *et al.*, "Determination and inference of eukaryotic transcription factor sequence specificity", *Cell*, vol. 158, no. 6, pp. 1431–1443, 2014.

[16] B. L. Welch, "The generalization of student's' problem when several different population variances are involved", *Biometrika*, vol. 34, no. 1/2, pp. 28–35, 1947.

[17] F. Wilcoxon, "Individual comparisons by ranking methods", *Biometrics bulletin*, vol. 1, no. 6, pp. 80–83, 1945.