Fakultät für Informatik
Technische Universität München

Understanding the genome via predictive models

Žiga Avsec

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:
Prof. Dr. Björn Menze

Prüfende der Dissertation:
1. Prof. Dr. Julien Gagneur
2. Prof. Dr. Dr. Fabian Theis

Die Dissertation wurde am 11.06.2019 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 22.10.2019 angenommen.

Acknowledgments

First and foremost, I would like to thank Julien Gagneur for being a tremendous mentor in all aspects. You established a great lab with an amazing culture and an ambitions vision. You gave me the freedom to work on things I was most excited about, supported me with great advice, as well as challenged me when necessary. I learned unbelievable many things from you. It was a real joy to work with you.

Second, I would like to thank all my other mentors from whom I also learned a lot:

- Anshul Kundaje for co-supervising the Kipoi and BPNet projects, hosting me in his lab over the spring in 2018, giving excellent guidance on radically new methods, supporting me, and showing me that science is much more than publishing papers.
- Julia Zeitlinger for co-supervising the BPNet project, infecting me with scientific curiosity through many fruitful discussions, and supporting me when I most needed it.

Third, I would like to thank all the people who worked closely with me during my PhD or who have helped otherwise:

- Gagneur lab members (past and present) for creating and maintaining such an amazing lab atmosphere. Especially, I would like to thank Jun for deep scientific discussions, Chriss for setting up stable lab infrastructure, Leo for constantly demonstrating the unexplainable will and strength, Vicente for fun music at lab parties, Daniel for establishing an open peer culture, Juri for explaining how the world works, Hasan for VariantSeqExtractor, Ines for productive office environment, Basak for useful input on my projects, and Bene for setting a high bar in the lab. Additionally, I would like to thank all the students I supervised including Agne, Akshat, Amin, Daniela, Julian, Florian, Goektug, Madalina, Nika, Shalvi, and Yen.
- Kundaje lab members for a great visit of the lab especially Avanti Shrikumar for amazing work including TF-MoDISco, Amr Alexandari for the help with the BPNet project, and Johnny Israeli for countless discussions.
- Oliver Stegle for great co-supervision of the Kipoi project.
- Stegle lab members especially Roman Kreuzhuber for pushing the Kipoi project forward and Thorsten Beier for making the codebase of Kipoi more robust.

- Gökçen Eraslan for hard work on the NRG paper and many enlightening discussions.
- Zeitlinger lab members (Sabrina, Khyati, Robin, George) especially Melanie Weiltert for great help on the BPNet project.
- TAC committee–Fabian Theis and Veit Hornung–for their valuable input.
- Prokish lab and Söding lab members for fun times outside the lab.
- Quantitative Biosciences Munich (QBM) graduate school for their financial support, all the organized events, and excellent courses.

Finally, I would like to thank my family for their unconditional support and Brigita, the love of my life, for keeping my life balanced and happy.

Abstract

Since the first complete assembly of the human genome in 2003, the technologies probing the sequence of the genome and its function have been improving at an astonishing rate. Individual genomes can now be sequenced at low cost. Additionally, many molecular processes or phenotypes such as transcription factor binding, DNA accessibility and gene expression can be measured genome-wide at high accuracy. Additionally, powerful modeling techniques such as deep neural networks have recently emerged. This opens a path to understand the human genome by relating DNA sequence to molecular phenotypes using predictive models. However, these new modeling techniques require significant adaptation to genomics problems to leverage their full potential, both in terms of predictive accuracy as well as interpretation. Additionally, there is a lack of predictive model exchange in the community despite their importance.

This thesis addresses these limitations in four ways. First, I develop a new neural network layer—spline transformation—to flexibly and robustly model distances to various genomics landmarks in neural networks. By modeling distances to genomic landmarks with spline transformation, I significantly increase the accuracy of predicting in vivo RNA-binding protein binding sites and splice branchpoint locations. Spline transformation can be applied to model quantities beyond distances and can be easily integrated into existing models implemented in Keras. This makes it a versatile component in the deep learning toolbox.

Second, I develop BPNet, a sequence-to-profile neural network predicting the read coverage profile of genome-wide assays at base-pair resolution from DNA sequence. I apply BPNet to ChIP-nexus coverage profiles which highlight the sites in the genome bound by transcription factors (TFs) at base-pair resolution. BPNet accurately predicts the ChIP-nexus coverage profiles, on par with replicate experiments. I observe similar results when applying BPNet to coverage profiles measured by ChIP-seq. By modeling the profile, BPNet learns better representations of TF binding compared to the binary classification models.

Third, I develop tools to extract knowledge about TF binding from BPNet. These interpretation tools extract motifs, map them back to the genome, and study their cooperativity. This approach discovers short motifs (5-15 bp), both directly or indirectly bound by TFs, and long motifs (>30 bp) corresponding to transposable elements bound by TFs. Our approach determines motif instances in the genome at a much lower false-positive rate than alternative approaches. Thanks to the much lower false-positive rate, I observe a strong preference for Nanog to bind in a ∼10 bp periodic pattern. By analyzing BPNet predictions of sequences containing specific combinations of motifs, I uncover rules of cooperative binding capturing both short-range protein-protein interactions and long-range pioneering activity of specific TFs. Altogether, the presented approach opens

the path for the systematic discovery of cis-regulatory code in experimentally accessible cell types.

Fourth, I developed Kipoi, a platform to exchange predictive models in genomics. Despite the importance of machine learning models in genomics, the lack of standards and limited centralized access to trained models have hampered their practical impact. Kipoi provides the definition of the trained model, a central repository containing over 2,000 trained models, and an API to load and use these models. By providing a unified framework to archive, share, access, use, and extend models developed by the community, Kipoi fosters the dissemination and use of machine learning models in genomics.

Altogether, I foresee the computational methods and tools developed in this thesis as a catalyst in the endeavour to understand the human genome.

Publications

The Kipoi repository accelerates community exchange and reuse of predictive models for genomics

Žiga Avsec[*1], Roman Kreuzhuber*, Johnny Israeli, Nancy Xu, Jun Cheng, Avanti Shrikumar, Abhimanyu Banerje, Daniel S Kim, Thorsten Beier, Lara Urban, Anshul Kundaje[1], Oliver Stegle[1], Julien Gagneur[1]
* co-first, [1] corresponding author
(2019) **Nature Biotechnology**, DOI: 10.1038/s41587-019-0140-0, Ref: [1]

Author contribution ŽA, RK, JI, AS, AK, OS and JG conceived the Kipoi API. ŽA, RK and TB implemented the Kipoi API. ŽA and RK conceived and implemented kipoi_veff. ŽA, RK and AS conceived and implemented kipoi-interpret. ŽA, RK and JC conceived and implemented kipoiseq. ŽA, RK, JI, NX and AB performed the analysis. DSK compiled the DNA accessibility dataset. ŽA, RK, JI, NX, AS and LU contributed models to the repository. AK, OS and JG designed and supervised research. ŽA, RK, AK, OS and JG wrote the manuscript.

Modeling positional effects of regulatory sequences with spline transformations increases prediction accuracy of deep neural networks

Žiga Avsec[1], Mohammadamin Barekatain, Jun Cheng, Julien Gagneur[1]
[1] corresponding author
(2017) **Bioinformatics**, DOI: 10.1093/bioinformatics/btx727, Ref: [2]

Author contribution ŽA and JG conceived spline transformation. ŽA implemented spline transformation. ŽA performed the analysis with help from MB and JC. JG supervised research. ŽA and JG wrote the manuscript.

Deep learning: new computational modelling techniques for genomics

Gökcen Eraslan*, **Žiga Avsec***, Julien Gagneur, Fabian J. Theis
* co-first
(2019) **Nature Review Genetics**, DOI: 10.1038/s41576-019-0122-6, Ref: [3]

Author contribution The authors contributed equally to all aspects of the article.

MMSplice: modular modeling improves the predictions of genetic variant effects on splicing

Jun Cheng, Thi Yen Duong Nguyen, Kamil J. Cygan, Muhammed Hasan Çelik, William G. Fairbrother, **Žiga Avsec**, Julien Gagneur
(2019) **Genome Biology**, DOI: 10.1186/s13059-019-1653-z, Ref: [4]

Author contribution JC and JG designed the model, with the help of ZA. JC implemented the software and analysed data. TYDN and ZA contributed to developing the modules. JC and JG wrote the manuscript, with the help of ZA, KJC, and WGF. KJC and WGF generated the MaPSy data. MHC wrote the VEP plugin. All authors read and approved the final manuscript.

OUTRIDER: A statistical method for detecting aberrantly expressed genes in RNA sequencing data

Felix Brechtmann*, Agne Matuseviciute*, Christian Mertes*, Vicente A Yepez, **Žiga Avsec**, Maximilian Herzog, Daniel Magnus Bader, Holger Prokisch, Julien Gagneur
* co-first
(2018) **American Journal of Human Genetics** , DOI: 10.1016/j.ajhg.2018.10.025, Ref: [5]

Author contribution JG conceived the project and overviewed the research with the help of ŽA, HP, and VAY. FB, AM, CM analyzed the data. FB, CM and AM developed the software. DMB and MH contributed to the software development and early stage data analysis. JG and ŽA devised the statistical analysis. FB, VAY, CM, and JG made the figures. FB, CM, AM, VAY and JG wrote the manuscript. All authors performed critical revision of the manuscript.

TF-MoDISco v0.4.2.2-alpha: Technical Note

Avanti Shrikumar, Katherine Tian, Anna Shcherbina, **Žiga Avsec**, Abhimanyu Banerjee, Mahfuza Sharmin, Surag Nair, Anshul Kundaje
(2018) **arXiv**, arXiv ID: 1811.00416, Ref: [6]

Author contribution Avanti Shrikumar conceived of and was the primarily developer of TF-MoDISco. Katherine Tian, Anna Shcherbina, Žiga Avsec and Surag Nair contributed features to the TF-MoDISco codebase. Anna Shcherbina proposed the idea of fast affinity matrix computation using kmers. Abhimanyu Banerjee, Mahfuza Sharmin and Žiga Avsec applied versions of TF-MoDISco immediately leading up to v0.4.2.2-alpha on biological data and identified areas of improvement. Anshul Kundaje provided guidance and feedback. Avanti Shrikumar wrote this technical note.

CAGI5 splicing challenge: Improved exon skipping and intron retention predictions with MMSplice

Jun Cheng, Muhammed Hasan Çelik, Yen Duong Nguyen, **Žiga Avsec**, Julien Gagneur (2019) **Human Mutation**, DOI: 10.1002/humu.23788, Ref: [7]

Cis-regulatory elements explain most of the mRNA stability variation across genes in yeast

Jun Cheng, Kerstin C. Maier, **Žiga Avsec**, Petra Rus, Julien Gagneur (2017) **RNA**, DOI: 10.1261/rna.062224.117, Ref: [8]

Mutations in MDH2, Encoding a Krebs Cycle Enzyme, Cause Early-Onset Severe Encephalopathy

Samira Ait-El-Mkadem, Manal Quere, Mirjana Gusic, Annabelle Chaussenot, ..., **Žiga Avsec**, Christian Mertes, ..., Julien Gagneur, ..., Véronique Paquis-Flucklinger (2017) **The American Journal of Human Genetics**, DOI: 10.1016/j.ajhg.2016.11.014, Ref: [9]

Contents

1 Introduction

1.1 Genome and the central dogma of biology

The information encoded in the genome includes instructions for the whole development of the organism starting from a single stem cell. It encodes the information about the variability between species, and it encodes the information about various genetic diseases. It is a fundamental source of information about each individual. Deciphering the genome would bring enormous benefit to the humanity both in terms of diagnosing diseases faster and more accurately, preventing diseases, as well as developing (personalized) drugs [10].

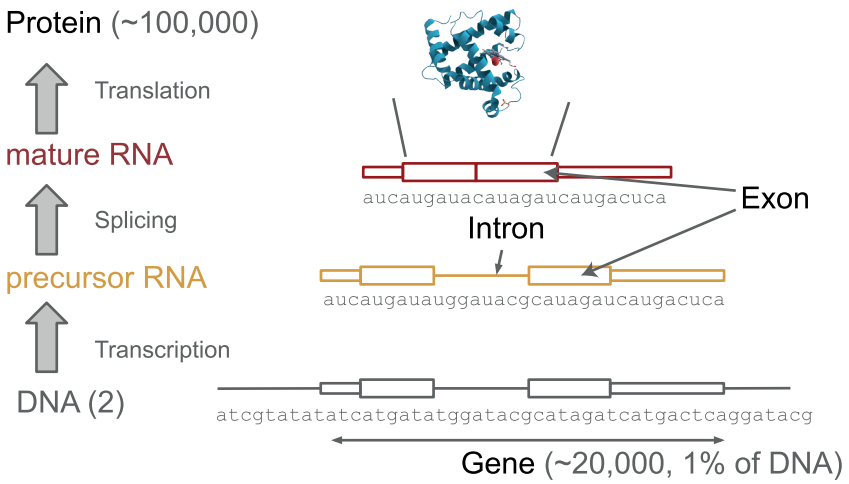


Figure 1.1: The central dogma of biology. Courtesy of Julien Gagneur.

Genome is a set of DNA molecules densely packed in the cell. The human genome is approximately 3 billion base-pairs long. The main building blocks of the cell—RNA and protein molecules—are produced from the genome through the processes called gene expression which occurs in several steps (also called the central dogma of biology, Figure 1.1). The sequence information about the produced RNA or proteins is encoded in genes, which are intervals in the genome. Here we describe gene expression in human cells. Genes cover roughly 1% of the human genome and are typically 26 kb long (median) for protein coding genes [11]. To produce RNAs or proteins, genes are first transcribed by the process termed RNA transcription into precursor RNA. The precursor RNA gets processed by, among other things, removing large parts (∼90%, [11]) of the RNA called introns and splicing (i.e. joining) the remaining parts called exons yielding the mature RNA. Some of these mature RNAs, called messenger RNAs, encode proteins.

The messenger RNA is translated into protein's amino-acid chain which is then folded into the 3D structure yielding the functional protein.

This thesis focuses on understanding how gene regulation, which refers to mechanisms that enhance or repress the expression of a gene, is encoded in the genome. We will first provide some biological background on transcription and then focus on transcription factor binding more in detail. Next, we will describe the computational methods with emphasis on predictive models that model transcription factor binding. These provide the foundations for predictive models modeling other steps of gene expression. We will explain how predictive models can be used to interpret genetic variants and to understand more complex biological processes. Finally, we will conclude the chapter by describing the aims and goals of this thesis.

1.2 Transcription

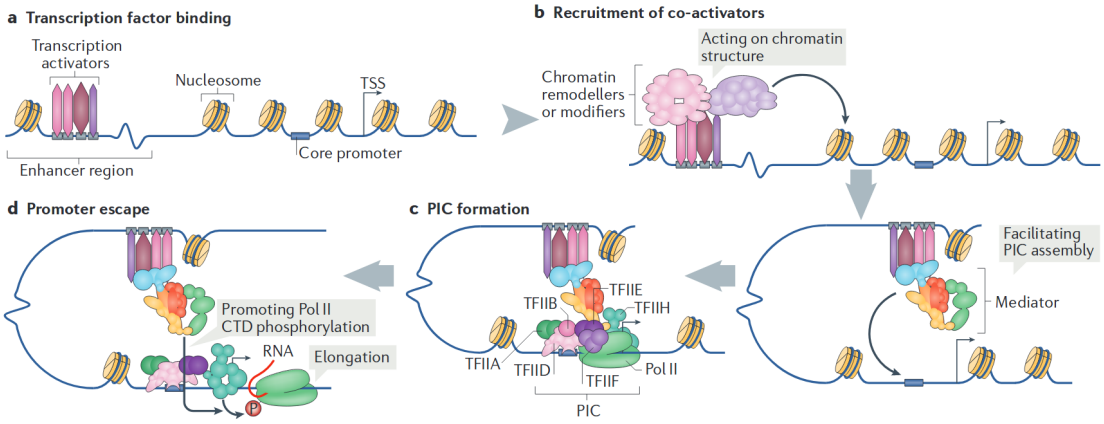


Figure 1.2: Transcription. Transcription is a multi-step process involving many proteins and protein complexes. **a)** In the first step of transcription, multiple transcription factors, called transcription activators, bind to enhancer regions. Enhancer regions are located between 100 bp to few mega bases away from the transcription start site (TSS) of a gene (denoted by an arrow) and the core promoter region (denoted as a blue box). **b)** Activators recruit other complexes such as chromatin modifiers or remodellers which make the enhancer region more accessible for other activator TFs or TF complexes including the Mediator co-activator complex. These activators then facilitate the assembly of the preinitiation complex (PIC). Despite the long linear distance on the DNA between the enhancer and the promoter, the enhancer is actually in close proximity with the promoter in the 3D space. **c)** The PIC is assembled at the core promoter and consists of many TFs and other complexes including Poll II. **d)** Pol II starts transcribing the gene and thereby producing the RNA after the carboxyl-terminal domain (CTD) of a specific Pol II subunit gets phosphorylated. Taken from [12].

The first step of protein production, transcription, is initiated by proteins called transcription factors (TFs) (Figure 1.2A). These TFs recognize short subsequences in the

genome and bind specifically to the DNA. After the preinitiation complex (PIC) is assembled at genomic regions bound by TFs (Figure 1.2B), the RNA polymerase II (Pol II) can start transcribing the gene (Figure 1.2D) into an RNA molecule. Transcription can not be initiated if correct TFs are not bound to the corresponding enhancer and promoter regions. Hence, the TF binding controls whether and how often the transcription of a certain gene will occur and thereby controls the abundance, also called expression, of a gene in the cell.

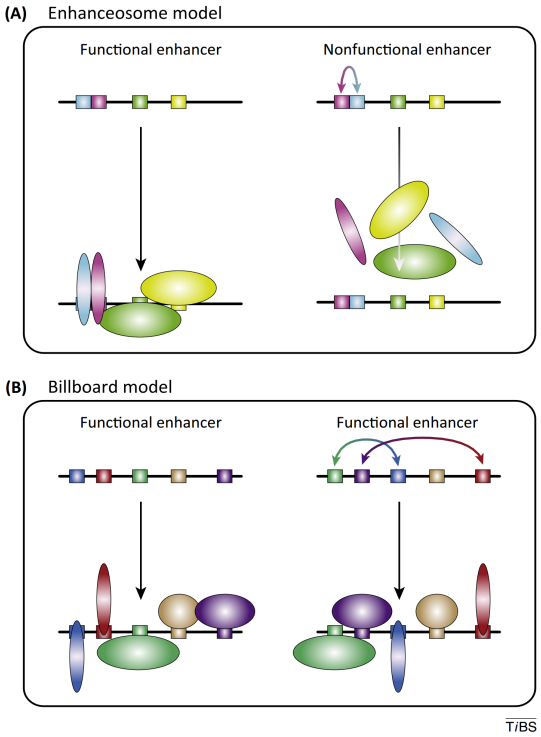


Figure 1.3: Models of TF assembly on enhancer DNA. (A) Left: The enhanceosome model is characterized by cooperative TF binding and highly constrained binding-site positioning. Right: Minor changes in enhancer sequence (i.e. inversion in this case, but insertions, deletions, mutations, etc., also apply) can lead to collapse of TF assembly and enhancer function. **(B)** Left: The billboard model is characterized by highly flexible binding-site grammars. Although all TFs are important for enhancer function, TF binding and enhancer function are not affected by significant changes in binding-site positioning or orientation. Taken from [13].

The specific combination of transcription factors and their spatial organization on the enhancer, also referred to as TF grammar, plays an important role in transcriptional activation. There are two different models of how spatial organization controls enhancer activation: the enhanceosome and the billboard model (Figure 1.3, [13]). The enhanceosome model requires strict spatial organization of different TFs for successful enhancer activation (Figure 1.3A). By contrast, the billboard model only requires a sufficient amount of a certain TF at the enhancer region regardless of the spatial organization

(Figure 1.3B). Due to the insufficient knowledge of where TFs bind in the genome and how they influence the binding of each other, understanding which and to what extent enhancers adhere to one of these two models is still an unsolved problem [14, 15, 16, 13]. Taken together, understanding TF binding is critical to understand where and how much the genes are expressed.

1.3 Transcription factor binding

1.3.1 Biological background

TFs bind to specific positions in the genome (Figure 1.4A) through physical interactions such as hydrogen bonds and hydrophobic contacts between the amino acid side chains of the TF and proximal DNA base-pairs [13]. For a particular TF, the affinity of these interactions is influenced by i) the nucleotide sequence of the binding site [17, 18, 19, 20, 21, 22, 23] which is typically 8-20 bp long [24], ii) the DNA shape of the binding site [25, 26, 27, 28], iii) binding of nearby co-factors [29, 30, 25, 31, 32, 33, 34, 35], iv) chromatin structure such as accessibility or nucleosome occupancy [36, 37, 38, 39, 40, 41, 42, 43], and v) DNA methylation [44] (Figure 1.4).

1.3.2 Measuring TF binding in vivo and in vitro

To determine the binding affinity of a TF for a particular DNA sequence and the locations of DNA bound by TFs in the genome, various *in vitro* [45, 46, 47, 48, 29, 49, 50] and *in vivo* [51, 52, 53, 54] experimental assays have been developed. *In vitro* assays such as SELEX [29, 49, 50] or PBM [45, 46] measure the protein-DNA affinity of short sequences (30-100 bp) in the presence of a single TF. The advantage of *in vitro* assays is that they can accurately determine the binding affinity. However, since these assays probe binding of a single TF to the naked DNA, the measured binding affinities might not correspond to the binding affinities *in vivo*. These are also influenced by binding of other TFs nearby as well as the chromatin context.

In vivo assays such as ChIP-seq or ChIP-exo probe TF binding to the genomic DNA and hence also capture the influence of chromatin organization and binding of other TFs (detailed description of ChIP-seq/exo is available in Section 2.4.2). These assays use the next generation short-read DNA sequencing technology [57] coupled with different experimental steps that enrich for genomic DNA fragments bound by a TF (ChIP-seq [52], ChIP-exo [53], ChIP-nexus [54]; explained in Section 2.4). After the enrichment step of the desired short sequence fragments (typically of 100-300 bp long), the fragments are sequenced and aligned to the genome to determine their original location. Thanks to the sufficient length of the sequenced DNA, the reads can typically be assigned to one unique position in the genome. The output of these assays is a genome-wide coverage track of aligned sequencing reads (Figure 1.5). Specific local enrichment of reads manifested as a peak in the coverage track indicates sites in the genome bound by the TF. These sites can be to some extent determined by peak callers such as MACS2 [58], GEM [59] or SPP [60]. In addition to the local enrichment of reads, the read coverage track profile

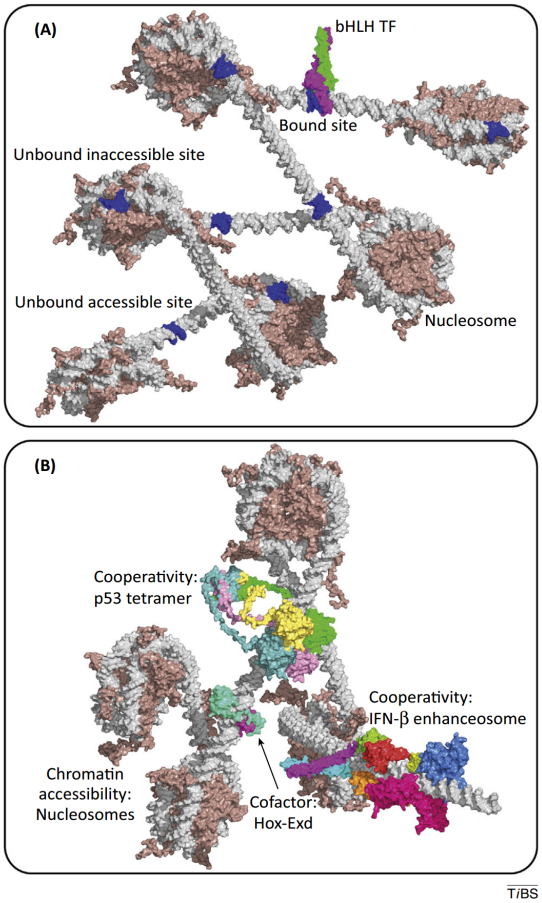


Figure 1.4: Structure-based illustration of multiple levels of TF-DNA binding specificity. (A) The basic helix-loop-helix (bHLH) Mad-Max heterodimer binds to only a subset of putative binding sites (blue). Some TFBSs are inaccessible due to nucleosome formation, while other accessible TFBSs are not selected by the TF by chance. **(B)** Higher-order determinants of TF binding include cooperativity with cofactors (e.g. Hox-Exd heterodimer), multimeric binding (e.g. p53 tetramer), cooperativity through TF-TF interactions (e.g. IFN-β enhanceosome) and chromatin accessibility due to nucleosome formation. Taken from [13].

gives important clues on the type of interactions present on the DNA especially in high-resolution TF binding assays such as ChIP-exo or ChIP-nexus.

1.3.3 Computational models of TF binding

Computational models of TF binding aim to accurately predict the TF-DNA binding affinity given the DNA sequence or aim to rank locations in the genome according to their likelihood to be bound by a TF in a particular cell type or tissue. These methods span a wide spectrum of modeling techniques including the simple position weight matrix

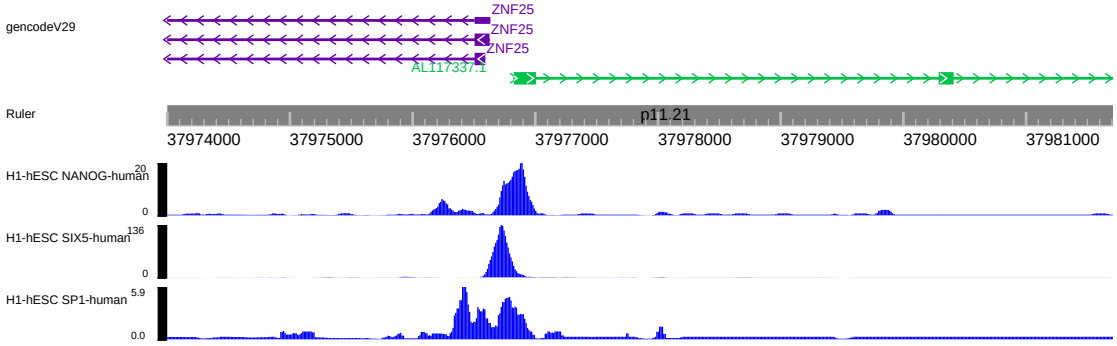


Figure 1.5: ChIP-seq coverage track showing binding of NANOG, SIX5 and SP1 transcription factors in human embryonic stem cells at the promoter of two genes. Screenshot from the WashU genome browser [55] using data from the ENCODE consortium [56].

introduced in 1982 [61] and the more complex convolutional neural networks applied to model TF binding first in 2015 [62, 63, 64].

1.3.3.1 Position weight matrix

Position weight matrix (PWM) characterizes the binding preference of a TF using a matrix of scores, where a single entry $w_{b,i}^{PWM}$ corresponds to the preference for base $b \in \{A, C, G, T\}$ at position $i \in \{1, \ldots, L_W\}$ in the binding site. The length of the binding site and thereby the length of the position weight matrix L_W is typically 6-20 bp. Let $S(i)$ return the sequence nucleotide at position i (i.e. $S(2)$ for 'GAC' returns 'A') and let $\mathbf{s}$ represent the one-hot encoded sequence such that $s_{S(i),i} = 1 \ \forall i = 1, \ldots, L_W$ and $s_{b,i} = 0 \ \forall i, \forall b \neq S(i)$. The preference of a TF towards the whole binding site sequence is the sum of individual base preferences:

$$\text{score} = \sum_{i=1}^{L_W} w_{S(i),i}^{PWM} = \mathbf{s} \cdot \mathbf{w}^{PWM} \ , \tag{1.1}$$

where $\cdot$ denotes the dot product operation.

For a set of DNA sequences bound by the TF at the same position, the coefficients of the PWM can be determined as follows

$$w_{b,i}^{PWM} = -\log \frac{f_{b,i}}{p_b} \ , \tag{1.2}$$

where f_{bi} is the frequency of base b at position i, also called the position frequency matrix PFM, and p_b is the frequency of base b in background sequences. If these sequences were identified as bound from a pool of random sequences using some experimental procedure, then the coefficient in the $w_{b,j}^{PWM}$ can be interpreted as the binding energy contribution of each base b at position i [65, 17, 66].

If the TF is bound at different sequence positions, then an expectation-maximization algorithm can be used [67, 68] to infer the PWM and the binding site positions within the sequences. After the PWM is obtained, the binding site candidates within a new sequence can be determined by computing the match score (Equation (1.1)) at each position within the sequence and discard sites with a score lower than some threshold. PWM predicts the binding affinity reasonably well for *in vitro* data [18]. Additionally, since the PWM is simple, easy to understand, easy to visualize using the sequence logo plot [69, 70], and since the potential binding sites within a new sequence can be determined at single base resolution, the PWM is still commonly used in the bioinformatics community.

However, if we scan the genome for potential binding sites using the PWM, most matches will likely be false positives [13]. As shown in Figure 1.4, TF binding code is complex and hence difficult to capture using a matrix with 20x4 coefficients. The PWM does not account for higher order dependencies among bases which can occur in practice due to the DNA shape [13]. Additionally, the sequence context beyond the binding site is not considered despite being important due to the interaction with other TFs *in vivo*. We note that some aspects (not all) of the PWM such as modeling higher order dependencies among bases were improved with algorithms such as the interpolated Markov models [71] and more recently the BAMM!motif [72].

1.3.3.2 MatrixREDUCE and FeatureREDUCE

The PWM is derived from a set of sequences bound by a TF. However, such an approach discards quantitative information for assays such as PBM, where a quantitative affinity measurement is provided with each probed DNA sequence. The MatrixREDUCE [73] extends the PWM framework to quantitative data. It also scans the sequence using a PWM-like matrix called position specific affinity matrix (PSAM). PSAM is defined as

$$w_{b,i}^{PSAM} = e^{\Delta\Delta G_{b,i}/RT} , \tag{1.3}$$

where $\Delta\Delta G_{b,i}$ is the change in free energy if the reference base at position i in the sequence is replaced by another base b. By definition, $\Delta\Delta G_{b,i} = 0$ if base b is the reference base at position i. MatrixREDUCE assumes the binding energies of individual bases ($\Delta\Delta G_{b,i}$) are independent of each other. Hence, the binding affinity of a TF to sequence $\mathbf{s}$ of length $L_S \geq L_W$ can be written as

$$\text{affinity} = C \sum_{i=1}^{L_S - L_W + 1} \prod_{j=1}^{L_W} w_{S(i+j-1),j}^{PSAM} + b \tag{1.4}$$

$$= C \sum_{i=1}^{L_S - L_W + 1} \exp\Big(\sum_{j=1}^{L_W} \log w_{S(i+j-1),j}^{PSAM} \Big) + b \tag{1.5}$$

$$= C \sum_{i=1}^{L_S - L_W + 1} \exp\left(\mathbf{s}_{:,i:i+L_W} \cdot \log \mathbf{w}^{PSAM} \right) + b \, . \tag{1.6}$$

Coefficients C and b are the scaling factor and the offset correspondingly. The notation $\mathbf{s}_{:,i:i+L_W}$ subsets the matrix $\mathbf{s}$ on the second index between positions i and $i + L_W$. Logarithm operates element-wise.

Instead of using a generative modeling approach of searching for enriched sequences to determine the PWM, MatrixREDUCE performs L2 regression to the output quantity (e.g. fluorescence intensity for the PBM assay). It thereby learns a $\mathbf{w}^{PSAM}$ that best explains the target variable. FeatureREDUCE [74] extends MatrixREDUCE by also accounting for di-nucleotide binding energies and by modeling the positional preference within the sequence.

The key limitation of the MatrixREDUCE (and FeatureREDUCE) is that it only models a single binding site. Hence, it is more appropriate for *in vitro* data with short sequences bound by a single TF. For explaining TF binding *in vivo*, it suffers from similar issues as the PWM. While the MatrixREDUCE (and FeatureREDUCE) biophysical approach provides some additional modeling flexibility (i.e. to model quantitative data), it can not be directly extended to more expressive models due to the complex multi-step optimization procedure.

1.3.3.3 Convolutional neural networks

MatrixREDUCE and PWM scanning can be seen as a special case of convolutional neural networks. We can re-write Equation (1.4) specifying MatrixREDUCE using a convolutional neural network layer:

$$f(\mathbf{s}) = \text{pool}_{\text{AVG}}(\exp(\text{conv}_1(\mathbf{s}))) \,, \tag{1.7}$$

where conv_1 represents the convolutional neural network layer with a single filter (Section 2.3.3.1), exp represents the exponential activation function, and pool_{AVG} is the global average pooling computing the average across the spatial axis (Section 2.3.3.1). The advantage of such formulation is that it can be easily extended with other layers or modeling techniques developed by the broad deep learning community. Also, the deep learning software frameworks used to train neural networks are mature, easy to use and do not require modelers to manually derive the optimization procedure.

DeepBind [62] was the first application of convolutional neural networks to model TF binding. The model architecture–composition of neural network layers–of DeepBind can be written as:

$$f(\mathbf{s}) = \text{FCN}(\text{pool}(\text{ReLU}(\text{conv}_M(\mathbf{s})))) \,, \tag{1.8}$$

where $\text{ReLU}(x) = \max(0, x)$ is the rectified linear activation function and FCN is the fully-connected layer with 0 or 1 hidden layers (Background Section 2.3.1, [62]). The pooling operation (pool) computes the maximum or average value across all positions for each channel in the activation map. DeepBind extends MatrixREDUCE by using multiple (M) filters in the convolutional layer followed by a fully connected layer (described more in detail in the background Section 2.3.3.1). Hence, the convolutional layer in DeepBind can be seen as scanning the sequence with multiple PWMs. DeepBind improved the predictive accuracy for predicting TF binding across a wide variety of TFs

[62]. The architecture DeepBind does not require multiple optimization steps involving motif seeding such as MatrixREDUCE and allows to model multiple binding sites in the sequence. Additionally, DeepBind was not only designed to work with PBMs as Matrix/FeatureREDUCE, but also allows to model binary outputs (peak vs no-peak). However, because DeepBind is using only a single convolutional layer applied to a short DNA sequence (100 bp), motif combinations as well as chromatin context can not be sufficiently captured.

To capture the interactions between multiple TF binding sites and possibly the chromatin context, DeepSEA [63] and Basset [64] used 3 convolutional layers and a large input sequence (1 kb and 400 bp). Additionally, they were trained to predict multiple binary output variables simultaneously (919 and 164). Such multi-task approach allows individual tasks (e.g. binding of a particular TF) to borrow statistical strength from other tasks (Section 2.3.6). Adding more layers, and thereby modeling the context, significantly improved the predictive performance compared to DeepBind [1].

Despite the tremendous modeling improvements in the recent years [62, 63, 64], *in vivo* predictive accuracy for some TFs is still not satisfactory [75]. Additionally, by introducing more parameters into predictive models such as deep neural networks to improve their predictive performance, the interpretability of the parameters is compromised. We will explain these two issues more in detail in the last section of this chapter.

1.4 Understanding molecular phenotypes beyond TF binding via predictive models

Transcription factor binding sites control transcription, the first step of gene expression. There are thousands of other regulatory sequences in the genome controlling other molecular phenotypes involved in gene expression. A wide range of molecular phenotypes involved in gene expression can be measured today thanks to the recent development of genome-wide assays such as DNase-seq [76], ATAC-seq [77], CUT&RUN [78], RNA-seq [79], Hi-C [80], iCLIP [81], eCLIP [82], Mass-spec [83], TT-seq [84] as well as assays probing sequence variation such as MPRA [85] and CRISPR-Cas9 [86, 87]. Because of the availability of training data and the development of flexible models such as NNs, we have seen many successful applications of sequence-based predictive models in genomics predicting a wide range of molecular phenotypes beyond TF binding including chromatin accessibility and splicing efficiency in the recent years (Table 1.1) [88, 62, 63, 75, 89, 90, 91]. We note that these models are conceptually very similar to the TF binding models described in the previous section.

There are four main use-cases for accurate sequence-based predictive models in genomics. First, they can be used to impute the missing data such as methylation [97] or TF binding in other cell types [75]. Second, they can improve our understanding of the biological processes through model interpretation. Third, models predicting molecular phenotypes such as TF binding can be used as building blocks in models predicting more complex molecular phenotypes such as gene expression. Fourth, they can be used to predict the pathogenicity of genetic variants. Here, we will describe more in detail

Mol. phenotype	Model	Publication
TF binding	PWM Scanning (Jaspar, HOCOMOCO)	[92, 93]
	DeepBind	[62]
	FactorNet	[75]
	lsgkm-SVM	[94]
Chromatin	DeepSEA	[63]
	Basenji	[95]
DNA methylation	CpGenie	[96]
	DeepCpG	[97]
DNA accessibility	Basset	[64]
RNA expression	Basenji	[95]
	ExPecto	[98]
RBP binding	iDeep	[99]
miRNA binding	TargetScan	[100]
	deepMiRGene	[101]
Splicing	MaxEntScan 5', 3'	[102]
	Labranchor	[91]
	HAL	[90]
Polyadenylation	APARENT	[103]
Translation	Optimus_5Prime	[104]

Table 1.1: List of predictive models predicting differnet molecular phenotypes from DNA or RNA sequence.

the last two of these applications: modular modeling of complex phenotypes and variant effect prediction.

1.4.1 Modules for more complex models

Since gene expression involves multiple steps (Figure 1.1), the predictive models trained on more basic steps such as TF binding can inform models predicting more complex phenotypes such as RNA expression. One such example is ExPecto [98], which predicts gene expression from TF binding, DNA accessibility and histone modification predictions. We note that ExPecto is a recent method published in parallel with the developments of this thesis. Such modular approach has multiple advantages. First, predicting a complex trait such as gene expression or cell growth directly from DNA sequence can be difficult because there is relatively little data available compared to the complexity of the phenomena. By contrast, predicting simpler molecular phenotypes from DNA sequence is easier since there is enough data to train an accurate model. Second, modularity is an inherent property of biology. Hence, learning a good predictive model of a core biological process can be useful for many other predictive tasks.

1.4.2 Scoring genetic variants

Understanding how genetic variants lead to genetic diseases is of immense importance for both better diagnostics as well as drug development [10]. First studies using whole exome sequencing (WES) on large cohorts of patients suffering from rare Mendelian diseases were able to pinpoint the causal mutation for roughly 25%–30% of the patients using simple interpretation rules for variants in the coding part of the genome [105]. The other 70% of the patients, remains undiagnosed. The causal variants for the undiagnosed ones are likely located in the non-coding part of the genome [105]. Genetic diseases, especially rare Mendelian diseases, are caused either due to the modification of the protein sequence or through inappropriate expression of the protein in a certain tissue [106]. Moreover, 90% of the GWAS variants associated with common diseases are actually located in the non-coding part of the genome and are hence influencing gene expression and not protein sequence [107]. Hence, there is an urgent need for tools which allow to interpret non-coding variants.

Sequence-based predictive models have the potential to serve as an excellent tool to interpret these genetic variants. Specifically, the impact of a genetic variants on molecular phenotypes can be assessed by comparing model prediction for the reference sequence to model prediction for the mutated sequence containing the variant of interest. The difference in model predictions serves as a proxy on how strongly the variant of interest perturbs the molecular phenotype. This approach, also termed in-silico mutagenesis, has been successfully employed across a wide range of studies [62, 63, 64, 96]. While molecular phenotypes do not yet represent the phenotype of interest (e.g. disease risk), understanding the impact of the variant on the disease can be much easier if we would know the impact on some molecular phenotypes. Often, a drastic impairment of some molecular phenotypes such as splicing can lead to a total depletion of the protein. In summary, accurate sequence-based predictive models are not only relevant for better understanding of the genome, but also for human genetics and biomedicine.

1.5 Aims and scope of this thesis

The contribution of this thesis towards the understanding of the regulatory code in the human genome is three fold:

1. Develop more accurate sequence-based models by extracting more knowledge from the raw data and integrating the context information.

2. Develop interpretation tools applicable to black-box predictive models such as neural networks to detect motifs, map them back to the genome and infer their cooperativity.

3. Develop a platform to exchange predictive models in genomics with focus on sequence-based predictive models as well as support for model interpretation and variant scoring.

Better sequence-based models for TF binding

Despite the more complex models such as NNs, predictive accuracy is still very low for some TFs [1]. One reason for low performance is that models such as DeepBind and DeepSEA treat the TF binding data as binary classification. Doing so, they discard the information about the amount of aligned reads as well as the read coverage profile shape. Both, the amount of aligned reads and the profile shape, namely contain important information about the binding event especially when multiple TFs are co-bound. Learning from the read coverage profile is especially important for high-resolution assays such as ChIP-exo and ChIP-nexus.

Here, I develop BPNet, an end-to-end neural network that predicts the profile shapes of ChIP-nexus and ChIP-seq data at base-pair resolution from DNA sequence. To this end, I develop a special model architecture, loss function, and a way to control for assay specific biases. I use BPNet to model ChIP-nexus profiles of pluripotency TFs Oct4, Sox2, Nanog and Klf4 in mouse embryonic stem cells (mESCs) [108]. BPNet accurately predicts ChIP-nexus binding footprints of the four TFs at single-nucleotide resolution on par with replicate experiments.

Better sequence-based models for RBP binding and splice branch-point prediction

Low predictive performance for some predictive tasks such as predicting the binding sites of RNA-binding protein (RBPs) can be partly attributed to poor context modeling. Distances to different genomic landmarks defined by gene annotation such as transcription start site or exon boundaries can partially capture this context. I develop a novel neural network layer termed spline transformation to flexibly and robustly model distances. By including distances to genomic landmarks into a convolutional neural network model using spline transformation, I improve the accuracy for predicting *in vivo* RBP binding sites and splice branchpoint locations. Spline transformation combined with a single convolutional layer generalizes FeatureREDUCE, similar to how DeepBind generalizes MatrixREDUCE.

To avoid unnecessary paper re-writing and self-plagiarism, the description and results of this method are not included in this thesis. Instead, the reader can find them in the following open-access publication:

Modeling positional effects of regulatory sequences with spline transformations increases prediction accuracy of deep neural networks
Žiga Avsec, Mohammadamin Barekatain, Jun Cheng, Julien Gagneur
Bioinformatics 2017, `https://doi.org/10.1093/bioinformatics/btx727`

Improved interpretation of neural networks modeling TF binding

The ideal TF binding model could accurately predict TF binding by capturing the whole complexity as illustrated in Figure 1.4 including TF binding sites of other TFs while still being interpretable. There are three key interpretation tasks: motif discovery, mapping motifs back to the genome and discovering the interactions between motifs

(i.e. the motif grammar). Accurate prediction of TF binding requires complex models such as NNs. However, by making the model more complex, the model parameters can not be interpreted any more in contrast to simpler models such as PWMs. Therefore, complex models such as DeepSEA, Basset, and DeepBind are often considered to be black-box predictors taking as input a long DNA sequence (100-1000 bp, much longer than the binding site) and predicting whether there is some binding happening within this sequence. This illustrates the classical tension between the explainability and the interpretabily of models [109]. While DeepBind and Basset performed motif discovery through the interpretation of convolutional filters, we note that their procedure was fragile and incomplete as the learned convolutional filters rarely represent complete binding motifs [64].

One of the goals of this thesis is to resolve the tension between model flexibility and interpretability, and to show that even if the model is very complex and contains many parameters (>100,000), it can still be interpreted in terms of simple rules. I develop a suite of model interpretation tools to extract motifs and motif cooperativity from the sequence-to-profile model BPNet. Specifically, I extend the current repertoire of model interpretation tools such as DeepLIFT [110] and TF-MoDISCo [6] in three ways:

- Allow DeepLIFT to be used with sequence-to-profile models such as BPNet.
- Develop a method called contribution weight matrix (CWM) scanning which maps the motifs discovered by TF-MoDISco back to the genome.
- Design the pairwise spacing analysis using synthetic and genomic sequences to extract interactions between motifs.

I apply this method to the four mouse pluripotency transcription factors Oct4, Sox2, Nanog and Klf4, where I accurately map hundred thousands of motifs in the genome and, together with collaborators, identify rules by which motifs influence the cooperative binding of transcription factors. The key biological questions I tackle are:

- What are the key motifs involved in the binding of Oct4, Sox2, Nanog and Klf4 in mESC? Which motifs are directly and ones indirectly bound?
- What are the spacing preferences of the motifs?
- Does binding of one TF influence binding of another TF and vice versa? How strong is this effect for different motif spacing?

Platform to exchange predictive models in genomics

Despite the long list of published sequence-based models (Table 1.1), it is very difficult to run them on new data, interpret them, use them as modules in more complex models, or score the impact of genetic variants with them. Lack of standards and limited centralized access to these trained models have hampered their practical impact. To address this, I developed Kipoi, a platform to exchange predictive models in genomics in collaboration with other co-authors of [1] (author contributions are described in Section). I illustrate

Kipoi for canonical use cases including model benchmarking, transfer learning, variant effect prediction, and the definition of new models from existing ones.

Parts of this thesis have already been published. The respective publications and the contributions of other co-authors are clearly indicated at the beginning of each chapter. The first person plural form 'we' is used throughout the thesis to avoid unnecessary switching between forms 'I' and 'we'.

2 Background

2.1 Supervised learning

In supervised learning, one is given a training dataset $(\mathbf{X}^{train}, \mathbf{y}^{train})$ comprising of N pairs of input $\mathbf{x}_i$ and target data y_i. While both, $\mathbf{x}_i$ and y_i can be tensors of arbitrary shape typically containing real numbers, we here restrict ourselves to $\mathbf{x}_i \in \mathbb{R}^D$ and $y_i \in \mathbb{R}$. The goal is to obtain or *learn* a predictive model—a parametrized mathematical function $f(\mathbf{x}; \mathbf{w})$—from the training dataset $(\mathbf{X}^{train}, \mathbf{y}^{train})$ such that the model predictions on new, unseen data called the test set $(\mathbf{X}^{test}, \mathbf{y}^{test})$ will be as accurate as possible. The accuracy of predictions is measured by the *evaluation metric* which compares model predictions to the target variable across the whole dataset. We note the evaluation metric does not need to be differentiable. One example evaluation metric is classification accuracy.

Learning the predictive model means to adapt the model parameters $\mathbf{w}$ using the training dataset such that model predictions become more accurate. We will assume that the data points $(\mathbf{x}_i, y_i)$ are independent and identically distributed. Also, we will assume there exists a differentiable loss function $L(y_i, \hat{y}_i)$ which measures how well the target variable y_i and model prediction $\hat{y}_i$ agree (the smaller the better). Loss function choice will be discussed in the later sections. Under these assumptions, the objective function $J(\mathbf{w})$ for supervised learning can be written as

$$J(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^{N} L(y_i^{train}, f(\mathbf{x}_i^{train}; \mathbf{w})) \, . \tag{2.1}$$

Parameters $\mathbf{w}$ are obtained by minimizing the objective function $J(\mathbf{w})$. We note that supervised learning is different from classical optimization since the goal is to obtain the best predictions on the test data as measured by the evaluation metrics and not necessarily find the minimum of the objective function $J(\mathbf{w})$.

Taken together, a supervised machine learning problem is specified by

- training dataset: $(\mathbf{X}^{train}, \mathbf{y}^{train})$,
- test dataset: $(\mathbf{X}^{test}, \mathbf{y}^{test})$,
- evaluation metric.

To solve this problem, three components need to be specified:

- predictive model $f(\mathbf{x}; \mathbf{w})$,

- loss function $L(y_i, \hat{y}_i)$,
- optimization procedure for the objective function $J(\mathbf{w})$.

In this chapter, we will present linear models and neural networks as an example of the parametrized functions. We will focus on the first-order optimization methods such as gradient descent and will discuss loss functions useful for count-based target variables frequently occurring with sequencing-based assays.

2.2 Generalized linear models

2.2.1 Linear regression

Let $\mathbf{x}_i$ be a D-dimensional real vector and y_i a real scalar. In linear regression, the parametrized function $f(\mathbf{x}; \mathbf{w})$ is a weighted sum of the input features $\mathbf{x}_i$:

$$f(\mathbf{x}; \mathbf{w}) = w_1 x_1 + \ldots + w_D x_D \,. \tag{2.2}$$

The loss function is the residual sum of squares:

$$L(y, f(\mathbf{x}; \mathbf{w})) = (y - f(\mathbf{x}; \mathbf{w}))^2 \,. \tag{2.3}$$

We define the parameter vector $\mathbf{w} \in R^D$ with components w_j, the vector $\mathbf{y}$ of target values with components y_i, and the design matrix $\mathbf{X}$ as

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_N^T \end{pmatrix} = \begin{pmatrix} x_{11} & \ldots & x_{1D} \\ \vdots & \ddots & \vdots \\ x_{N1} & \ldots & x_{ND} \end{pmatrix} = (\mathbf{x}_1, \ldots, \mathbf{x}_D) \in R^{N \times D} \,. \tag{2.4}$$

We assume that the columns of $\mathbf{X}$ are linearly independent, i.e. $\mathbf{X}$ has full rank. Nice property of linear regression is that the minimum of the objective function $J(\mathbf{w})$ can be obtained analytically:

$$\mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \,. \tag{2.5}$$

2.2.2 Regularization

Since the goal of the model is to generalize to unseen data, simpler models should be preferred over more complicated ones [111]. Namely, even though a more complex model can perform excellently on the training data, it can still miserably fail on the unseen data. This happens when the model overfits the training data by simply memorizing the target values $\{y_1, \ldots, y_N\}$ instead of learning the rules to predict y_i from $\mathbf{x}_i$. To favor simpler models and to overcome issues if $\mathbf{X}$ is not full rank, a widely used elastic net regularization term [112] can be added to the loss function[1]. Elastic net penalizes large values of $\mathbf{w}$

$$L(y, f(\mathbf{x}; \mathbf{w}), \mathbf{w}) = (y - f(\mathbf{x}; \mathbf{w}))^2 + \alpha |\mathbf{w}|_2 + \beta |\mathbf{w}|_1 \,. \tag{2.6}$$

[1] We note that there are many other ways to regularize the model. Dropout, described in Section 2.3.3.5, is one of them.

First additional term $|\mathbf{w}|_2$ is called the ridge penalty and the second term $|w|_1$ is called the LASSO penalty [113]. While the ridge regularization prefers to have all parameters closer to 0, the LASSO regularization prefers to have most parameters exactly equal to 0 while allowing a few parameters to be large and non-zero. Hence LASSO acts as a feature selection method [113]. Nice property of these two terms is that the objective function $J(\mathbf{w})$ is still convex and hence easy to optimize.

2.2.3 Hyper-parameter optimization

The scalars α and β in Equation (2.6) are hyper-parameters. Hyper-parameters are parameters of the predictive function, loss function or the training procedure which are not optimized during model training (i.e. during the optimization of the objective function). Instead, they are optimized to yield the best evaluation metric on held-out data using techniques such as random search [114] or Bayesian optimization [115]. Since test dataset it not available during model training or hyper-parameter tuning, a subset of the training set, called the validation set, is held out from model training to optimize the hyper-parameters.

2.2.4 Logistic regression

Linear regression uses the residual sum of squares loss function which is appropriate for regression. Logistic regression extends the linear regression framework to binary classification where the target variable y_i can only take values 0 or 1. It achieves this by using a sigmoid activation function which maps the model output to the [0, 1] range representing the probability of class 1:

$$f(\mathbf{x}; \mathbf{w}) = \sigma(w_1 x_1 + \ldots + w_D x_D) \in [0, 1] \,, \tag{2.7}$$

where σ is the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$. Categorical cross entropy is used for the loss function written as

$$L(y, f(\mathbf{x}; \mathbf{w})) = y \log f(\mathbf{x}; \mathbf{w}) + (1 - y) \log(1 - f(\mathbf{x}; \mathbf{w})); \,. \tag{2.8}$$

While the objective function of logistic regression is still convex, the analytical solution does not exist anymore. Hence, numerical optimization such as gradient descent (first order method) or Newton's method (second order method) have to be used to optimize the objective function and thus to train the model.

2.2.5 Loss function for count distributions

Generalized linear models provide a common framework to deal with a wide range of target variable distributions while still using a weighted sum of the input features as the predictive model. The task of supervised learning can be framed as estimating the conditional distribution $p(y|\theta = f(\mathbf{x}; \mathbf{w}))$, where θ are the parameters of the distribution (e.g. mean of the normal distribution). For generalized linear models, the predicted

parameters θ typically correspond to the distribution mean while the scale of the distribution is kept fixed. The model is accurate when the observed data are very likely under the conditional distribution, that is when $\prod_i^N p(y_i|\theta = f(\mathbf{x}_i; \mathbf{w}))$ is large assuming the data points are independent and identically distributed (i.i.d.). Hence the log-likelihood of the conditional distribution can be used for the loss function

$$L(y, f(\mathbf{x}; \mathbf{w})) = -\log p(Y = y|\theta = f(\mathbf{x}; \mathbf{w})) \,. \tag{2.9}$$

For example, if we write $p(y|\theta = f(\mathbf{x}; \mathbf{w})) = \mathcal{N}(y|\mu = f(\mathbf{x}; \mathbf{w}), \sigma^2 = 1)$, we obtain the squared loss function used in linear regression. Similarly, if $p(y|\theta = f(\mathbf{x}; \mathbf{w})) = \text{Bernoulli}(y|\theta = f(\mathbf{x}; \mathbf{w}))$, then we obtain the loss function of logistic regression. Hence, to train a model from the count data, which are frequently present in genomics, we can use a count-based distribution for $p(y|\mathbf{x})$. For example, $p(y|\mathbf{x})$ can be the Poisson distribution:

$$p(y|\mu) = \frac{\mu^x e^{-\mu}}{x!} \,. \tag{2.10}$$

If we are estimating the values of multiple (T) count variables $y_{i,j} \in \{0, 1, 2, \ldots\}$ with a fixed and pre-defined total sum $n_i = \sum_j^T y_{i,j}$, a multinomial distribution can be used:

$$\text{Multinomial}(\mathbf{y}_i|\mathbf{p} = \mathbf{f}(\mathbf{x}_i; \mathbf{w}), n_i) = \frac{n_i!}{y_{i,1}! \cdots y_{i,T}!} \, p_{i,1}^{y_{i,1}} \times \cdots \times p_{i,T}^{y_{i,T}} \,, \tag{2.11}$$

where $\mathbf{p}$ is the vector of probabilities for each output variable such that $\sum_j^T p_j = 1$. Hence, the function $\mathbf{f}(\mathbf{x}; \mathbf{w})$ has to return a vector with values from range [0, 1] that collectively sum to 1. This can be obtained by using a different set of parameters $\mathbf{w}_j$ for each target variable j and the softmax activation function $\text{softmax}(x_1, \ldots, x_T)_j = \frac{e^{x_j}}{\sum_{k=1}^T e^{x_k}}$ such that

$$f_j(\mathbf{x}_i; \mathbf{w}) = \text{softmax}(\mathbf{w}_1^T \mathbf{x}_i, \ldots, \mathbf{w}_T^T \mathbf{x}_i)_j \,. \tag{2.12}$$

2.2.6 Feature engineering

The key question when building models that predict different values from DNA sequence is how to transform the sequence, for example 'ATCG', into a D-dimensional vector of real numbers. Generalized linear models such as linear regression and many other supervised learning algorithms such as random forests [116] require $\mathbf{x}_i \in \mathbb{R}^D$. Without the loss of information, each letter in the sequence 'ATCG' can be transformed using the so called 'dummy' encoding as follows: A can be mapped to [1,0,0], C to [0,1,0], G to [0,0,1] and T to [0,0,0]. To obtain the feature vector for the entire sequence, the vectors of individual letters can be concatenated. Additionally, a feature always taking value 1 can be added to represent the intercept in Equation (2.2).

While this featurization of the DNA sequence is lossless as we can reconstruct the original sequence given the feature vector, it may be inappropriate for sequences where the regulatory motifs occur at different positions within the sequence while still exhibiting the same behaviour. Assume the molecular phenotype (for example gene expression)

is high when there is a CG substring present in the large string. If we use the dummy encoding, then sequences 'ACGT' and 'ATCG' will have very different feature vectors and a linear model will not be able to exactly distinguish between sequences containing a CG and not containing a CG. To create translationally invariant features, the number of k-mers in the sequence can be used as the feature vector. In that case, the sequences containing the CG subsequence can be easily distinguished as the model will predict a high value whenever the number of CG instances is larger than 0. However, some important properties of the regulatory sequences can be lost during the k-mer featurization. For example, k-mer featurization can not capture the spacing between motifs since the location of the k-mer is not stored in the feature vector.

2.3 Deep neural networks

This section partially follows the argumentation and uses figures from the manuscript 'Deep learning: new computational modelling techniques for genomics' [1]. Author contributions for the original publication are described in Section . Overlapping content was reformulated and significantly extended to provide more technical details.

2.3.1 Modeling non-linear dependencies with fully-connected neural networks

So far, predictive models using a linear combination of input features were discussed. Since any pre-defined transformation of the input data can be used to generate the features (for example using sequence k-mers or polynomials of different degree), linear models can also capture non-linear dependencies between the raw input features. However, it is often difficult to hand-craft these feature transformation. Instead, we would like the predictive model to learn the optimal feature transformation directly from the raw data. This can be achieved by using a parameterized function to compute the feature transformation. For example, we can require the feature vectors to be the output of a linear model followed by a non-linear activation function σ:

$$f(\mathbf{x}; \mathbf{w}) = w_1^{(1)} \sigma(w_{11}^{(0)} x_1 + w_{12}^{(0)} x_2 + \ldots) + w_2^{(1)} \sigma(w_{21}^{(0)} x_1 + w_{22}^{(0)} x_2 + \ldots) + \ldots \quad (2.13)$$

This formulation corresponds to a fully-connected neural network (Figure 2.1). A frequently used activation function is the rectified linear unit (ReLU) defined as $\text{ReLU} = \max(0, x)$. We note that neural networks are a generalization of (generalized) linear models. Hence loss functions for count distributions presented in the previous section are also applicable to neural networks.

By using a fully connected neural network, the hidden layers transform the input features into a feature space where the final task of regression or classification can be more easily performed (Figure 2.1B).

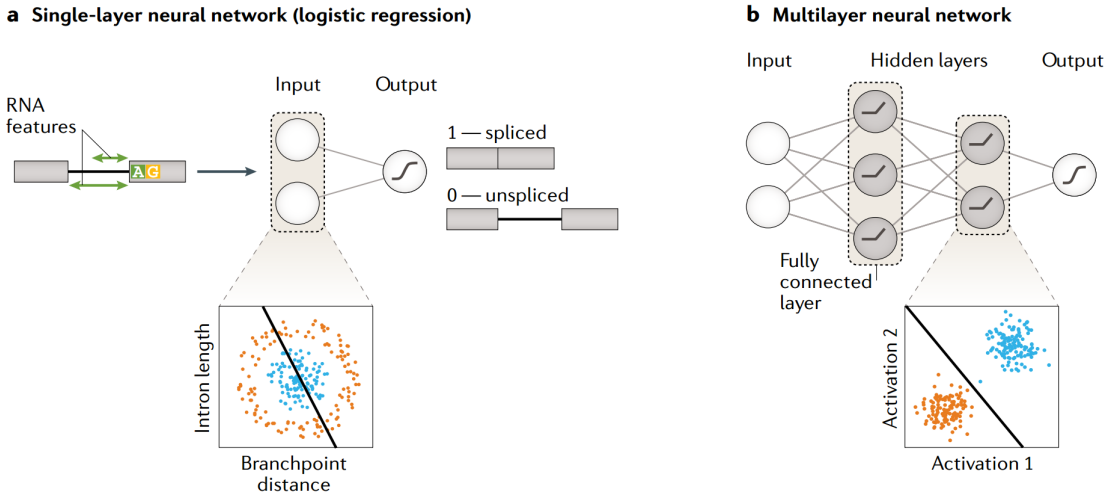


Figure 2.1: Neural networks with hidden layers model nonlinear dependencies. A) Example of splice site classification given pre-defined features (intron length and branchpoint distance). Depicted is a single-layer neural network with sigmoid activation function, which corresponds to logistic regression. In this example, the goal is to discriminate spliced-out from not spliced-out introns as a function of the intron length and the distance of the branchpoint to the acceptor site. Too short or too long distances prevent splicing (bottom). Hence, linear combinations of these two features as implemented in logistic regression cannot separate the spliced (blue) from the unspliced (orange) data points. **B)** Neural networks with intermediate layers, also called hidden layers, transform the inputs using nonlinear transformations into a space where the classes become linearly separable. The depicted layers are said to be fully connected because every neuron receives input from all neurons in the upstream layer. Deep neural networks are neural networks with many hidden layers. Taken from [3].

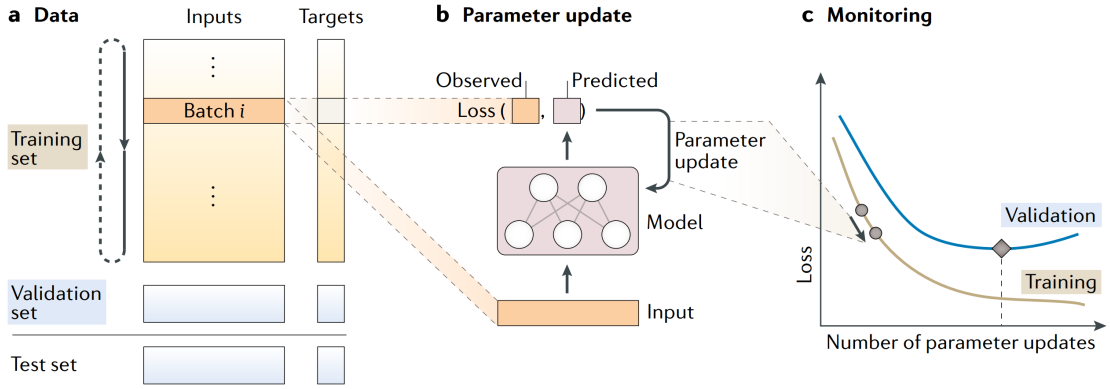


Figure 2.2: Deep learning training workflow. Taken from [3].

2.3.2 Training workflow

Due to the nested parameters in NNs (Equation (2.13)), the objective function has to be optimized numerically and is in most cases not convex anymore. Mini-batch gradient descent is the default optimization algorithm in neural networks for optimizing the objective function [117]. The model parameters are first randomly initiated (typically using a uniform or normal distribution). We note that the variance of the initialization distribution is an important parameter to prevent exploding or vanishing gradients. Typically the 'He normal' [118] or 'Glorot uniform' [119] initializations are used. After the random initialization, small subsets, so-called batches, of input-target pairs are iteratively sampled from the training set. For each batch, the gradient of model parameters w.r.t. the objective function defined over the batch of data is computed. This gradient computation can be efficiently performed using the backpropagation algorithm [120]. Backpropagation exploits the chain-rule of derivatives to incrementally compute the gradients of all model parameters starting from the loss function. The computational cost of backpropagation scales linearly with the number of model parameters. After the gradients of each parameter w.r.t. the batch objective function are computed, the NN parameters are updated using a single gradient descent step with a fixed step size η also called the learning rate:

$$w_j^{new} \leftarrow w_j^{old} - \eta \sum_{i \in \text{batch}} \frac{\partial L(y_i, f(\mathbf{x}_i; \mathbf{w}))}{\partial w_j} \,. \tag{2.14}$$

In practice, a modification of the gradient descent step called ADAM [121] is frequently used. ADAM dynamically adapts the learning rate based on past updates and adds a 'momentum' term to Equation (2.14) to overcome flat regions near saddle points of the objective function.

Model training can be stopped after the evaluation metric on the validation dataset (blue line in Figure 2.2c) stops improving or even starts degrading. This is namely a sign that the model started over-fitting the data. Early stopping is a convenient regularization technique since it does not add any extra hyper-parameters and it directly optimizes the

evaluation metric on the held-out (validation) dataset. Early stopping has also been shown to act like L2 regularization since the size of the parameters $\mathbf{w}$ is limited by the number of gradient descent update steps [117].

Since only a small random subset of the training set is used at each optimization steps, the optimization algorithm requires a constant amount of memory regardless of the total training dataset set size. This allows the model to be trained on very large datasets which may not fit into computer's main memory. Moreover, using only a small subset of the data introduces noise to the computed gradient. The added noise was shown to improve model's generalization performance since it acts as regularization [122, 123].

2.3.3 Neural network layers

2.3.3.1 Convolutional

We have seen that featurizing the DNA sequence into a D-dimensional vector using k-mer counts exhibits translational invariance. However, k-mer counts can not capture the scenario where multiple motifs are spaced at specific pairwise distances. Consider an example where the expression of a gene is high only when two TF motifs in the promoter sequence (say GATA and TAL1) are spaced within a certain distance. Such a pattern, also referred to as motif grammar, can be effectively captured by sequentially applying multiple convolutional (neural network) layers.

A convolutional layer can be seen as applying the same fully-connected layers (also called filters) locally to all sequence positions or to all patches of an image. In this section we will focus on the 1-dimensional convolutional layer applied to one-hot-encoded DNA sequences. As discussed in the introduction chapter, a convolutional layer applied to one-hot-encoded DNA sequence can be viewed as multiple position weight matrices (PWM) scanning the sequence [61, 17, 124] (Figure 2.3, first convolution). Since the same filter is used across all scanned positions, the total number of parameters is kept small regardless of the sequence length. Moreover, since the scanning procedure is translationally invariant, the model can generalize to motif positions not seen during training. The output of the convolutional neural network, also called the activation map, is followed by a nonlinear activation function, similar to fully-connected layers (Figure 2.3, ReLU).

To coarsen the activation map and reduce the sequence length, a pooling operation can be used. Pooling refers to aggregating local patches of the activation map using either the max or the average aggregation function (Figure 2.3, max pooling). We note that pooling can be also applied across the entire sequence length (global pooling). When multiple convolutional layers are used, pooling increases the receptive field of the downstream convolutional layers. Downstream convolutional layers can detect specific local arrangements of patterns that were detected by previous layers (e.g. two motifs spaced at a certain distance range, Figure 2.3e). In practice, the convolution operation is implemented using matrix multiplication. This allows dedicated linear algebra hardware accelerators such as GPUs to significantly speed up the computation. Most importantly, the parameters of the convolutional layer (i.e. filters) can be trained by backpropagation.

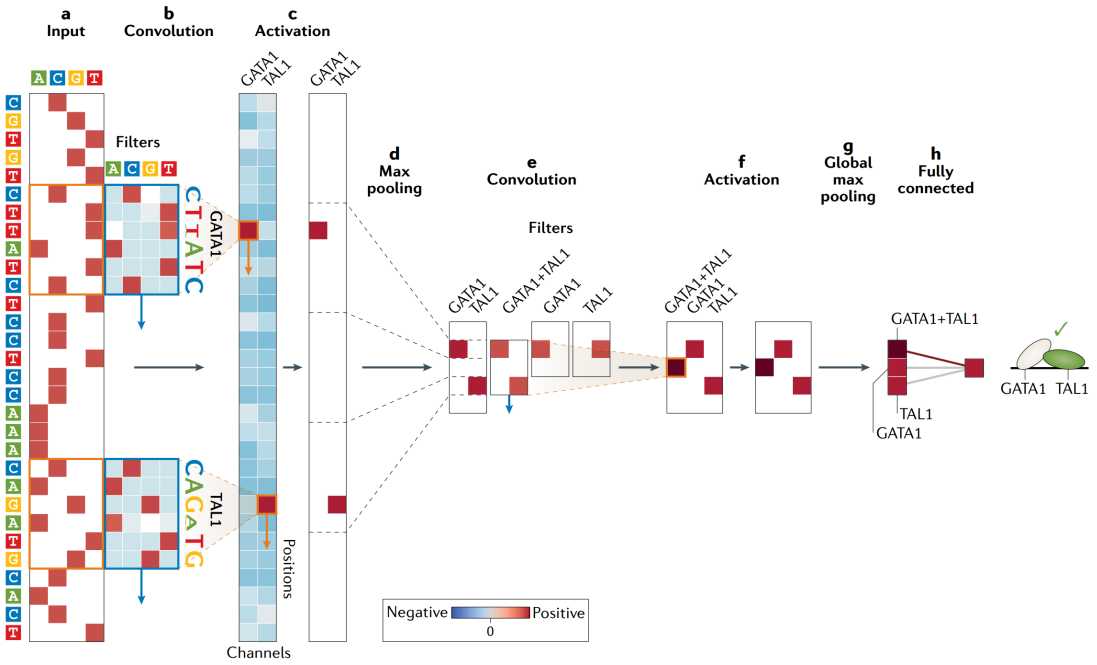


Figure 2.3: Convolutional neural network predicting whether GATA1 and TAL1 motifs are spaced at a particular distance range from DNA sequence. a) One-hot encoded DNA sequence is used as input. **b)** Multiple convolutional filters visualized as PWM matrices scanning the sequence. **c)** The output of the convolutional layer is passed through a non-linear activation function ($\mathrm{ReLU}(x) = \max(0, x)$). **d)** Max pooling is used to coarsen the signal in multiple contiguous bins of the activation map for each channel separately (2 columns of the matrix). **e,f)** The second convolutional filter allows to scan for motif combinations (GATA1+TAL1) followed by an activation function. **g)** Only the best match for each channel across the sequence is considered using global max pooling. **h)** The final prediction—whether GATA1 and TAL1 are spaced at a specific distance—is made using the fully-connected layer. Taken from [3].

Therefore, the two different types of neural networks, convolutional and fully-connected, can be used in the same model and can be also trained simultaneously. Therefore the quantity of interest can be modelled end-to-end. One 'end' (the input) refers to the raw one-hot-encoded DNA sequence and the other 'end' refers to the predicted quantity of interest.

There are two key advantages of convolutional neural networks compared to the classical PWM approach. First, the filters in the convolutional neural network are optimized to be most predictive for the task at hand. Hence, they are not derived using an unsupervised approach of determining the enrichment of sequences in positive classes as done for PWMs [125]. Second, by using multiple convolutional layers, motif combinations can be captured. Third, the end-to-end approach greatly simplifies the programming code. Manually derived features typically require complex code for their generation and potentially also additional external resources. Concretely, to be able to run or evaluate the model on a new dataset, both, the external resources (like PWMs) and the preprocessing code need to be provided.

2.3.3.2 Dilated convolutions

To increase the receptive field of convolutional layers, dilated convolutions can be used. Dilated convolutions use a special filter which skips some input values. The dilation rate—number of positions skipped by the filter—typically doubles with every subsequent dilated convolutional layer in the model. This allows the model to capture exponentially large receptive fields. One example of a dilated convolutional neural network is WaveNet [126] which allows to generate human speech of high quality. In genomics, Basenji [95] was the first model to employ dilated convolution and thereby achieved receptive fields of 32 kb.

2.3.3.3 De-convolutional

De-convolutional layer [127] can be seen as the inverse operation of the convolutional layer. The convolutional filter performs a weighted sum of the local input activation map and returns a single scalar. De-convolutional filter takes as input a single scalar value and outputs the whole patch of activations. This property is useful when building models that generate high-dimensional outputs with local structure such as images. The deconvolutional filters namely represent the basic building blocks for generating a realistic image such as edges of different orientations.

2.3.3.4 Inductive biases in different neural network layers

There are two other frequently used neural network layers which we will only briefly explain: recurrent and graph-convolutional NN layers. Recurrent neural network layer applies the same operation at each time (or sequence) step (Figure 2.4c). Graph-convolutional layer applies the same operation at each node or edge in the graph by aggregating the features of the neighbouring edges or nodes (Figure 2.4d).

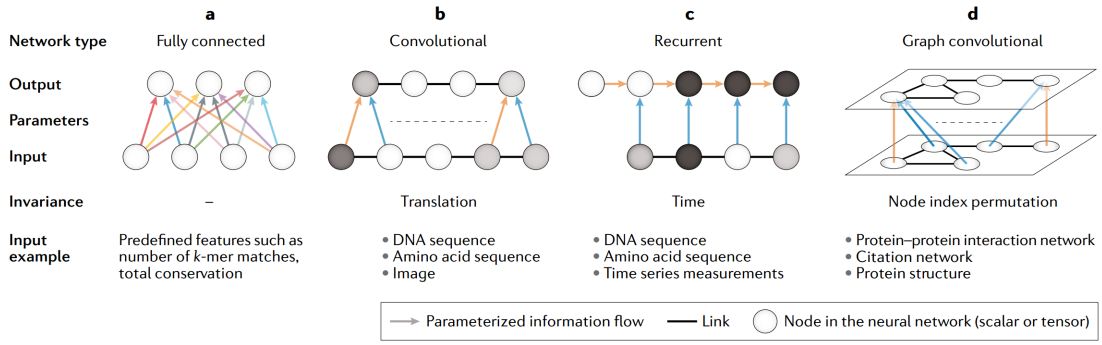


Figure 2.4: Parameter-sharing schemes for different neural network layers. a) Fully-connected layers do not share any parameters across different input features. **b)** Convolutional layers apply the same filter to multiple input patches. That way, they assume that the spatial relationship between input features is local and that these local patterns can occur at differnet positions in the input (translational invariance). **c)** Recurrent neural networks apply the same operation at each time-step. Thereby, they are invariant to the scenario where the relevant sequence of events occurs earlier or later in time. **d)** Graph convolutional layers use the graph connectivity to define the locality similar to convolutional layers. Taken from [3].

The parameter sharing schemes shown in Figure 2.4 are the key reason why neural networks work incredibly well in practice. These layers namely drastically reduce the number of parameters via parameter sharing schemes and model the compositionality of the raw data much better. Motif grammar is one such example of compositionality. Convolutional neural networks allow to effectively model this compositionality via parameter sharing and multiple layers. The parameter sharing scheme is an inductive bias [128] as it constraints the models (bias) in order to generalize better (inductive) on unseen data.

2.3.3.5 Dropout regularization

To add regularization, a dropout layer is frequently used in NNs. In dropout, multiple activation units are randomly set to 0 during model training. This prevents the downstream layers to rely too much on the activation of a single unit as it might be randomly set to 0 by dropout. Instead, the NN has to learn to detect patterns in a redundant manner (i.e. multiple neurons have to detect the same pattern). This constraints the number of patterns that the network can reliably detect and forces the network to focus only on the most important ones.

2.3.3.6 Batch normalization

Batch normalization [129] normalizes the layer activation (subtracts the mean and divides by standard deviation) across multiple samples in the batch and spatial positions for convolutional layers. It is typically used after every layer in the NN. This results in more stable training as the magnitude of the activations throughout the network is

relatively uniform (hence preventing exploding or vanishing gradients). Other normalization layers include weight normalization [130], layer normalization [131] and group normalization [132].

2.3.4 Model is a composition of layers

A deep neural network is a composition of neural network layers. The simplest form of composition (and also the most frequently used one) is a sequence of layers where the next layer takes as input the output of the previous layer $f^{(3)}(f^{(2)}(f^{(1)}(\mathbf{x})))$. Layers can also accept multiple inputs, they can output multiple values, and they can be re-used multiple times in the architecture. Since neural networks can contain many layers, residual skip connections [133] are frequently added to each layer (or layer group): $f^{(2)}(f^{(1)}(\mathbf{x}), \mathbf{x}) = f^{(1)}(\mathbf{x}) + \mathbf{x}$. Thereby, each layer only slightly modifies the activation map of the previous layer (idea similar to boosting [134]). This allows the gradients to flow more freely to the early layers during optimization. It has been shown that residual connections make the objective function more smooth and thereby simplify the optimization [135]. Other skip connections include dense-connections [136] and highway networks [137].

2.3.5 Neural network frameworks

Thanks to the availability of excellent deep learning software frameworks, specifying the model architecture and training the model can be achieved in few lines of code. For example, the following code snippet specifies the architecture from Figure 2.1 and Figure 2.3. Thanks to automatic differentiation performed by these frameworks, training the model can be achieved in a single line of code (final line):

```
import keras.layers as kl
from keras.models import Sequential

# Fully connected model architecture
model = Sequential([
    kl.Dense(3, activation='relu', input_shape=(2,)),
    kl.Dense(2, activation='relu'),
    kl.Dense(1, activation='sigmoid')
])

# Convolutional neural network architecture
model = Sequential([
    kl.Conv1D(2, activation='relu', input_shape=(4, 30), padding='same'),
    kl.MaxPool(6),
    kl.Conv1D(3, activation='relu', padding='same'),
    kl.GlobalMaxPool(),
    kl.Dense(1, activation='sigmoid')
    ])

# Specify optimizer, loss and evaluation metric
model.compile(optimizer='adam',
    loss='binary_crossentropy',
```

```
23      metrics=['accuracy']
24      )
25
26  # Load the dataset
27  x, y = load_dataset(...)
28
29  # Train the model for 10 epochs
30  model.fit(x, y, epochs=10)
```

Listing 2.1: Architecture for Figures 2.1 and 2.3 specified in the Keras deep learning framework `https://keras.io`. Adapted from [3].

2.3.6 Multi-task, multi-modal models

Thanks to the flexibility of NN layers, NN models can simultaneously output multiple values (multi-task, Figure 2.5b) and also integrate multiple input data modalities (multi-modal, Figure 2.5c). Consider the example of simultaneously predicting DNA accessibility in multiple tissues. Motifs important for DNA accessibility can be the same across multiple tissues. Hence, a shared submodel, can be used to detect them (Figure 2.5b, NN on the left). However, their contribution to the DNA accessibility might vary across tissues. Hence, task-specific output layers or submodels should be used to make predictions for each task (Figure 2.5b, NNs on the right). Multi-task models (Figure 2.5b) works better than single task models when similar features have to be detected from raw data for multiple predictive tasks. Namely, by using a common submodel to extract features shared across the tasks, that common submodel is effectively trained on more data (from all the output tasks).

Different data modalities require different feature transformations. For example, when considering the model that predicts TF binding from DNA sequence and DNase accessibility tracks (Figure 2.5c), we can use two different submodels to process the DNA sequence and the DNase accessibility data. The optimal filter sizes and the number of layers might be different for the two data modalities. After applying the modality-specific submodels, the outputs can be concatenated and processed by the final submodel making the final prediction. This type of integration is also called late integration. Late integration is different from early integration where the input data are simply concatenated and processed by a single submodel instead of two separate ones [138].

2.3.7 Transfer learning

Transfer learning is based on the same NN property as multi-task learning. Namely, some features extracted by the early NN layers can be predictive for multiple tasks. However, instead of simultaneously optimizing the model to make prediction for multiple tasks, transfer learning uses an already trained source model (Figure 2.5d top). To re-use the feature extraction performed by the source model in the new model, the parameters of the source model are used to initialize the parameters of the new target model (Figure 2.5d bottom). The target model is trained until convergence while potentially keeping some of the transferred parameters intact. Transfer learning allows to train the target model with

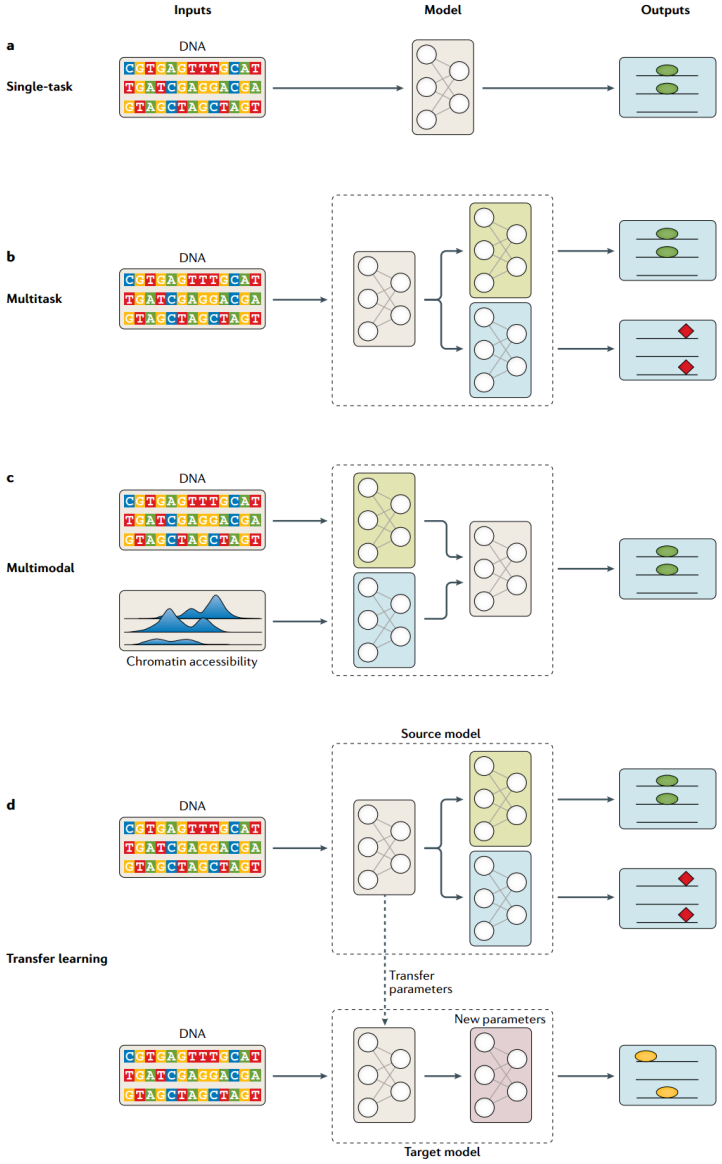


Figure 2.5: Multitask models, multi-modal models and transfer learning. a) Shown is a single-task model predicting the binding of a single transcription factor (green oval). **b)** A multitask model is shown that simultaneously predicts binding for two transcription factors (green oval and red diamond). There are three submodels depicted: a common submodel and two task-specific submodels. **c)** A multi-modal model is shown that takes as input DNA sequence and chromatin accessibility. Each data modality is first processed using a dedicated submodel, and the outputs are processed by the shared submodel. Parameters of all submodels are trained jointly as shown in both parts **b** and **c**. **d)** Transfer learning. Parameters of the original model trained on a large data set (top) are used for initialization for the second model trained on a related task (target task) but with much less data available (bottom). In this example, the first task of the source model is similar to the target task (both are ovals); hence, the transferred submodel may contain features useful for the target task prediction. Taken from [3].

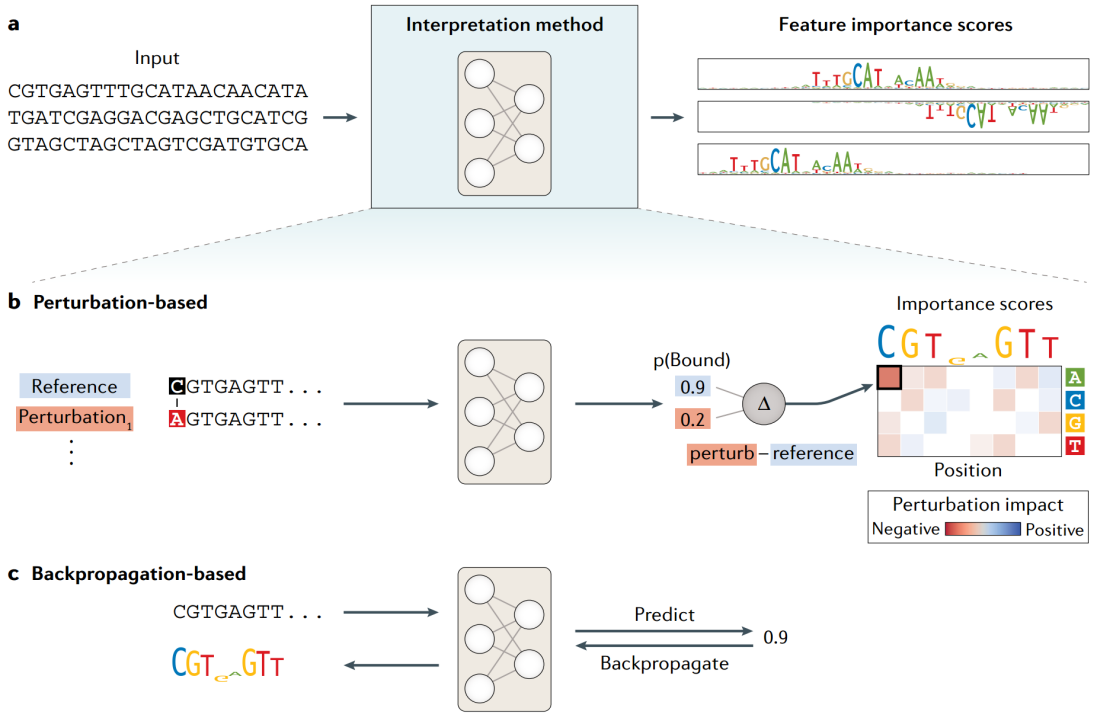


Figure 2.6: Explaining predictions and thereby interpreting the model via feature contribution scores. **A)** Feature contribution scores explain model prediction for a particular input by assigning contribution or importance scores (real valued scalars, can also be negative) to the input features. **B)** To obtain contribution scores for DNA sequence-based models, one can perturb every base in the input (left) and observe the change in model prediction (right). The input base is more important if the prediction drastically changes (large letters, right). **C)** Alternatively, faster and neural-network specific contribution scores using the backpropagation algorithm can be used. These compute the gradient or augmented gradient of the output w.r.t. model prediction to obtain the contribution scores. Taken from [3].

much less data compared to the same mode trained from scratch (i.e. with randomly initialized parameters). Transfer learning has been extremely successful in computer vision thanks to the availability of high-quality models trained on the ImageNet data [139]. In genomics, Kelley *et al* [64] showed that transfer learning is also useful when building models of DNA accessibility in new cell-types.

2.3.8 Model interpretation

Neural networks are often criticised for being 'black-boxes'. They consist of multiple layers, each layer contains multiple parameters and the output of the layers has a non-linear dependency with the output. Hence, directly interpreting the parameters is not particularly meaningful. However, this does not mean that we can not reason about why

a NN is making a certain prediction. Namely, similar to how humans reason about their decisions by providing additional explanations, NNs can also be asked to provide extra explanation of why a certain prediction was made. These extra explanations are manifested in NNs by using feature contribution scores (also called importance or attribution scores). For each input, feature contribution scores assign a score to the input features explaining how 'important' that feature was for making a particular prediction. The notion of 'important' changes between different contribution scores. Here, we will explain three contribution scores: in-silico mutagenesis, input-masked gradient and DeepLIFT.

2.3.8.1 In-silico mutagenesis

We define in-silico mutagenesis (ISM) contribution score (Figure 2.6b) for sequence $\mathbf{s}$, model M at position i in the input sequence as follows:

$$ISM(\mathbf{s}, M, i) = M(\mathbf{s}) - \frac{1}{4} \sum_{b \in \{A,C,G,T\}} \mathrm{M}(\mathrm{mutate}(\mathbf{s}, i, b)) \ , \tag{2.15}$$

where 'mutate' is function mutating sequence $\mathbf{s}$ by replacing the character at position i to character b. Hence, if model prediction of the reference sequence $\mathrm{M}(\mathbf{s})$ is very different than model predictions for other bases $\mathrm{M}(\mathrm{mutate}(\mathbf{s}, i, b))$, then the base is said to be important. While this approach is fairly intuitive, it is quite computationally expensive. Namely, to compute the contribution scores for a 1 kb long DNA sequence, 3001 model predictions have to be made. If each model prediction takes roughly one millisecond on the GPU and if we want to compute contribution scores for 100,000 sequences (say for all potential TF binding sites in the genome), the total compute time will be 83 hours on a single GPU. This is not only computationally expensive but also slows down the iteration cycle of research.

2.3.8.2 Input-masked gradient

Alternatively, backpropagation-based methods can be used. These use backpropagation to 'track-back' the prediction through the NN. They are much more efficient than ISM as they require only a single backpropagation pass through the NN. A single backpropagation pass takes roughly the same amount of time as model prediction ($\sim$1 ms) which is much faster than having to compute 3,000 predictions.

The simplest and easiest to implement backpropagation-based contribution score is the input-masked gradient (Figure 2.6c):

$$\mathrm{inp\_grad}(\mathbf{s}, M, i) = \frac{dM}{ds_{b,i}}|_{\mathbf{s}} \ , \tag{2.16}$$

where b is the observed base at position i for sequence $\mathbf{s}$ (i.e. where $s_{b,i} = 1$). It is termed 'input-masked' because the input sequence is one-hot-encoded and only the gradient at the observed base is used as the contribution score. The gradients at the unobserved bases are ignored. Since automatic differentiation is the integral part of neural network software frameworks, computing these scores can be achieved in a single line of code.

2.3.8.3 DeepLIFT

One issue with ISM and input-masked gradients is the so-called neuron saturation problem. Consider a model that classifies a sequence as positive if it observes a GATA1 transcription factor motif. If there are two GATA1 motifs present in the sequence, we could delete one, and the model prediction would remain the same (the other motif would still be there). Hence, both the ISM and the input-masked gradient would assign low contribution to both GATA1 motifs as none of them individually is necessary. Nevertheless, both are actually important since if we did not have them in the sequence, the model prediction would be very different.

To address this saturation problem, methods like DeepLIFT [110] and integrated gradients [140] were developed. DeepLIFT is defined implicitly by the 'summation-to-delta' principle. It requires the difference in model predictions Δt (or any other activation map unit) for the current input $\mathbf{x}$ and the reference input $\mathbf{r}$ to be explained by the sum of the individual feature contribution scores $C_{\Delta x_i, \Delta t}$:

$$\Delta t = f(\mathbf{x}) - f(\mathbf{r}) = \sum_i C_{\Delta x_i, \Delta t} \ . \tag{2.17}$$

DeepLIFT heuristically defines this decomposition for a broad range of neural network layers. For example, the contribution scores for the fully connected layer

$$f^{(FC)}(\mathbf{x}) = \mathbf{w}^T \mathbf{x} \tag{2.18}$$

are defined as

$$C^{(FC)}_{\Delta x_i, t} = w_i (x_i - r_i) \ . \tag{2.19}$$

The contribution scores for any non-linear activation function f is defined by the definition of the summation-to-delta property as

$$C^{(f)} = f(x) - f(r) \ . \tag{2.20}$$

The contribution scores for the composition of layers $f^{(2)}(f^{(1)}(\mathbf{x}))$ is computed by defining a similar chain-rule as for gradients. Let DeepLIFT multiplier be defined as:

$$m_{\Delta x, \Delta t} = \frac{C_{\Delta x, \Delta t}}{\Delta x} \ , \tag{2.21}$$

where $\Delta x = x_i - r_i$. DeepLIFT multipliers can be seen as a finite version of the partial derivatives. Similar algebraic rules apply to DeepLIFT multipliers as for partial derivatives including the chain rule. Consider a simple 2-layer neural network with D input features $x_1, \ldots, x_D$, H hidden units $h_1, \ldots, h_H$ and a single scalar output t. The DeepLIFT chain rule can be written as

$$m_{\Delta x_i, \Delta t} = \sum_{k=1}^{H} m_{\Delta x_i, \Delta h_k} m_{\Delta h_k, \Delta t} \ . \tag{2.22}$$

By definition, the DeepLIFT contribution scores $C_{\Delta x_i,\Delta h_k} = \Delta x_i m_{\Delta x_i,\Delta h_k}$ and $C_{\Delta h_k,\Delta t} = \Delta h_k m_{\Delta h_k,\Delta t}$ satisfy the summation-to-delta property for any i and k. To prove Equation 2.22, we have to show that the contribution scores $C_{\Delta x_i,\Delta t}$ also satisfy the summation-to-delta property:

$$\sum_{i=1}^{D} C_{\Delta x_i,\Delta t} = \sum_{i=1}^{D} \Delta x_i m_{\Delta x_i,\Delta t} \stackrel{?}{=} \Delta t \ . \tag{2.23}$$

This can be shown by inserting the chain rule from Equation (2.22) into Equation (2.23), swapping the summation order, and using the summation-to-delta property

$$\begin{aligned}
\sum_{i=1}^{D} \Delta x_i m_{\Delta x_i,\Delta t} &= \sum_{i=1}^{D} \Delta x_i \sum_{k=1}^{H} m_{\Delta x_i,\Delta h_k} m_{\Delta h_k,\Delta t} \\
&= \sum_{k=1}^{H} m_{\Delta h_k,\Delta t} \sum_{i=1}^{D} \Delta x_i m_{\Delta x_i,\Delta h_k} \\
&= \sum_{k=1}^{H} m_{\Delta h_k,\Delta t} \sum_{i=1}^{D} C_{\Delta x_i,\Delta h_k} \\
&= \sum_{k=1}^{H} m_{\Delta h_k,\Delta t} \Delta h_k \\
&= \sum_{k=1}^{H} C_{\Delta h_k,\Delta t} \\
&= \Delta t \ .
\end{aligned}$$

■

Hence, DeepLIFT contribution scores can be efficiently computed for the input features of a deep neural network by backpropagating DeepLIFT multipliers and using the DeepLIFT rules for individual neural network layers. By comparing the predictions to the reference predictions, DeepLIFT does not suffer from the saturation issue. We note that choosing the right reference is problem specific. Frequently used references in genomics are obtained by shuffling the input sequence while preserving the di-nucleotide count frequency or setting all values to 0.

2.4 Measuring in vivo TF binding

2.4.1 ChIP-seq

ChIP-seq is a frequently used technique for determining which positions in the genome are bound by proteins (Figure 2.7a) [142, 143, 144, 52, 145]. ChIP-seq for measuring TF-binding is a 'bulk' assay requiring typically 20 millions cells in the biological sample (i.e. cell culture or tissue). In the first step of ChIP-seq, the TF is crosslinked with the DNA (typically by adding formaldehyde to the sample). Crosslinking 'freezes' the interaction between the protein and the DNA. Next, the sample is exposed to ultrasound sonication. Sonication introduces double stranded breaks in the DNA yielding

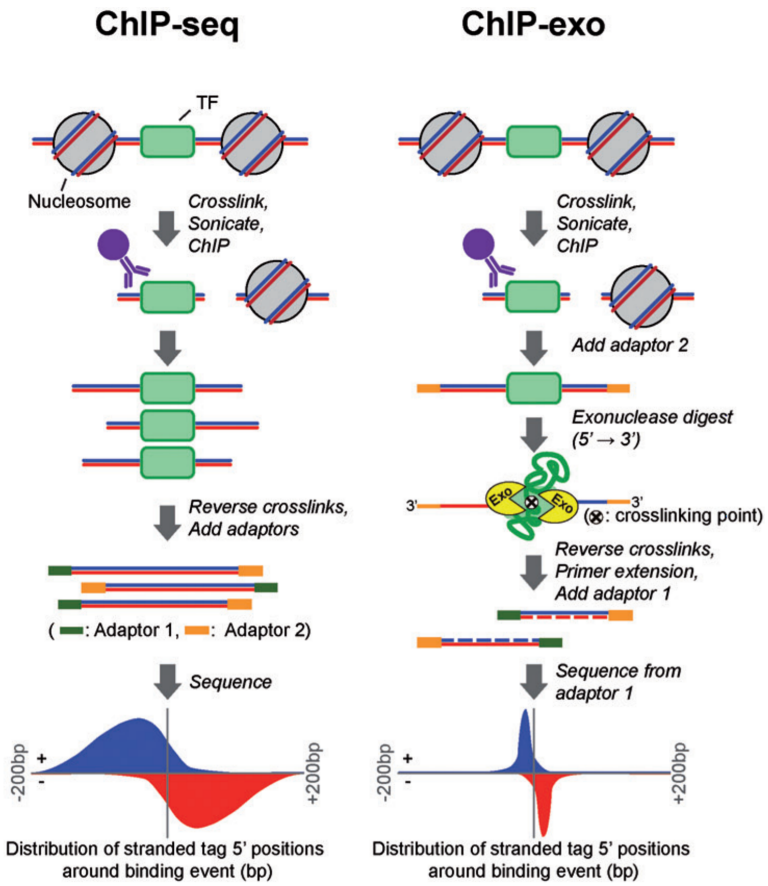


Figure 2.7: Outline of the ChIP-seq and ChIP-exo assays probing in vivo TF binding. Adapted from [141].

short DNA fragments preferably 100-300 bp long. To enrich for short fragments of the DNA that are actually bound by the protein of interest, an antibody specific to the protein of interest (for example CTCF) is used. The DNA sequences captured by the antibody get released by reversing the crosslinking and prepared for sequencing (library preparation). Next the DNA sequences are amplified or cloned using PCR. Finally, the sequencing is performed until ∼10-100 Million reads are sequenced [146]. The whole experimental procedure is performed multiple times, each time on a different biological sample to assure reproducibility.

The sequenced reads (DNA fragments) are aligned or mapped to the genome. Mapping means that the read sequence is compared against the reference genome. The read is assigned to one or multiple positions with the best string match. Reads mapped to multiple positions are typically discarded. Next, if multiple reads align to the same position and have the same start and end point of the fragment (for paired-end data), only one of them is kept. These are namely very likely to be PCR duplicates. Finally, the coverage tracks are produced by summing the number of reads covering at each position. Local regions of high enrichment—peaks—are called using peak callers such as MACS2 [58], GEM [59] or SPP [60].

In almost every step of ChIP-seq, different biases are introduced favoring specific sequences or positions in the genome. For example, sequences with different GC-contents can have a different yield during library preparation or amplification [57], and accessible chromatin regions are more easily fragmented during sonication. To quantify the biases, a similar experiment to ChIP-seq is performed but where either a non-specific antibody is used during the IP step (immunoglobulin mock-ChIP) or the IP step is skipped completely (input-DNA). This control experiment can be thought of as the 'null distribution' of ChIP-seq.

2.4.2 High resolution ChIP-exo and ChIP-nexus

Since the sonication step in ChIP-seq randomly introduces cuts in the DNA, the distribution of the read coverage is spread across ∼100 bp (Figure 2.7b). To increase this spatial resolution, ChIP-exo uses an additional λ-exo-nuclease digestion step in ChIP-seq. Exonuclease digests one strand of the DNA in the 5' to 3' direction. It stops digesting when it encounters DNA protected by the bound protein. Each strand of the DNA (one digested on the positive strand upstream of the protein and one digested on the negative strand downstream of the protein) is then sequenced. A special library construction protocol is used to ensure that only 5' end of the fragment is read during sequencing. After mapping the reads back to the genome, the 5' ends of the reads are located precisely at the positions where the exonuclease stopped. Hence, the resulting footprint (read coverage) can have the resolution even as high as single bp. This allows to call the DNA-protein events more precisely due to higher resolution and also allows to determine whether the protein was directly bound on the DNA or whether it was indirectly bound to the DNA via another protein. Namely, sharp footprints correspond to directly bound proteins and fuzzy footprints correspond to indirectly bound proteins.

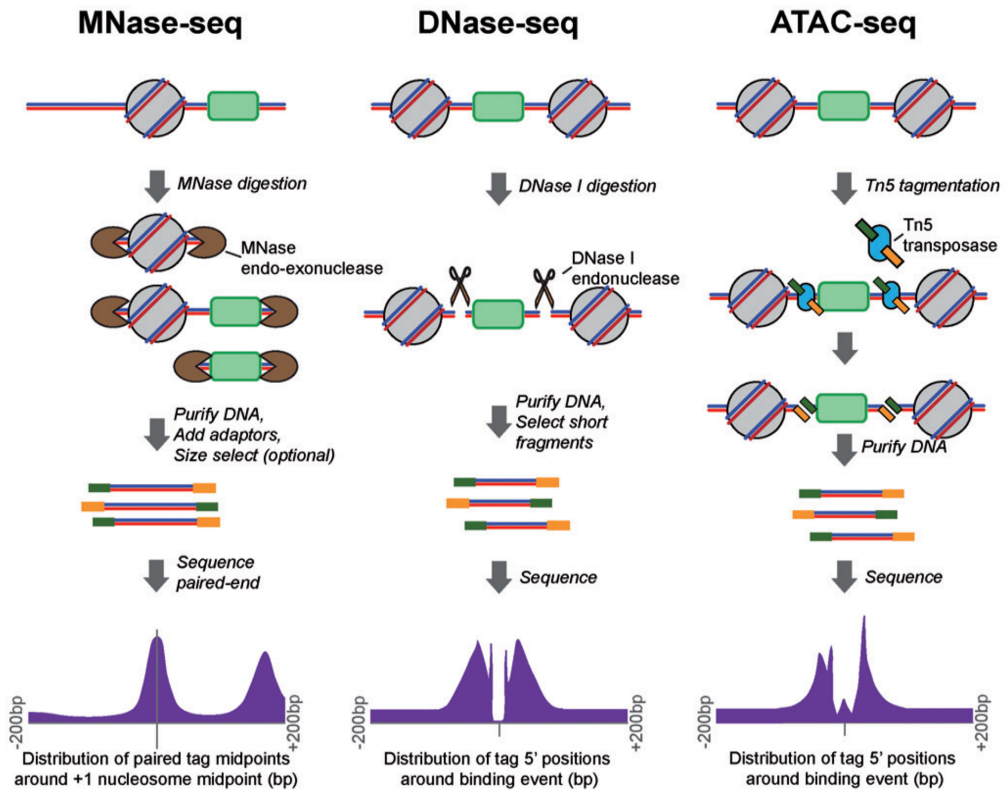


Figure 2.8: Unspecific assays probing the protein-DNA or histone-DNA interactions. Adapted from [141].

An improvement of ChIP-exo termed ChIP-nexus [54] adds random barcodes to the DNA fragments and a self-circularization step. Random barcodes allow to determine the PCR duplicates more accurately. Self-circularization steps improves the efficiency of the protocol by requiring only a single successful ligation per DNA fragment. This means that the same amount of cells (and hence DNA) is required as for the original ChIP-seq experiments while still yielding high-resolution results. By contrast, ChIP-exo requires more cells and is less robust than ChIP-nexus [54].

2.4.3 Other footprinting assays

In addition to the TF-specific assays (ChIP-seq, ChIP-exo and ChIP-nexus), there exist other non TF-specific assays including MNase-seq [147], DNase-seq [76], and ATAC-seq [77]. These measure whether the DNA is potentially bound by *some* TF or wrapped around a nucleosome (Figure 2.8). This is done by treating the sample with a DNase I endo-nuclease for DNase-seq, Tn5 transposase for ATAC-seq or MNase endo-exonuclease for MNase-seq. These enzymes introduce double-stranded breaks in the DNA (ATAC-seq and DNase-seq) or digest the DNA (MNase-seq). Accessible regions are cut or digested more frequently than regions bound by the TF or wrapped around the histones.

Therefore, the footprints in the read coverage contain information about what parts of the DNA were bound by proteins or histones. However, since these assays do not use a specific antibody as done by ChIP-based assays, they do not reveal which protein is bound at a particular location. Instead, they just reveal whether some protein is bound at a particular locus. The advantage of these methods is that they are easier and cheaper to perform, require fewer cells[2], and probe the binding for all TFs simultaneously.

[2] ATAC-seq can be also be applied to single cells [148].

3 BPNet: sequence-to-profile model predicting read coverage tracks from DNA sequence at base-pair resolution

This chapter is based on collaborative work supervised by Anshul Kundaje at Stanford and Julia Zeitlinger at the Stowers Institute for Medical Research. I conceived BPNet including the model architecture, loss function and bias correction. I performed all the analyses presented in this chapter with supervision from Anshul Kundaje. I have received the following help from others. I received help from Johnny Israeli when preparing the ChIP-nexus data for the first BPNet prototype. I received help from Amr Alexandri when applying BPNet to ChIP-seq data and analyzing the results. Ziga Avsec, Melanie Weilert, Jin Lee, and Amr Alexandri implemented the ChIP-nexus data processing pipeline. Ziga Avsec, Melanie Weiert, Amr Alexandari processed all the ChIP-nexus and ChIP-seq data. Robin Fropf, Sabrina Krueger and Khyati Dalal performed the experiments at the Stowers Institute for Medical Research and the Institute's Molecular Biology core facility performed the Illumina sequencing. I prepared the figures together with Julia Zeitlinger. I wrote the chapter with input from Anshul Kundaje and Julia Zeitlinger.

3.1 Motivation

Understanding the cis-regulatory code of the genome is vital for understanding when and where genes are expressed during embryonic development, in adult tissues, and during disease. Extensive molecular profiling efforts have mapped millions of putative enhancers in a wide variety of cell types and tissues [56, 149, 150]. Enhancers contain short sequence motifs that are bound by sequence-specific transcription factors (TFs). However, the exact combinatorial rules (or grammar) by which motifs influence *in vivo* TF binding, enhancer activity, and gene expression remain elusive [14, 15, 16]. The cis-regulatory code of enhancers remains a fundamentally unresolved problem [13].

An important part of the cis-regulatory code is the mechanisms by which TFs bind to DNA in a cooperative fashion [14, 15, 151, 16]. TFs are known to cooperate on DNA directly through protein-protein interactions and indirectly through nucleosomes, but the nature and the degree by which these interactions constrain the distance and orientation between motifs in the genome is still unclear. For example, motifs that are spaced by a multiple of 10 bp may promote cooperative binding since they are found on the same side of the DNA double helix [152, 153, 154], but whether this mechanism is common or specific to certain regulatory contexts is not clear. Likewise, pioneer TFs are

known to affect nucleosome occupancy and chromatin accessibility [155], but how such activity is encoded in the cis-regulatory sequence has not been shown.

A critical hurdle to deciphering motif grammars is the limited ability of current experimental and computational methods to map the precise positions of all motifs that influence TF binding with high confidence *in vivo* [13]. Although the binding specificities of hundreds of TFs have been mapped *in vitro* [156], TF binding *in vivo* depends on the cis-regulatory context such as chromatin accessibility and partner TFs, and thus cannot be predicted without further experimental evidence *in vivo* [151, 13]. Experimental approaches that can map TF binding *in vivo*, including standard ChIP-seq experiments, are however usually limited in their resolution, allowing only a small fraction of TF binding sites to be confidently mapped at nucleotide resolution in the genome. More general assays such as large-scale reporter assays or chromatin accessibility assays measure more downstream biological readouts and provide limited information on the contributing TFs and their cognate binding motifs. Therefore, new integrative approaches are necessary to discover high-resolution motifs and motif grammars affecting *in vivo* transcription factor binding.

Recent advances in ChIP-seq technology allow us to study *in vivo* transcription factor binding at a much higher resolution. Specifically, the near-nucleotide resolution and high specificity of ChIP-exo methods such as ChIP-nexus reveal precise TF binding footprints on the motif of interest [54, 53]. These TF footprints are generated by digesting the 5' ends of the immunoprecipitated DNA with an exonuclease and mapping these stop sites to the genome (Section 2.4.2). In addition to characteristic TF footprints, ChIP-exo/nexus data also contain weaker binding patterns associated with partner TFs [157, 158], but extracting such binding patterns and associated sequence motifs systematically using computational methods is challenging.

Computational methods typically identify putative TF binding sites (peaks) in genomic regions heuristically as local statistical enrichment of ChIP-seq/exo fragments ([58, 60, 159]. Methods that model the positional distribution of ChIP-seq/exo reads [160, 161, 162] improve resolution by being able to distinguish multiple closely spaced binding events. Since these approaches do not model the underlying sequence, motif discovery methods [67, 163] are typically applied post-hoc to identify sequence motifs enriched at putative binding sites. Integrative approaches such as GEM [59] and ChExMix [164] improve spatial precision by learning enriched motifs supported by ChIP-seq/exo read profiles. However, these methods cannot capture the TF binding grammar as they model each binding event independently. Further, these models typically use position weight matrix (PWM) representations for sequence motifs, which make several simplifying assumptions that are violated *in vitro* and *in vivo* [13]. Hence, there is a need for more unbiased computational approaches that can model the predictive relationship between cis-regulatory sequence grammars and high-resolution quantitative *in vivo* TF binding profiles.

Deep neural networks are ideally suited for this problem since they can learn arbitrarily complex predictive patterns *de novo* from unstructured inputs such as DNA sequences to predict complex outputs such as TF binding profiles with high accuracy. Convolutional neural networks (CNNs) have been successfully trained on binarized TF

ChIP-seq profiles (determined by peak calling methods) to classify ∼200 bp genomic intervals as bound or unbound using the underlying DNA sequence [62, 63, 75]. These CNN models learn hierarchical layers of non-linear transformations on *de novo* pattern detectors that collectively encode predictive sequence motifs and grammars. However, the use of low-resolution, binarized training data hinders these models from learning the precise positional influence of motif combinations on the magnitude and shape of *in vivo* TF binding profiles. Recently, Kelley et al. [95] introduced a sequence-to-profile neural network architecture to map regulatory DNA sequence to quantitative profiles of chromatin accessibility (DNase-seq), histone modifications (ChIP-seq) and gene expression (CAGE) at a relatively course resolution of 128 bp bins across the genome. This class of models has yet to be adapted to single-nucleotide resolution profiles such as those obtained from ChIP-exo/nexus experiments.

Here we develop BPNet, a sequence-to-profile neural network that can accurately predict single-nucleotide resolution profiles obtained from ChIP-exo/nexus experiment given DNA sequence. We use BPNet to model ChIP-nexus profiles of pluripotency TFs Oct4, Sox2, Nanog and Klf4 in mouse embryonic stem cells (mESCs) [108]. BPNet accurately predicts ChIP-nexus binding footprints of the four TFs at single-nucleotide resolution on par with replicate experiments. In the following chapter, we will demonstrate how this model can be interpreted with feature importance scores to discover motifs and learn the regulatory grammar rules of TF binding.

3.2 BPNet

3.2.1 Architecture

BPNet is a sequence-to-profile convolutional neural network that uses one-hot-encoded DNA sequence (A=[1,0,0,0], C=[0,1,0,0], G=[0,0,1,0], T=[0,0,0,1]) as input to predict single nucleotide-resolution read count profiles. We note that the read count profiles are strand-specific for ChIP-nexus TF binding experiments. The architecture of BPNet can be compartmentalized into two parts: the body and multiple task-specific output heads. The separation of the BPNet body and head components makes the architecture more flexible, allowing the features learned in the body to be used for the prediction of various output data sets.

The body of BPNet consists of a sequence of convolutional layers with residual skip connections [133]. The first convolutional layer uses a wide filter of 25 bp to scan the 1 kb region for relevant sequence motifs. This layer is then followed by 9 dilated convolutional layers (filter width 3) where the dilation rate (number of skipped positions in the convolutional filter) doubles at every layer. To preserve the base-pair resolution, pooling is not used in the architecture. Thanks to a large receptive field achieved by dilated convolutions, the BPNet body is designed to not only detect sequence motifs, but also to analyze these motifs in a larger sequence context within regions of at least 500 bp. Thus, any sequence patterns that shape the read count profiles in ChIP-nexus, including motif combinations, should be learnable by BPNet. The output of the final

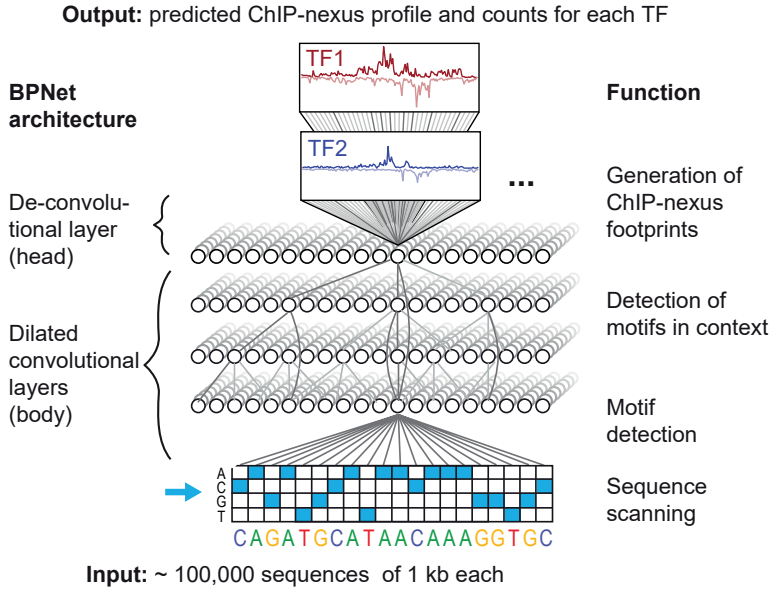


Figure 3.1: A convolutional neural network (BPNet) is trained to predict the number of aligned reads measured by ChIP-nexus for all TFs simultaneously at each nucleotide position from 1 kb DNA sequence for each strand.

convolutional layer within the BPNet body (also referred to as the bottleneck activation map) then serves as input for TF-specific output heads.

We use $2T$ output heads where T is the number of predicted tracks. For each track, we use two output heads: a deconvolutional layer (width=25, typical ChIP-nexus footprint width) predicting the probability distribution of read counts within the sequence (i.e. normalized profile) for each strand and a fully connected layer predicting the total number of counts aligned to the input sequence at each strand. This decoupling allows the network to learn the binding profiles even when the total number of read counts is not representative of the binding strength. The training occurs for all TF ChIP-nexus experiments together in a multi-task fashion.

3.2.2 Loss function

Let $\mathbf{k}^{obs}$ be the L dimensional array of observed read counts for a particular strand and a particular task along the sequence of length L. Let $\mathbf{p}^{pred}$ be the L dimensional array of predicted probabilities along the sequence, such that $\sum_i p_i = 1$ and let $n^{obs} = \sum_i k_i^{obs}$ be the total number of observed counts and n^{pred} the total number of predicted counts for the sequence. BPNet is trained using the following loss function for one particular sequence, strand and task:

$$Loss \;=\; -\log p_{mult.}(\mathbf{k}^{obs}|\mathbf{p}^{pred}, n^{obs}) \;+\; \lambda(\log(1 \;+\; n^{obs}) \;-\; \log(1 \;+\; n^{pred}))^2 \quad . \quad (3.1)$$

First term is the multinomial negative log-likelihood of observed base-pair read counts conditioned on the predicted probabilities and the total number of observed counts. The second term is the squared error loss for predicting total number of counts in the sequence. The total loss function is the sum of individual loss functions across all sequences, all tasks and both strands.

The key question is how to choose a good value for the hyper-parameter λ. In the appendix Section A.3, we show that if $\lambda = \frac{1}{2}\bar{n}^{obs}$, where $\bar{n}^{obs}$ is the average number of total counts in our training set, the profile loss and the total count loss will be roughly given equal weight. As we will see later, we will use $\lambda = \frac{\alpha}{2}n^{obs}$ with $\alpha < 1$ to put less weight on accurately predicting total counts.

3.2.3 Controlling for biases

As described in Section 2.4, the experimental assays such as ChIP-seq (and to a small extend also ChIP-nexus) have certain biases. These biases can be experimentally measured by performing control experiments such as input-DNA for ChIP-seq and PAtCh-CAP for ChIP-nexus [165]. To prevent the sequence-to-profile model from learning these non-informative bias signals, we try to explain the target experimental track using both, the sequence-based model predictions and the control experiment track

$$\mathbf{y}_{pred} = \mathbf{f}_{model}(\text{seq}; \mathbf{w}) + \mathbf{f}_{ctl}(\text{ctl}; \mathbf{w}_{ctl}) \,, \tag{3.2}$$

where $f_{ctl}(\text{ctl}; \mathbf{w}_{ctl})$ is some transformation of the control track with the requirement that $\mathbf{f}_{ctl}(\text{ctl}; \mathbf{w}_{ctl}) = 0$ if the control track is 0 (i.e. bias not present). For the total count prediction head, $f_{ctl}(\text{ctl}; \mathbf{w}_{ctl})$ is simply $w_{ctl}\log(1+n_{ctl})$, where n_{ctl} is the total number of reads from the control experiment in the modeled local region. For the profile prediction head, $f_{ctl}(\text{ctl}; \mathbf{w}_{ctl})$ is a weighted sum of i) the raw counts and ii) a smoothed version of the raw counts using a sliding window sum of 50 bp. We use the sliding window to deal with typically very sparse data from the control experiment. During model training, the parameters of $f_{ctl}(\text{ctl}; \mathbf{w}_{ctl})$ are also trained to best explain the output using the control track. We note that this framework also allows to easily integrate multiple control tracks as well as control tracks predicted from sequence using a bias model learned on other data such as deproteinized genomic DNA for DNase-seq [166].

3.2.4 Training

We used ChIP-nexus profiles of Oct4, Sox2, Nanog and Klf4 TFs in mouse embryonic stem cells (ESCs) to train and evaluate BPNet (≈100 million reads per TF, pooled from multiple replicates). The ChIP-nexus datasets exhibited high replicate concordance, signal-to-noise ratios and strong overlap of peaks with corresponding ChIP-seq experiments targeting the same TFs. PAtCh-CAP experiment was used as the control. For each TF, the ChIP-nexus profile coverage is defined by the number of reads with the 5' end aligned to a specific position and strand. Regions of enrichment (peaks) were identified using MACS2 [58] on smoothed read densities to obtain a ChIP-seq-like signal (Section A.1.4). We restrict model training and evaluation to 1 kb regions around the

150,908 summits in autosomes that ranked consistently across replicates genomic regions as measured by the irreproducible discovery rate (IDR, [167]) threshold of 0.05. Regions from chromosomes 2,3,4 (20%) were used as the validation set for hyper-parameter tuning. Regions from chromosomes 1,8,9 (20%) were used as the test set. The remaining regions were used for model training.

We implemented and trained all neural network models in Keras (v2.2.4) [168] (TensorFlow backend v1.6) using the Adam optimizer [121] (learning rate = 0.004) and early stopping with patience of 5 epochs.

We chose mouse embryonic stem cells (ESCs) since they have been extensively used as a model system to identify and study cis-regulatory code. ESCs have been well studied due to their pluripotent potential to give rise to all cell types and their promise in regenerative medicine [169, 108]. Maintaining the pluripotency of these cells requires the pluripotency TFs Oct4, Sox2, Nanog, and Klf4 [170]. These pluripotency TFs have been extensively studied using genetics, genomics, proteomics, and biochemistry. Despite these efforts, the binding specificity of Nanog, as well as the ability of these TFs to bind cooperatively with each other in the genome remains incompletely understood. Nevertheless, ESCs are extremely well-studied and therefore serve as an excellent system to explore the capabilities of new computational methods to decipher cis-regulatory code.

3.3 Results

3.3.1 BPNet can predict ChIP-nexus data in mouse ESCs at nucleotide resolution

We evaluated the predictive performance of BPNet in three ways: by visualizing observed and predicted profiles, treating the profile prediction as a binary peak classification and inspecting the Spearman correlation between the predicted and observed total counts. We compared our approach to replicate experiments which roughly represent an upper-bound on the best possible predictive performance.

We first visualized the observed and predicted profiles in putative enhancers located on the test chromosomes. Intriguingly, the profiles of the observed and predicted read counts were noticeably similar, as shown by the Sall1 [171, 172], Zfp281 [173], and Lefty1 [174] enhancers (Figure 3.2). We observed that the predicted summits of the read counts for both strands occur at the same locations as the observed summits and that the model also captures the more subtle aspects of the profiles quite well.

To quantify the ability of the model to distinguish positions with high read counts from lower read counts within each ChIP-nexus profile, we evaluated the model predictions against binary labels at each position in the test set ChIP-nexus profiles. Positions with more than 1.5% of the reads were considered as positive class and positions with less than 0.5% of reads in the 1kb region were considered as negative class. For each TF, replicate experiments were divided into two groups with approximately equal numbers of sequencing reads. Read counts from one group was used as ground truth and the read counts from the other group were treated as a predictor similar to BPNet. The

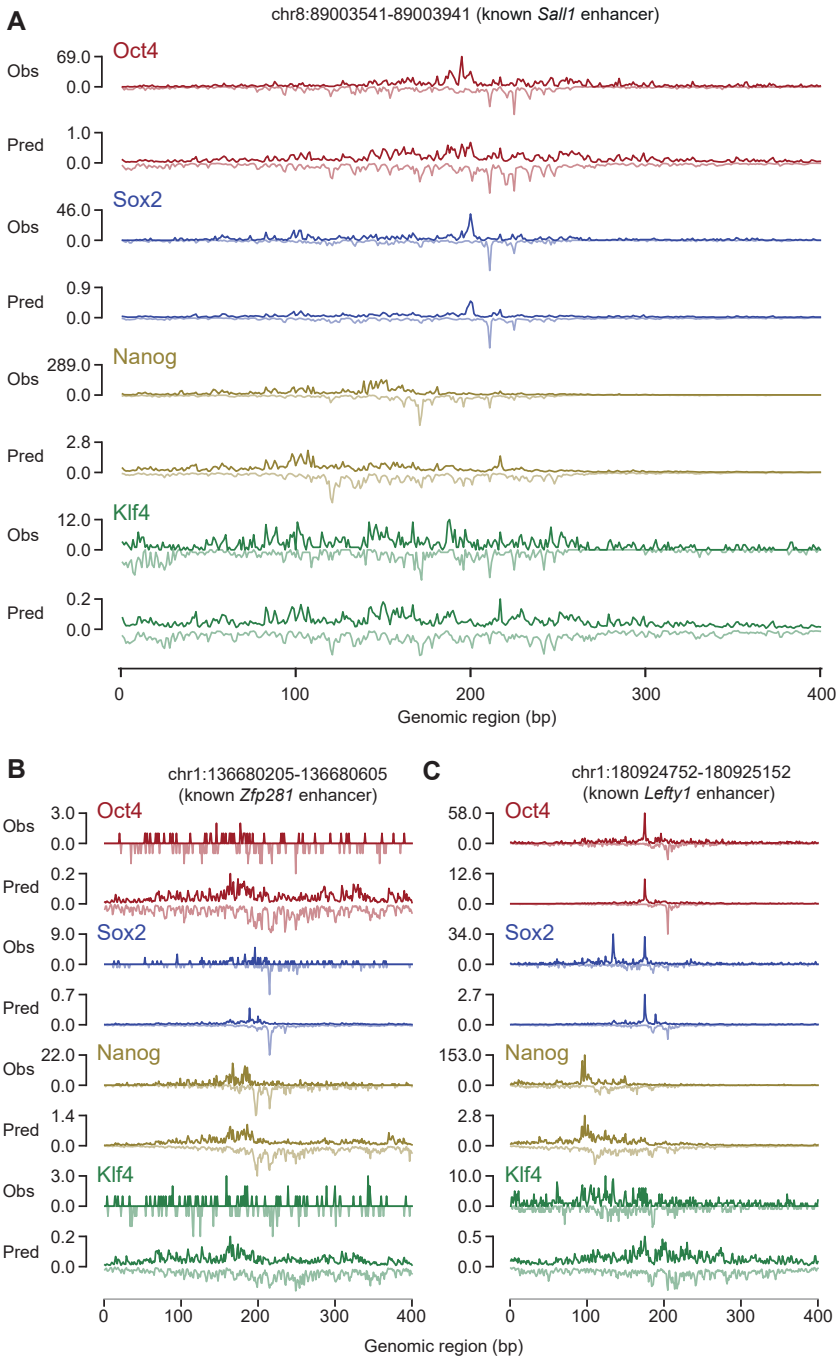


Figure 3.2: A-C Observed and predicted ChIP-nexus read counts for known enhancers located in the held-out (test) chromosomes.

roles of the replicate groups were then swapped and the final predictive performance was averaged across both scenarios.

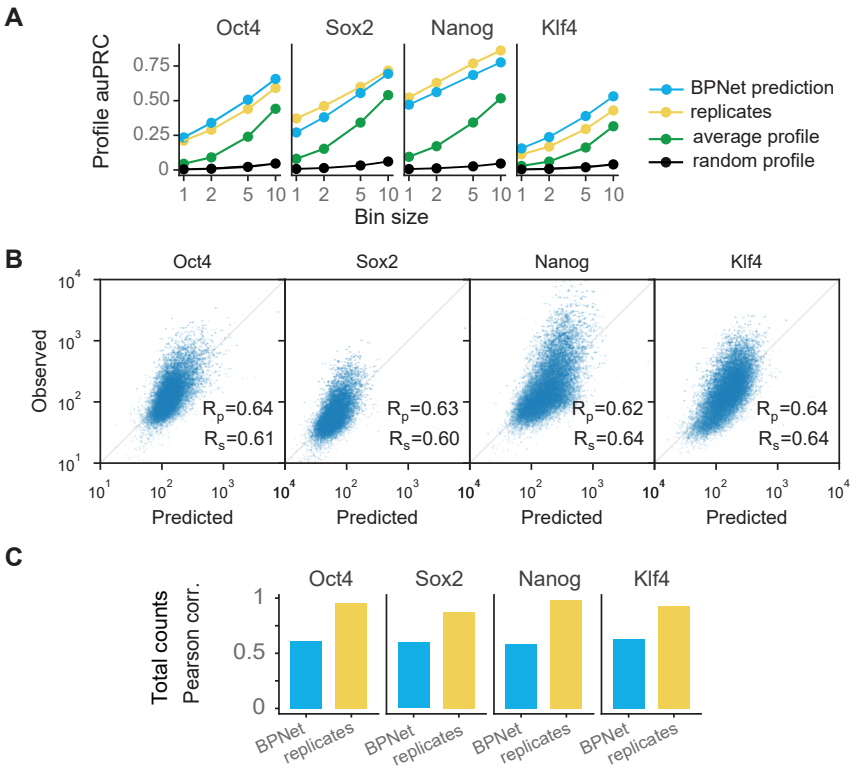


Figure 3.3: Predictive performance of BPNet in the held-out test chromosomes A) BPNet predicts the profile shape at replicate-level accuracy as measured by the area under prediction-recall curve (auPRC) for predicting the read count summits at different resolutions (from 1 bp to 10 bp windows) in held-out chromosomes 1, 8 and 9 (top, Section 3.2.4). **B,C)** Total counts in the 1kb regions centered at the peak summits in the region can be predicted (bottom, blue bar) at a decent accuracy level as measured by Spearman correlation but doesn't surpass replicate performance (bottom, orange bar).

We found that BPNet predicts the profile summits of the ChIP-nexus data remarkably well, comparable even to technical replicates (Figure 3.3A). This holds also true when the resolution of the summit detection window is gradually decreased from 1 bp to 10 bp. We conclude that BPNet predicted the ChIP-nexus profiles for all four TFs with high accuracy.

Next, we used Spearman correlation to quantify the similarity between measured and predicted total counts of profiles in the test set (Figure 3.3B). The total number of reads predicted in these regions is not as consistent as between technical replicates (Figure 3.3C), but Spearman correlations between observed and predicted read counts are still high for all TFs, with an average of Rs = 0.52. This indicates that parameters that determine overall transcription factor binding levels, such as chromatin context, may not be fully captured by BPNet, while the shape of the binding profiles can more

easily be predicted from the 1 kb sequences. Hence, the profile loss function should be given more weight than the total counts loss (Section 3.2.2).

3.3.2 Hyper-parameter investigation

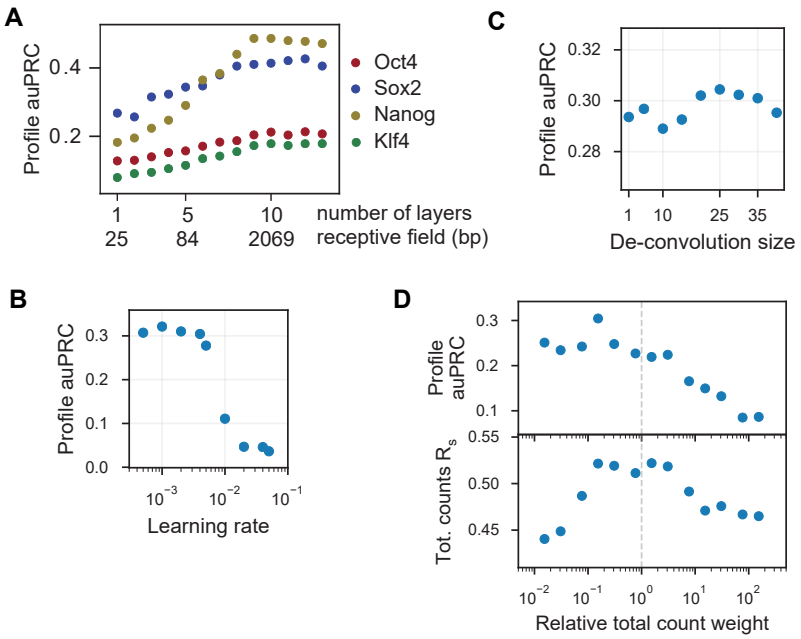


Figure 3.4: Influence of BPNet hyper-parameters. A) More convolutional layers (x-axis) yield increasingly more accurate profile shape predictions (auPRC) on the validation set chromosomes 2-4 (y-axis). **B)** Model is robust across a wide range of learning rate choices. **C)** Deconvolutional layer slightly improves the performance compared to a point-wise convolutional layer (deconvolution size=1) as used in Basenji [95]. **D)** Relative total count weight α in the BPNet loss function (Equation 3.1) parameterized as $\lambda = \frac{\alpha}{2} n^{obs}$. See also Sections 3.2.2 and A.3.

To better understand how the different components of the BPNet model architecture led to the high predictive accuracy, we analyzed the predictive performance for a range of different hyper-parameters using the data from validation chromosomes 2, 3, and 4 (Figure 3.4). Most notably, we find that the large number of layers is key to the good predictive performance and that some TFs such as Nanog benefit from the increase in layers more than others (Figure 3.4A). This indicates that the learned sequence patterns required to predict ChIP-nexus profiles span over larger sequence regions and may be relatively complex, especially for Nanog.

We also observed that the architecture was robust to a wide range of learning rates (Figure 3.4B). This means that extensive hyper-parameter tuning is not necessary to achieve good performance. As expected, we also found that using the deconvolutional layer for the output head helps with the profile shape prediction compared to using a pointwise convolution (Figure 3.4C). Giving the profile prediction more weight over

the total count prediction increased the overall performance while maintaining the same accuracy for predicting the total counts (Figure 3.4D). This supports the notion that the profile contains more information than the total number of counts and is primarily only influenced by the local sequence. We note that multi-task training of multiple transcription factors was not necessary for the high predictive performance since training on single data sets resulted in similar predictive accuracy (average auPRC 29% versus 32%, average Spearman correlations 0.53 versus 0.52).

3.3.3 Bottleneck activation layer is predictive for enhancer activity

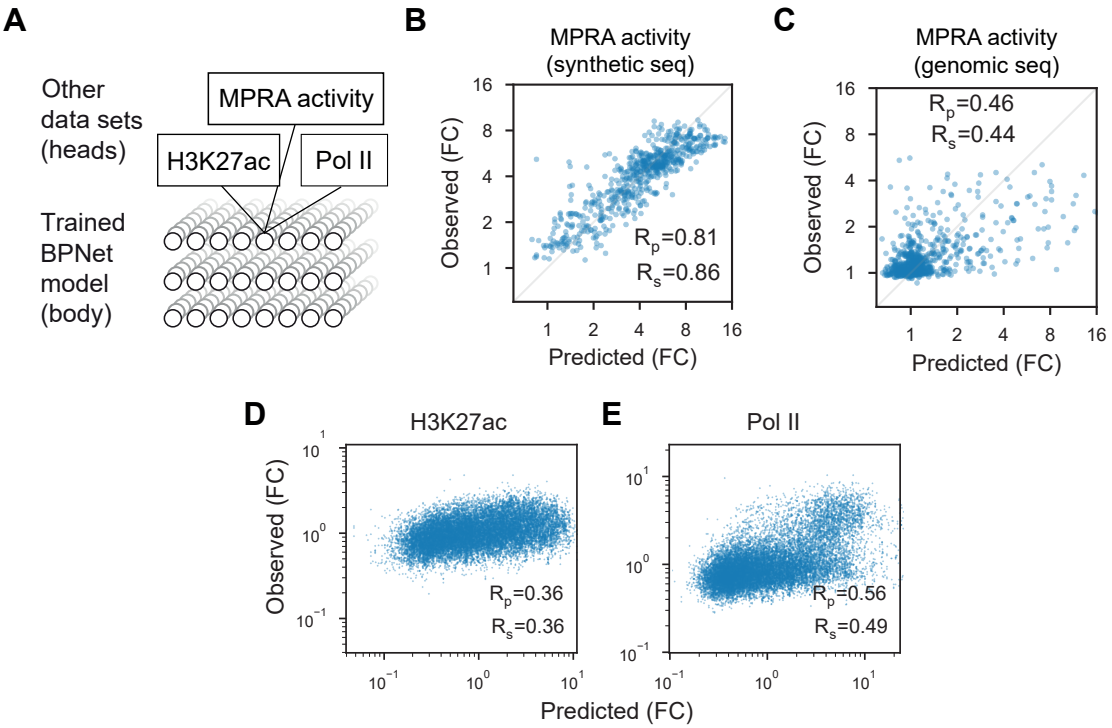


Figure 3.5: **A)** The trained BPNet body can be combined with different heads to predict other data sets relevant to the trained data. **B-E)** Here the BPNet body was used to predict the expression of massively parallel reporter assays (MPRA), and ChIP-seq data of H3K27ac and Pol II, which are markers for enhancer activity. In all three cases, the predictions show reasonable correlations with the observed data (both measured as fold-ratio (FC))

Our results demonstrate that transcription binding can be predicted from sequence. It is however unclear to what extent binding predicts enhancer activity. To test whether the learned features of BPNet helped predict enhancer activity, we took the bottleneck activation map of the trained BPNet model as input and used it to predict the enhancer activity as measured by i) MPRA from [175] and ii) H3K27ac and Pol II ChIP-seq data in ESC [176] (Figure 3.5A).

The authors of [175] kindly provided us both MPRA datasets: a set of synthetic sequences (SYN) and a set of genomic sequences (GEN) with their corresponding expression values measured across multiple replicates. We used the same barcode sequence AAAGACGCG for all the sequences (150 bp long) and padded them to random sequence on both sites to reach the length of 1 kb as required by BPNet. We used the log2 average normalized expression value as described in [175] as the response variable.

After computing the bottleneck activation values (1000 X 64 matrix) of BPNet from sequence, we trimmed the bottleneck activation values to the 80 bp interval [444, 524) corresponding to the typical 80 bp length of the probed sequence (yielding an 80 X 64 activation map). Next, we used max pooling with pool size (and stride) of 15 bp and 'same' padding (i.e. padded the input with 0 to reach the divisible length of 90 bp). We flattened the resulting matrix into the final feature vector. We trained random forests models [116] with 100 trees (`RandomForestRegressor` from scikit-learn v0.20.2 with `n_estimators=100`) in 5-fold cross validation to obtain the out-of-fold predictions (using scikit-learn `cross_val_predict`).

We observed a very high predictive performance for the synthetic sequences (Figure 3.5B) and rather low performance on the genomic sequences (Figure 3.5C). Low predictive performance on the genomic sequences can be explained by the fact that most sequences were very lowly expressed likely due to poorly chosen sequences. Nevertheless, we observed a higher predictive performance distinguishing high vs lowly expressed sequences (defined as the fourth expression quartile vs first expression quartile) compared to the lsgkm-SVM model used in [175] (auROC of 0.83 vs 0.75).

To assess the predictive performance on the ChIP-seq experiments of H3K27ac and Pol II, we used all ChIP-nexus peaks of Oct4, Sox2, Nanog and Klf4 (150,908 in total) and split them into train, validation and test intervals according to the chromosome as done for BPNet. For each ChIP-nexus peak, we extracted the sequence from the 1 kb interval centered at the ChIP-nexus peak summit and obtained the predictions for the bottleneck activation of BPNet. As a response variable, we mapped both replicates of wild type (WT) H3K27ac and WT Pol II ChIP-seq from GEO accession GSE98063 to the 2 kb region centered at the ChIP-nexus peak summit [176]. Next, we transformed the total number of aligned reads to the final response variable as follows: $y = \log_{10}(1 + \text{counts})$.

To train a multi-task model predicting the response variables (H3K27ac and Pol II) from the bottleneck activation we used global average pooling followed by a 2 fully connected (FC) layers with 64 and 2 activation units. ReLU activation was used for the first FC layer. We used the Adam optimizer [121] using default parameters in Keras (v2.2.4, learning rate=0.001), mean squared error loss function and batch size of 512. The model was trained using the data from train chromosomes (as for BPNet training) for at most 100 epochs with early stopping patience of 5 epochs using the evaluation data from validation chromosomes (2, 3 and 4). Using this strategy, we were able to predict H3K27ac levels with a Pearson correlation of 0.36 and Pol II with a Pearson correlation of 0.56 (Figure 3.5). Altogether, these results suggests that enhancer activity can be predicted from BPNet transcription factor binding predictions at cis-regulatory sequences.

3.3.4 Improving binary peak classification by simultaneously predicting the profile

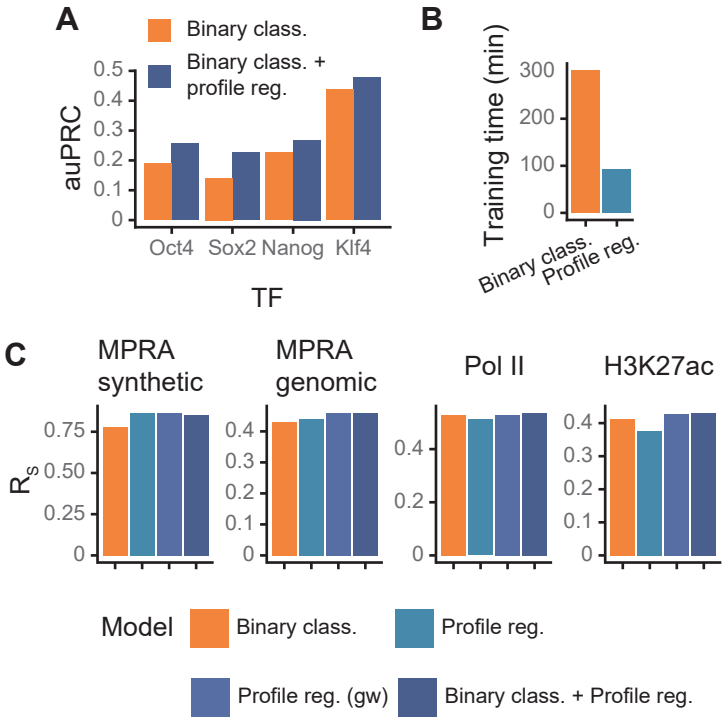


Figure 3.6: Binary classification vs profile regression. A) Predictive performance of the binary classification models predicting the presence or absence of ChIP-nexus peaks from 1 kb DNA sequence evaluated across the held-out (validation) chromosomes 2, 3 and 4. Model trained to only classify the sequences is shown in orange and the model trained to also predict the ChIP-nexus profiles is shown in blue. **B)** Training time of the binary classification model trained genome-wide and the sequence-to-profile model (BPNet) trained in ChIP-nexus peaks. **C)** Predictive performance of a model using the bottleneck activation from 4 different models as features to predict transcriptional activity as measured by MPRA or ChIP-seq Pol II and H3K27ac. Profile reg. (gw) model was trained genome-wide in contrast to the default BPNet model denoted here by Profile reg. which was only trained in ChIP-nexus peaks. All binary classification models were trained genome-wide.

A frequently used approach for training deep learning models is to treat the TF binding prediction as a binary classification problem [62, 63]. In this approach, the training examples are sequences extracted from contiguous bins in the genome and the sequence label is positive if a TF binding peak overlaps the region (and negative otherwise). The benefit of such an approach is two fold. First, the assay-specific biases are already accounted for in the peak-calling process. Second, the resulting machine learning task–binary classification–is well understood. Hence the standard loss function such as binary cross-entropy and the standard evaluation metrics such as the area under precision-recall

curve (auPRC) can be used. However, compressing the observed data into binary labels discards information about the strength of binding (number of reads) and specific details on the co-binding mode (profile shape) present in the ChIP-nexus read coverage track.

To investigate the benefit of training the model on the ChIP-nexus read coverage tracks as performed by BPNet to the frequently used binary classification, we modified the BPNet architecture and replaced the output heads performing profile regression with the output heads performing binary classification. These consisted of weighted global average pooling using spline transformation [2] and a dense layer followed by sigmoid activation. We trained the model on contiguous bins of 50 bp (flanked to 1 kb) spaced across the genome and labeled them as positive if the central 200bp of the bin overlapped the peak as called by MACS2. The predictive performance on the held-out validation chromosomes (2, 3 and 4) was 25% auPRC in average across the 4 TFs after tuning the optimal learning rate (Figure 3.5A). We also observed that training the binary classification model genome-wide took 3 times longer to train (Figure 3.5B) than BPNet, which is trained only on 150,908 peaks. To ensure that the dilated convolutional layers are also appropriate for binary classification, we trained and evaluated the Basset [64] and factorized Basset [177] architectures. After tuning the dropout rate with random search, we obtained a slightly lower auPRC of 24% for both models, suggesting that our original architecture with dilated convolutions was also a good fit for binary classification. Next, we asked whether the predictive performance of the binary classification model could be improved by adding another output head predicting the stranded ChIP-nexus profile as originally done by BPNet. Indeed, the classification performance increased for all TFs yielding an average of 31% auPRC (Figure 3.5A). We conclude that the read coverage track indeed provides additional information not captured by the binary labels and allows learning more informative features in the shared NN layers.

When the bottleneck layer activation was shared, the information about TF binding learned from predicting the profile increased the binary classification performance. We therefore asked whether a similar increase could be observed for predicting the transcriptional activity as measured by MPRA or ChIP-seq of Pol II and H3K27ac (Figure 3.5C) from the bottleneck activation. We compared the bottleneck activation of three models trained genome wide performing either binary classification, profile regression or both, and the bottleneck activation a profile regression model trained only in ChIP-nexus peaks (Figure 3.5C). The bottleneck activation of models predicting the profile yielded a higher predictive accuracy for both MPRA datasets (synthetic and genomic sequences) compared to the bottleneck activation of the binary classification model (Figure 3.5C). This is expected since predicting the transcriptional activity requires knowledge about the subtle variation of TF binding and the combinatorial TF grammar both of which the profile regression model had to learn in order to successfully predict subtle variations of ChIP-nexus profiles. On the lower resolution data of transcriptional activity as measured by Pol II and H3K27ac ChIP-seq, we observe that the bottleneck activation of the binary classification model performed similar to the bottleneck activation of the profile regression model (Figure 3.5C). Models trained genome-wide showed a better performance on Pol II and H3K27ac predictive tasks since training the model only in peaks might lack sequences containing repressive motifs. Altogether, we observe that learn-

ing to predict the ChIP-nexus profiles yields more informative features for the activity prediction compared to binary classification.

3.3.5 BPNet is directly applicable to ChIP-seq

The BPNet model can be readily applied to ChIP-seq, since it does not make any modeling assumptions specific to ChIP-nexus profile shape. The major difference of ChIP-seq compared to ChIP-exo/nexus is that the 5' ends of the reads mapping to a particular strand are dispersed in a 100-200 bp window around the peak whereas the ChIP-exo/nexus peaks frequently achieve single-base resolution. To demonstrate that BPNet is also applicable to ChIP-seq, we performed ChIP-seq for 3 out of 4 previously studied TFs (Oct4, Sox2 and Nanog). We processed the data using the ENCODE ChIP-seq pipeline and generated the strand-specific 5' read count tracks as for ChIP-nexus. We used the same architecture structure for ChIP-seq as for ChIP-nexus and determined the optimal hyper-parameters using a random search. We observed that the BPNet model for ChIP-seq overall requires almost the same hyper-parameters as for ChIP-nexus. The only hyper-parameter that differed was the increased width (50) of the deconvolutional layer (25 was optimal for ChIP-nexus). Similar to the ChIP-nexus control experiment PatchCap, we used the ChIP-seq input control experiment using an unspecific antibody in the loss function to control for the biases (Section 3.2.3). We also added data augmentation (genomic intervals shifted uniformly from [-200, 200] bp with random reverse complementation). This is more important when ChIP-seq data are trained on peaks only since the shape of the profiles will be fairly constant hence a constant model can already fit the data well.

To gain more intuition about the prediction quality of BPNet compared to replicate experiments, we first investigated the known Zfp281 and Lefty1 enhancers as done before for ChIP-nexus. Since the model evaluation was performed in the peak regions, we added data augmentation (genomic intervals shifted uniformly from [-400, 400] bp with random reverse complementation) to prevent a simple constant model to show good performance. We observed that the predicted profile shapes indeed resemble the smoothed ChIP-seq signal (averaging sliding window of 50bp, Figure 3.7A,B).

To evaluate the predictive performance of the ChIP-seq BPNet model, we performed a similar analysis as for ChIP-nexus with the difference that we assessed the quality of profile shape prediction using the multinomial log-likelihood. We found that BPNet outperformed the smoothed replicate experiments in terms of profile shape prediction on almost all TFs except Nanog where both performed similarly. The total count predictions of BPNet were not as good as for the replicate experiments. As already discussed for BPNet trained on ChIP-nexus data (Section 3.3.1), the total counts can be influenced by DNA accessibility which depends on a larger chromatin context. Altogether, we conclude that BPNet is also applicable to ChIP-seq where it also shows high predictive accuracy on par with replicate experiments.

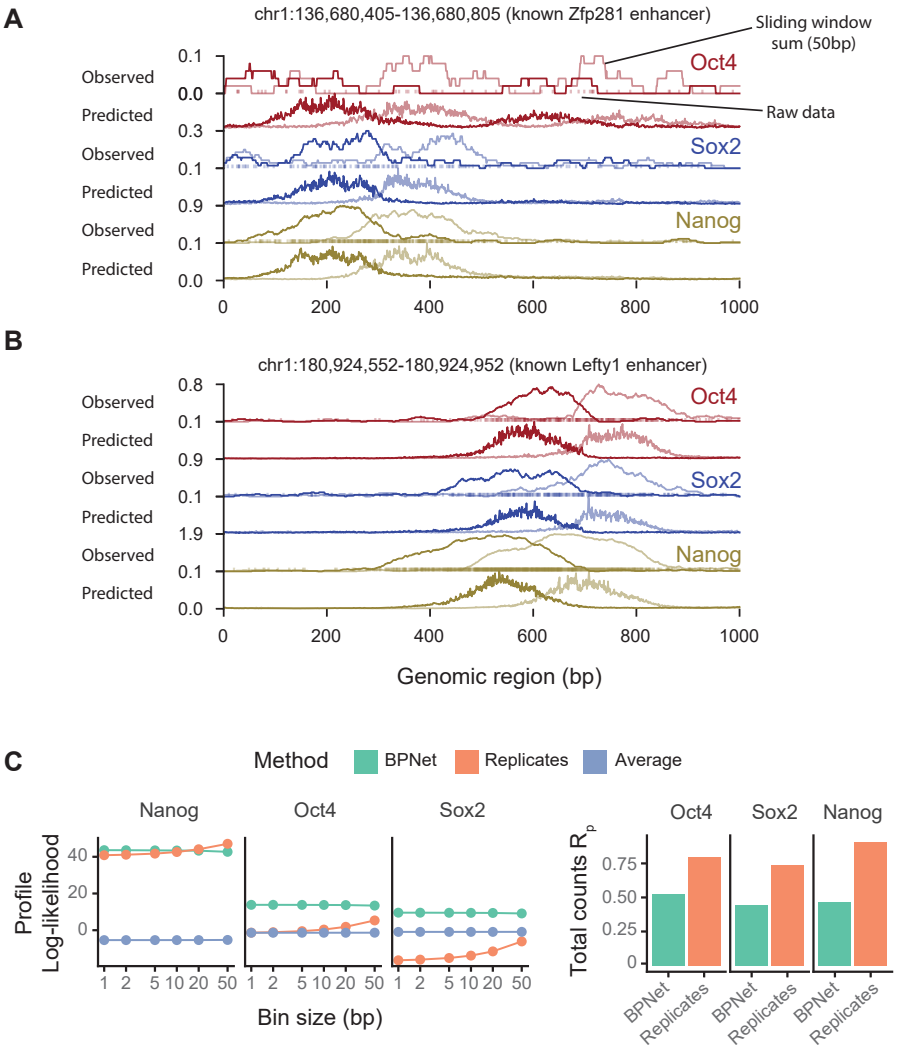


Figure 3.7: BPNet applied to ChIP-seq data. A,B) Observed and predicted read counts mapping to the forward strand (dark) and the reverse strand (light) for the Zfp281 and Letfy1 enhancers located on the held-out (test) chromosome 1. For the observed read counts, a sliding window of 50 bp was used to smooth the raw 5' end read counts (line). Raw counts are shown as points on the bottom at y=0. **B)** BPNet predicts the ChIP-seq profile shape better than the replicates. Multinomial-log likelihood conditioned on the observed number of total counts was used to evaluate the profile shape quality at different resolutions (from 1 bp to 10 bp windows) in held-out chromosomes 1, 8 and 9 (left). The log-likelihood of 0 corresponds to the constant model. Total counts in the 1kb regions centered at the peak summits in the region can be predicted (green) at a decent accuracy level as measured by Spearman correlation but doesn't surpass replicate performance (orange).

3.4 Discussion

In this chapter, we developed BPNet, a novel predictive model BPNet using convolutional neural networks. BPNet can predict the ChIP-nexus read counts at high accuracy while controlling for assay-specific biases. We demonstrated that the ChIP-nexus profile shape is informative and should not be discarded as done by current state-of-the-art approaches [62, 63, 75, 95]. Since we were able to readily apply BPNet to ChIP-seq with a minimal change in hyper-parameters, it could hence be also applied to other genome profiling assays exhibiting footprints. We note that the choice of the right control experiment still a point of debate for various assays including DNase-seq and ATAC-seq.

As discussed in the introduction, one of the usages of sequence-based predictive models is to understand the regulatory elements and their interactions. Having an accurate model does not yet help us to understand the motifs of the studied TFs or their binding grammar. To be able to do this, we need advanced model interpretation tools described in the next chapter.

4 Learning the grammar of transcription factor binding by interpreting sequence-to-profile models

This chapter is based on collaborative work supervised by Anshul Kundaje at Stanford and Julia Zeitlinger at the Stowers Institute for Medical Research. It extends the work described in the previous chapter. I performed almost all the analyses presented in this chapter and have received the following help from others. I conceived the contribution score for the sequence-to-profile model together with Anshul Kundaje. I implemented the first version using input-masked gradient contribution scores. Amr Alexandari implemented the weighted contribution score for DeepLIFT with input from Avanti Shrikumar. I conceived CWM scanning with input from Avanti Shrikumar and Anshul Kundaje. Ziga Avsec, Julia Zeitlinger, and Anshul Kundaje conceived the motif interactions analysis with input from Avanti Shrikumar. I received help from Avanti Shrikumar when applying TF-MoDISco. Melanie Weilert performed the initial annotation of transposable elements with RepeatMasker. Melanie Weilert performed motif validation analyses depicted in Figure A.1. Robin Fropf, Sabrina Krueger and Khyati Dalal performed the experiment at the Stowers Institute for Medical Research and the Institute's Molecular Biology core facility performed the Illumina sequencing. Julia Zeitlinger interpreted the biological results. I prepared the figures together with Julia Zeitlinger. I wrote the chapter with input from Anshul Kundaje and Julia Zeitlinger.

4.1 Motivation

In the previous chapter we have introduced BPNet, a sequence-to-profile model predicting read coverage tracks from DNA sequence at base-pair resolution. To perform this predictive task, the model had to learn the rules of TF binding. This includes the detection of short DNA sequences (motifs) that TFs recognize and the way multiple neighboring binding sites affect each other (grammar). Unfortunately, this knowledge can not be directly extracted from the model. BPNet consists of more than 10 convolutional layers sequentially performing non-linear transformations and a total of 130,984 parameters.

Here, we will extract the knowledge from BPNet by adapting DeepLIFT [110], a feature contribution score, and apply TF-MoDISco [6] to cluster the motifs highlighted by the contribution scores. Finally, we will use the model as an 'oracle' to study the interactions between motifs.

4.2 Methods

4.2.1 Efficient and robust interpretation of sequence-to-profile models

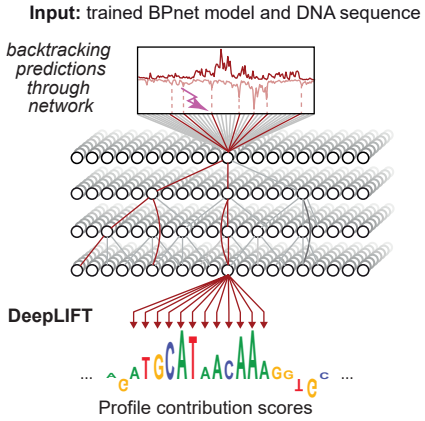


Figure 4.1: DeepLIFT identifies nucleotides predicted to be important for the binding profile by backtracking the prediction for a specific TF through the network. The relative nucleotide importance or contribution is called the profile contribution score.

DeepLIFT is a feature contribution score which decomposes the difference in prediction from the reference input as a linear combination of input features:

$$f(\mathbf{x}) - f(\mathbf{r}) = \sum_{i}^{D} c_i \ , \tag{4.1}$$

where c_i is the contribution of feature i to the output $f(\mathbf{x})$ compared to model prediction the reference input $\mathbf{r}$ (Figure 4.1, Section 2.3.8.3). We note that $f(\mathbf{x})$ is a function returning a scalar. For BPNet, the profile output head for a particular TF returns a LxS tensor, where L is the sequence length and S is the number of output channels or strands for ChIP-nexus. Since the output of BPNet is a tensor and not a scalar, DeepLIFT can only be directly used to compute the contribution with regard to a particular output position on a particular strand.

To compute the contribution for the entire output profile, and not just a single position, we define the profile contribution scores as the weighted sum of the individual contribution scores for all output positions and all output strands (or channels in general):

$$c^{(profile)} = \sum_{i,s} c_{is} p_{is} \tag{4.2}$$

where p_{is} is the predicted probability values obtained by normalizing the (logit outputs) using the softmax function along the sequence axis: $\mathbf{p} = \text{softmax}(\mathbf{f}(\mathbf{x}))$. The rationale for performing a weighted sum is that positions with high profile values should be given more weight than positions with low profile values. In this manner, spiky profiles as

seen in the aggregate ChIP-nexus footprints will get higher overall contribution scores. The downside of such weighted sum formulation is that it would normally require the contribution scores to be computed LxS (=2,000) times for each 1 kb input sequence per TF.

To drastically speed up this computation we exploit the backpropagation algorithm used in DeepLIFT and the linearity of DeepLIFT. We define a new TensorFlow operation as follows:

$$\hat{f}(\mathbf{x}) = \sum_i \text{Const}(p_i(\mathbf{x}))f_i(\mathbf{x}) \ , \tag{4.3}$$

where Const denotes the `tf.stop_gradients` operation which treats the wrapped expression $p_i(x)$ as a constant. By applying DeepLIFT to $\hat{f}(\mathbf{x})$ we obtain, in a single DeepLIFT step, the desired result:

$$c^{(profile)} = \sum_{i,s} c_{is}p_{is} \ . \tag{4.4}$$

Therefore, the computational cost of computing the profile contribution scores is drastically reduced. Pseudo-code of the described op in TensorFlow code looks as follows:

```
wn = tf.reduce_mean(
  tf.reduce_sum(
    tf.stop_gradient(tf.nn.softmax(f, dim=-2)) * f,
    axis=-2
  ),
  axis=-1
)
```

Listing 4.1: TensorFlow op to with respect to which the effective weighted sum of the contribution scores can be computed. f is the predicted pre-softmax output.

We used all zeroes for the reference input $\mathbf{r}$ since it showed the highest correlation with in-silico mutagenesis contribution scores. We used the DeepExplain repository fork available at `https://github.com/kundajelab/DeepExplain/` together with TensorFlow v1.6 to compute DeepLIFT contribution scores.

4.2.2 TF-MoDISco

We computed the profile contribution scores for each TF in all peaks (resized to 1 kb) from the training, validation and test set (i.e. peaks from all autosomes). A null distribution of contribution scores was generated by randomly selecting 4,800 peaks, extracting the sequences, shuffling them and computing the profile contribution scores for the shuffled sequences. We shuffled the sequences in such a way that dinucleotide counts are preserved. TF-MoDISco (v0.5.1.1) was then run for each TF separately using the corresponding contribution scores of the TF in all regions where the corresponding TF was bound.

The TF-MoDISco algorithm (Figure 4.2, [6]) consists of three stages. In the first stage, the total contribution in sliding windows of length 21 (`sliding_window_size`)

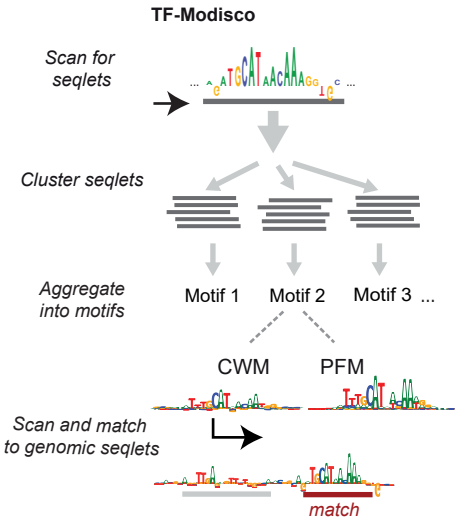


Figure 4.2: TF-MoDISco scans for regions with high contribution scores (seqlets) and clusters them to obtain motifs. Each motif is summarized by the contribution weight matrix (CWM) obtained by averaging the seqlet contribution scores, as well as the position frequency matrix (PFM) obtained by computing the frequency of nucleotides at each position in the corresponding seqlets. Motif instances are mapped back to the genome by scanning the contribution scores with the CWM obtained for each motif.

is computed, both for contribution scores from the real sequences and for contribution scores on the shuffled sequences. The distribution of sliding window scores on the shuffled sequences is used to define a 'null distribution' against which sliding window from the real sequences that pass a FDR threshold of 0.01 (`target_seqlet_fdr`) are identified. Sliding windows are expanded on either side by 10 bp (`flank_size`) are selected in such a way that no two sliding windows overlap by more than 50%. The segments underlying these expanded sliding windows are termed 'seqlets', and are provided to the next stage for clustering. We limited the total number of seqlets to 50,000 for each run of TF-MoDISco in order to always satisfy the memory constraints (250GB).

In the second stage, seqlets are clustered into motifs. First, a similarity for each pair of seqlets is computed using the seqlet contribution scores. For a given pair of seqlets, different possible alignments of the seqlets are considered, and for every alignment, the similarity of the contribution scores is calculated using a correlation-like metric called continuous Jaccard [6]. The best similarity across all alignments is then taken to be the similarity of the seqlet pair. The similarities of the seqlets are provided to a clustering algorithm, after transforming the similarities in a way that grants robustness to the fact that different clusters can have different densities. The clusters are found using a Louvain community detection algorithm [178] that automatically determines the number of clusters by optimizing graph modularity.

After the clusters have been identified, seqlets within a cluster are aligned to each other, and the coordinates of the seqlets are expanded to fill out any overhangs in the

alignment. This kind of seqlet expansion makes it possible to discover motifs that are longer than the sliding window used for seqlet identification in the first stage. A Position Frequency Matrix (PFM) and a Contribution Weight Matrix (CWM) are computed from the aligned seqlets by averaging the base frequencies and the contribution scores respectively. The seqlet coordinates are then re-centered such that the region of highest contribution falls towards the middle of the CWM. Because these seqlet coordinates can be slightly different from the original seqlet coordinates, the second stage is run a second time using the seqlets with the new coordinates, for added robustness.

In the third and final stage, heuristics are applied to postprocess the motifs using the default TF-MoDISco settings for version 0.5.1.1. Clusters appearing to consist of two distinct motifs are split apart, following which clusters with highly similar motifs are iteratively merged. After all merging is complete, any clusters with fewer than 60 seqlets are treated as noise and disbanded, with their seqlets reassigned to larger clusters. Finally, motifs are expanded to the length of 70 bp and then trimmed down to their final lengths by removing flanking positions with an information content of less than 8% of the information of the base with the maximal information content in the motif.

4.2.2.1 Motif clustering and pairwise alignment

To identify and pairwise align similar motifs detected across different TFs, we performed the following motif clustering approach. First, we obtained all possible pairwise alignments of two motifs (i.e. all possible offsets and strand combinations) and identified the smallest continuous Jaccard distance metric using the PFM information content. We then generated a pairwise distance matrix and performed hierarchical clustering in scipy (v1.2.1) using the Ward variance minimization algorithm [179] (`method='ward'`) and optimal leaf ordering [180].

4.2.3 Determining motif instances by scanning the contribution scores instead of DNA sequence

To allow new sequences to be scored for motif instances similar to PWM scanning, we developed a method for scanning the contribution scores with the contribution weight matrix (CWM) from the TF-MoDISco motifs. We note that even though TF-MoDISco already identifies motif instances as seqlets, the detection of motif instances is not comprehensive since the number of considered seqlets (and hence the number of detected motif instances) was capped at 50,000 due to memory constraints.

There are three key differences between PWM and CWM scanning. First, a CWM instead of the PWM is used. CWM is obtained by averaging the contribution scores of all seqlets corresponding to a specific TF-MoDISco motif. Second, in CWM scanning, the contribution scores are scanned instead of the raw sequence. Third, we use a different similarity metric between the contribution scores and the CWM. Let $\mathbf{w}^{CWM} \in \mathbb{R}^{L_W \times 4}$ denote the CWM of length L_W and $\mathbf{C} \in \mathbb{R}^{L_S \times 4}$ denote the contribution scores for one-hot-encoded sequence $\mathbf{s}$ of length $L_S \geq L_W$. The contribution score $C_{i,b}$ for base b at position i is 0 if base b was not observed in the actual sequence (i.e. if $s_{i,b} =$

0). We decompose the similarity metric between the CWM scanning position i of the contribution scores into two parts: i) the L1 norm of the contribution scores at positions between i and $i + L_W$:

$$Score_{contrib}(\mathbf{w}^{CWM}, \mathbf{C}, i) = \sum_{j=1}^{L_W} \sum_{b=1}^{4} |C_{i+j-1,b}| \ , \tag{4.5}$$

and ii) the Jaccard similarity measure between the CWM and L1 normalized contribution scores:

$$Score_{match}(\mathbf{w}^{CWM}, \mathbf{C}, i) = Jaccard(\frac{\mathbf{w}^{CWM}}{||\mathbf{w}^{CWM}||_1}, \frac{\mathbf{C}_{i:i+L_W,b}}{||\mathbf{C}_{i:i+L_W,b}||_1}) \ , \tag{4.6}$$

where $Jaccard$ is the continuous Jaccard distance metric defined in [6]. At each position i, the 'match' score ($Score_{match}$) is computed for $\mathbf{w}^{CWM}$ and its reverse-complement version. The maximum of the two scores is used as the final 'match' score at each position. Note that we did not scan the 'hypothetical contribution' scores as performed by TF-MoDISco since we observed a higher number of false positives using that approach.

To put the obtained scores into the perspective of original seqlets discovered by TF-MoDISco, we computed the 'contrib' and 'match' scores for all the seqlets at their extracted locations. That way, we obtain a distribution of scores determining the corresponding TF-MoDISco motif. We define the normalized 'contrib' score as the fraction of TF-MoDISco seqlets with a 'contrib' score smaller than the 'contrib' score of the CWM at a particular position.

A motif instance is called if:

- 20% or more of the TF-MoDISco seqlets had a 'match' score lower than the considered 'match' score
- at least one TF-MoDISco seqlet had a 'contrib' score lower than the considered 'contrib' score
- The classical PWM score is larger than 0.

We note that the CWM scanning procedure does not use the ChIP-nexus profile information. However, the ChIP-nexus profile score of a particular instance can easily be added to the procedure to further refine the motif instance filtering.

We called motif instances in the union of 1kb wide TF peak regions (150,908) on which TF-MoDISco was ran. We scanned the contribution score of the corresponding TF from which the motif originated (i.e. we scanned Oct4 contribution scores for the motifs discovered by running TF-MoDISco on the Oct4 contribution tracks). We used the trimmed CWMs (Section 4.2.2) for scanning. We removed the motif instances of short motif which overlapped any of the motif instances matching the long (IC$>$ 30) motifs.

4.3 Results

4.3.1 BPNet contribution scores highlight relevant motifs

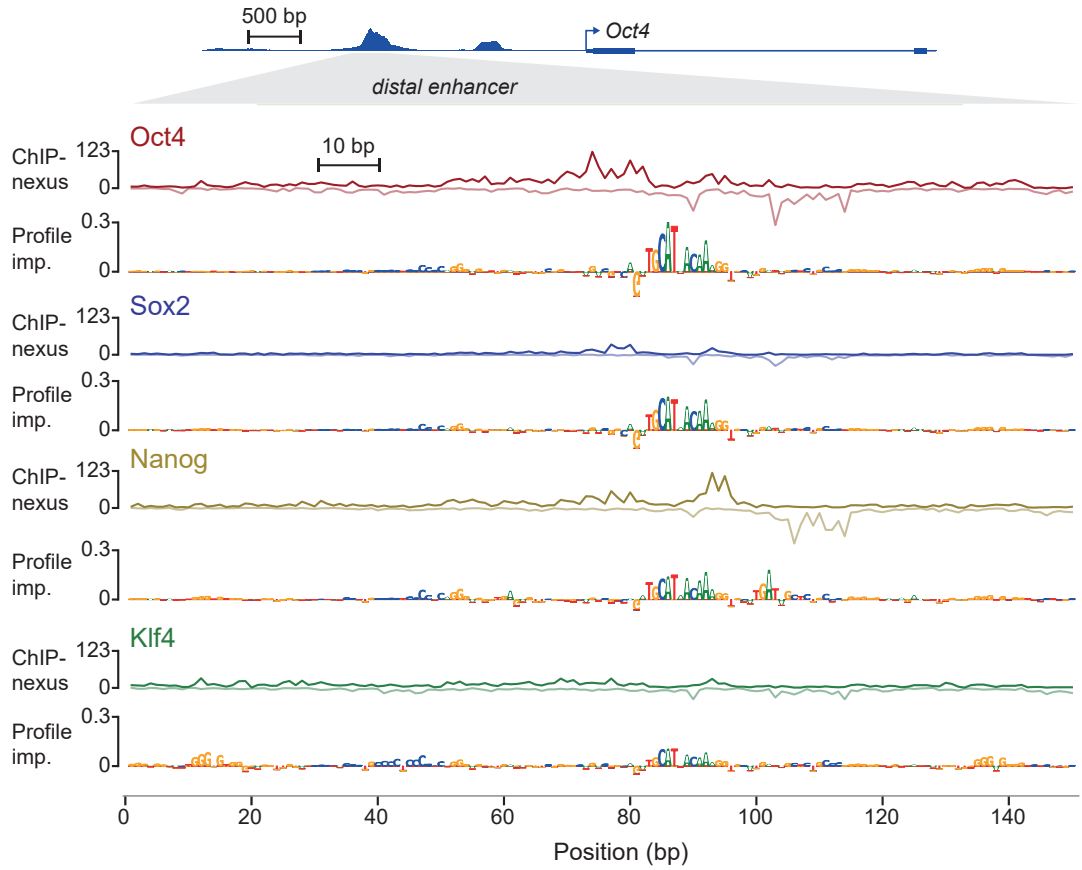


Figure 4.3: DeepLIFT contribution scores of BPNet highlight known sequence motifs. Observed ChIP-nexus read count (reverse strand visualized on the negative axis) and the contribution scores highlighting nucleotides of high importance for the distal Oct4 enhancer (top, chr17:35504453-35504603). ChIP-seq track for Oct4 is shown at the top.

Having shown that BPNet can predict ChIP-nexus profiles, we asked which nucleotides in the input were important for making these predictions. Assigning a 'contribution' or 'importance' score to each input feature using the DeepLIFT algorithm [110] is derived by tracing back the output signal through the network (Figure 4.1). To efficiently calculate the contribution for the entire output profile, which consists of read counts for all positions at each strand for each TF, we introduced a novel way to compute the weighted sum of the individual contribution scores as described in the previous section. Using this method, we obtain a contribution score for each nucleotide in every genomic region for each TF.

The well-studied distal Oct4 enhancer illustrates the nature of the DeepLIFT-derived contribution scores (Figure 4.3). We chose this locus because we observed strong predicted ChIP-nexus footprints for all four TFs, which matched the observed ChIP-nexus data. In the contribution scores, we observed strings of nucleotides that stood out because of their high contribution scores. Strikingly, these local enrichments of contribution scores, which we call *seqlets*, resemble TF binding motifs.

One of the most prominent seqlets is the known Oct4-Sox2 bipartite motif (TGCATNACAA). We note that this motif has high contribution scores for not only Oct4 and Sox2, which are directly bound to the motif, but also for Nanog and Klf4 (Figure 4.3). This suggests that the Oct4-Sox2 motif is indirectly important for the binding of other transcription factors.

The Oct4-Sox2 motif, as well as the known Klf4 motif (CGCCCC), were identifiable as seqlets of the Oct4 enhancer, but the binding specificities of other seqlets were less clear. For example, we observe a short sequence of TGAT being highlighted in the middle of the Nanog footprint (position ∼100). This could be a Nanog motif, but previous reports on the Nanog motif have been conflicting [68, 181, 182, 183, 184, 185, 186] and therefore it is unclear whether this is a specific and commonly found motif for Nanog. We therefore needed a systematic way to identify the most common recurrent patterns in the seqlets, identify which TFs specifically bind to them, and use these motifs to label the seqlets in the genome. This process should then give us a comprehensive list of labeled motif instances in the genome.

4.3.2 TF-MoDISco discovers motifs beyond the known transcription factor binding motifs

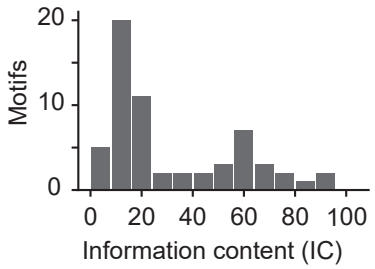


Figure 4.4: A histogram of the information content (IC) of all motifs obtained from TF-MoDISco shows a bimodal distribution. Motifs with an IC < 30 were classified as short motifs, while those with > 30 as long motifs.

To systematically discover recurrent sequence patterns or motifs in seqlets, we ran TF-MoDISco (Figure 4.2, [6]). TF-MoDISco first scans and identifies seqlets that contribute to the binding of each transcription factor, and then clusters the seqlets to identify sequence motifs with similar contribution scores. Significant seqlets were identified by comparing the contribution scores to that of a null distribution derived by from shuffled sequences. The number of seqlets that passed the significance threshold ranged from

17,735 seqlets (Sox2) to 104,209 seqlets (Klf4). From these seqlets, 61 motifs were discovered by TF-MoDISco. Each motif can be represented in two different forms, either as contribution weight matrix (CWM) or as position frequency matrix (PFM) (Figure 4.2). The CWM is the average seqlet contribution score and thus is the best representation for the relative predicted contribution of each nucleotide to the binding of its cognate TF. The PFM, on the other hand, is the average frequency of all nucleotides among the clustered seqlets and thus may contain overrepresented sequences that are not predicted to contribute to TF binding, but co-occur with bases contributing to TF binding.

As we inspected the discovered motifs, it became evident that the PFM of the motifs showed large differences in the sequence information content. The values followed a bimodal distribution and spanned from 5 bits to 100 bits (Figure 4.4). TF-MoDISco clusters seqlets of 70 bp, and each nucleotide can have at most 2 bits. An information content of 100 bits is therefore very high (e.g. corresponding to a sequence motif of length 50 bp with perfectly consistent nucleotides), and considerably larger than expected for a transcription factor binding motif. As we will see below, these long motifs correspond to transposable element sequences, which are frequently found in multiple, very similar copies in the genome, resulting in long PFMs. We note that each copy is nevertheless a unique motif instance in the genome since we only considered uniquely alignable reads when processing the ChIP-nexus data.

Since the primary focus had been to discover TF binding motifs, which tend to be relatively short, we first analyzed 33 short motifs that have low information content (<30 bits) and have support from at least 100 seqlets (Figure 4.5). We found the known Oct4-Sox2, Sox2 and Klf4 motifs, confirming that transcription factor motifs were indeed discovered. However, we found that some of the motifs were discovered multiple times. One reason for this was that motifs that contributed to the binding of more than one TF were discovered for each TF independently. For example, the Oct4-Sox2 motif was discovered from contribution scores of all TFs. Furthermore, some motifs that were discovered for Klf4 or Oct4 were very similar, but it was unclear how distinct these motifs were and whether such differences may be biologically relevant.

4.3.2.1 Discovered short motifs extend beyond the known transcription factor binding motifs

We identified 10 non-redundant short motifs from the 24 short motifs with strong seqlet support (>300, Figure 4.5) using the clustering approach described in Section 4.2.2.1. To investigate which motifs were directly and specifically bound by each transcription factor, we analyzed the ChIP-nexus average footprints at each motif (Figure 4.6B). For the cognate motifs: Oct4-Sox2, Sox2 and Klf4 motifs (Figure 4.6) we observed a sharp average ChIP-nexus footprints for the corresponding TF (grey in Figure 4.6), suggesting that a sharp footprint is indicative of direct binding. Interestingly though, the same motifs sometimes also show a more fuzzy footprint for a TF that is not directly bound. For example, the Sox2 motif not only has a sharp binding footprint of Sox2 but also a weaker, more fuzzy footprint of Nanog (Figure 4.6), suggesting that Nanog is indirectly bound to the Sox2 motif [187, 188]. Such indirect binding is consistent with Nanog

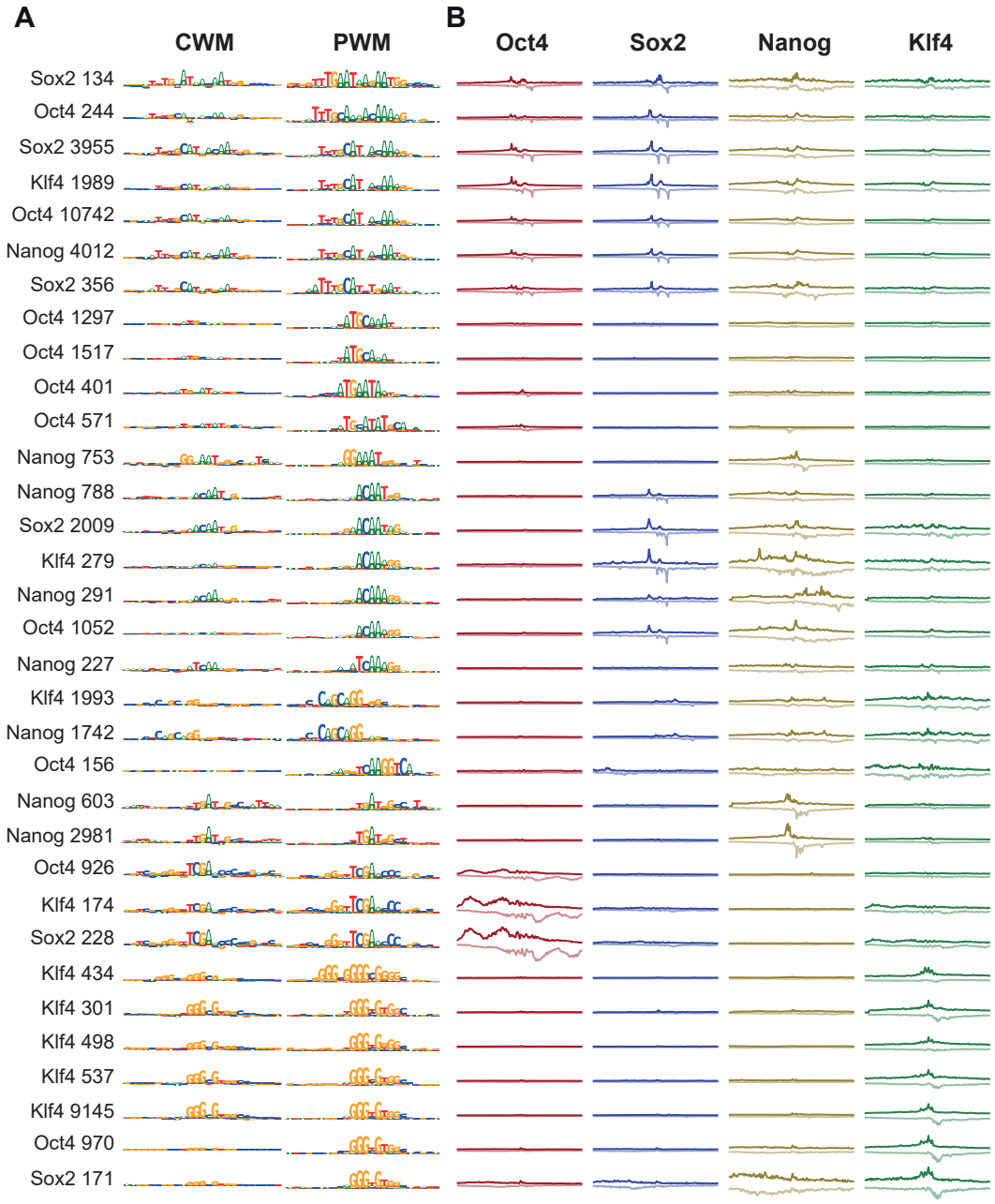


Figure 4.5: Discovered short motifs (IC < 30). **A)** From left to right: TF for which the motif was discovered, number of seqlets supporting the motif, CWM, PFM. **B)** The average read count distribution (footprint). All sequence logos and profile plots share the same y-axis across each column.

interacting with Sox2 through protein-protein interactions [182, 189], although other mechanisms of TF cooperativity could also produce such fuzzy footprints. In summary,

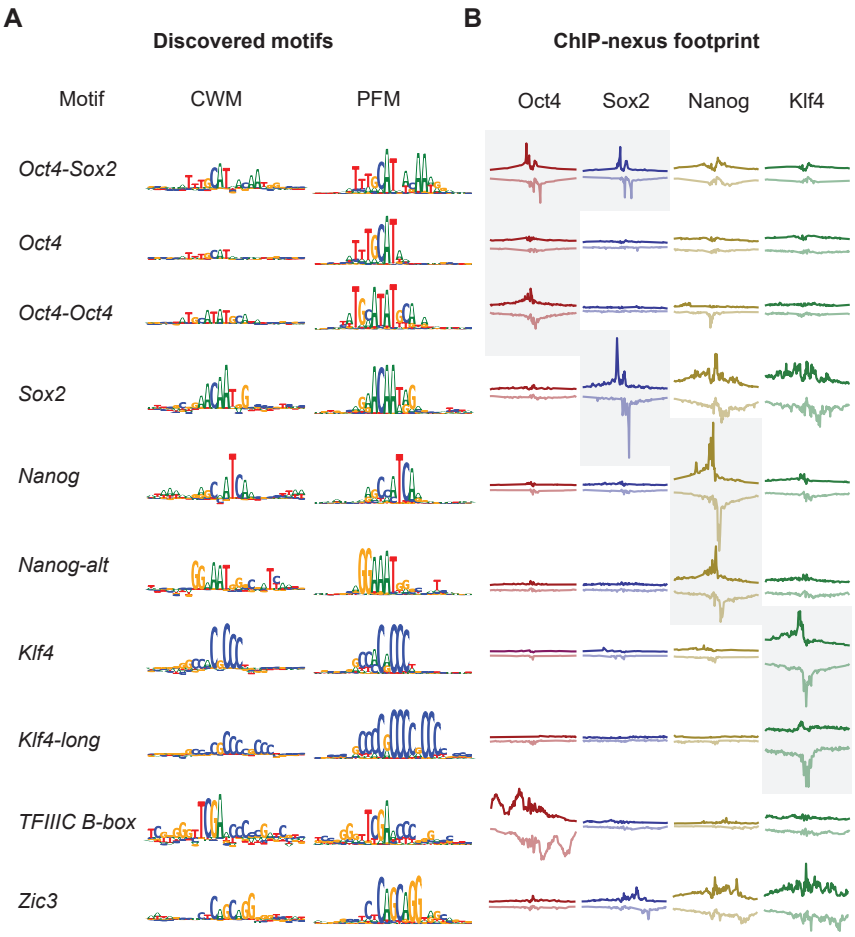


Figure 4.6: TF-MoDISco discovers short motifs that extend known motifs. A) Selected 10 non-redundant motifs from Figure 4.5. **B)** The average read count distribution (footprint) of all TFs at each motif indicates whether a motif is directly bound (sharp profile), indirectly bound (fuzzy profile) or not bound at all. All footprints for each TF share the same y-axis.

the ChIP-nexus footprints serve as a simple way to characterize direct and indirect binding.

Another interesting and related feature was that some motifs contribute to the binding of multiple TFs. As observed for the distal Oct4 enhancer (Figure 4.3), the Oct4-Sox2 motif did not only contribute to the binding of both Oct4 and Sox2, which showed sharp ChIP-nexus footprints over this motif, but also contributed to the binding of Nanog and Klf4, which showed weaker, more fuzzy binding over the motif (Figure 4.6). This suggests that motifs indirectly contribute to the binding of other TFs, revealing potential cooperativity between them.

Since we were able to confirm the cognate motifs known to be specific for the four TFs and important for ESC, we next turned our attention to the remaining motifs

(Figure 4.6). Some of them also mapped to known motifs. For example, while the Oct4-Sox2 motif, which is known to be important in ESCs, was most strongly bound by Oct4, there were two motifs with lower Oct4 binding: a canonical Oct4 motif that binds monomeric Oct4 [22] and a near-palindromic motif that resembles the MORE and PORE motifs, which have previously been shown to bind Oct4 homodimers *in vitro* [190, 191], and to which we refer to as the Oct4-Oct4 motif. Interestingly, this motif has not previously been shown to be bound in ESCs *in vivo*, but is known to be important during neuronal differentiation [192]. We also found an additional, longer motif for Klf4 (Klf4-long, Figure 4.6).

Importantly, we discovered Nanog motifs that appear to be highly specific based on their ChIP-nexus footprints. Nanog is a homeodomain TF whose binding motif has previously been reported with conflicting results [68, 181, 182, 183, 184, 185, 186]. The most frequent Nanog motif returned by TF-MoDISco contains a TCA core (as identified in the distal Oct4 enhancer in Figure 4.3). This motif resembles the Nanog motif identified previously by a thermodynamic model from ChIP-seq data [183].

Finally, we discovered motifs that are known to be bound by TFs that we did not directly measure, suggesting that these TFs act as cooperating partners. The first motif is the motif for Zic3, an important ESC transcription factor. The Zic3 motif has weak Sox2, Nanog and Klf4 footprints, suggesting that Zic3 helps these factors bind. Among the motifs with less than 300 motif instances, we also discovered the motif for Essrb, another important ESC transcription factor (Figure 4.5). Finally, we identified a long palindromic motif that was not expected since it resembles a motif called the B-box, which is bound by TFIIIC and mediates RNA polymerase III (Pol III) transcription [193]. We found high levels of Oct4 bound to this motif in an unusual pattern (Figure 4.6).

To obtain more insights and validation for the newly discovered motifs, we analyzed these motifs in more detail and performed additional ChIP-nexus experiments (presented in Section A.2).

4.3.2.2 Long motifs resemble transposable elements frequently bound by TFs

While we previously examined short motifs to identify TF binding motifs, we now examined the long motifs (PFM information content > 30 bits and more than 100 motif instances). Annotations by Repeat Masker [194] show that the vast majority of motif instances (often $\sim$90%) overlap transposable elements (TEs) (Figure 4.7A). Sorting the motif instances by their number of mutations from the consensus PFM (Kimura distance) reveals that the strongest similarity is found at the transcription factor binding motifs (e.g. Klf4 motifs in LTR ORR1A2 in Figure 4.7B). We identified a total of 18 TEs, supported by a total of $\sim$3,500 seqlets (i.e. positions in the genome). Based on each motif's most frequent annotation by Repeat Masker, the majority of TEs are classified as long-terminal repeats (LTR) transposons, with RLTR9D being the most common (Figure 4.7C).

TEs have the ability to become active during early mouse development and in ESCs. Some TE insertions give rise to functional cis-regulatory sequences that activate nearby genes and can rewire the transcriptional regulatory network of ESCs [195]. However, the

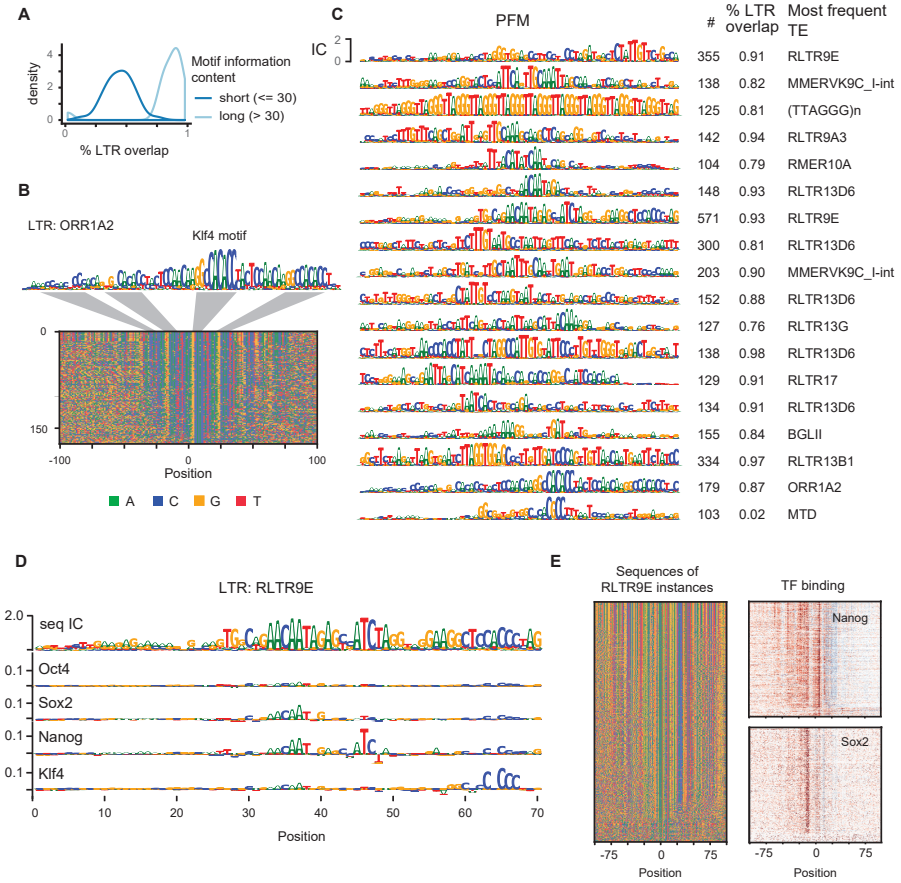


Figure 4.7: Discovered long motifs match transposable elements. **A)** Fraction of seqlets that overlap a transposable element (TE) as annotated by Repeat Masker (Section A.3.1). Note that seqlets of most long motifs with high information content (> 30) overlapped TEs annotated by Repeat Masker [194]. **B)** Example of a discovered long motif that is predominantly annotated as Long Terminal Repeat (LTR) transposon ORR1A2. The PFM is shown at the top and Klf4 motifs are highlighted in grey below. A sequence plot below of all individual motif instances in the genome, sorted by the number of substitutions (Kimura distance) from the consensus motif, suggests that the transposon sequence is subject to mutations. **C)** Overview of all discovered TEs reporting the corresponding PFM, number of seqlets, fraction of motif instances overlapping with a TE, and the most frequent TE Repeat Masker annotation. **D)** While the PFM represents the TE consensus sequences, the CWM highlights the sequences within the TE that contributes to the binding of the four TFs. The RLTR9E transposon is shown as an example, which harbors a binding motif for Sox2, Nanog and Klf4. **E)** The substitution-sorted (Kimura distance) sequences of each individual RLTR9E motif instance and the corresponding Sox2 and Nanog ChIP-nexus binding patterns indicate a correlation between increased number of mutations and decreased TF binding.

extent and mechanisms by which this occurs has been unclear as there is currently no robust method to systematically identify binding sites derived from TEs. This widespread identification of TEs is likely due to the unique ability of BPNet and TF-MoDISco to identify relevant binding motifs while detecting embedded, larger-scale patterns across these aligned sequences. Because of this, we were provided with an opportunity to characterize TF binding at TEs in greater detail.

We noticed that the PFM and CWM of TEs often differed. The PFM typically exhibits high information content across many bases, while the CWM accentuates the predicted binding motifs of each TF (Figure 4.7D). This implies that the long sequence is not necessary for TF binding, but is instead present for some other reason, such as many of the TEs descending from a common ancestor. By analyzing these featured CWM patterns, we noticed that many long motifs contained multiple binding motifs for TFs. For example, the RLTR9E transposon contains binding motifs for Sox2, Nanog, and Klf4 that are each emphasized by their respective CWMs (Figure 4.7D). As expected, these binding motifs also exhibited ChIP-nexus footprints suggesting they are indeed bound by the corresponding TFs (Figure 4.7E).

These findings highlight BPNet's ability to simultaneously identify large sequence patterns and still pinpoint the specific nucleotides important for binding. More generally, these unexpected insights regarding TEs illustrates the unique advantage of deep learning to derive insight without reliance upon prior biological assumptions (such as the nature of a TF binding motif) enabling more unbiased discovery of biological phenomena.

4.3.3 CWM scanning reveals the preference for 10 bp spacing of homeodomain TFs

Another unexpected pattern that emerged from the data was a strong periodicity of Nanog binding. When we inspected the PFM and the CWM of the main Nanog motif (at full length before trimming it to the core sequence), we noticed that the flanks of the CWM exhibited periodic patterns of contributing A or T nucleotides at ∼10 bp distances (Figure 4.8A top). This pattern was not found in the PFM, suggesting that these nucleotides are not statistically overrepresented in these sequences, yet contribute to the Nanog binding predictions. Indeed, the same periodic pattern was observed across the individual contribution scores at Nanog motif instances (Figure 4.8A middle), suggesting that this periodicity was a consistent feature for the Nanog binding predictions.

To better understand the periodicity observed for Nanog, we calculated the average contribution scores across a 200 bp region centered on the Nanog motif (Figure 4.8A bottom). We then subtracted the smoothed averaged contribution to account for the higher binding in the center and performed a Fourier power spectrum analysis. This revealed a strong periodicity pattern averaging around 10.9 bp (± 0.6 bp) (Figure 4.8B). Strikingly, this ∼10.9 bp periodic pattern corresponds to the helical periodicity of the DNA, a biophysical parameter of DNA that BPNet was not designed to explicitly account for and thus was discovered *de novo*.

To test whether other motifs also show periodic spacing, we calculated the fraction of the power spectrum at 10.9 bp periodicity pattern for all discovered motifs for each TF

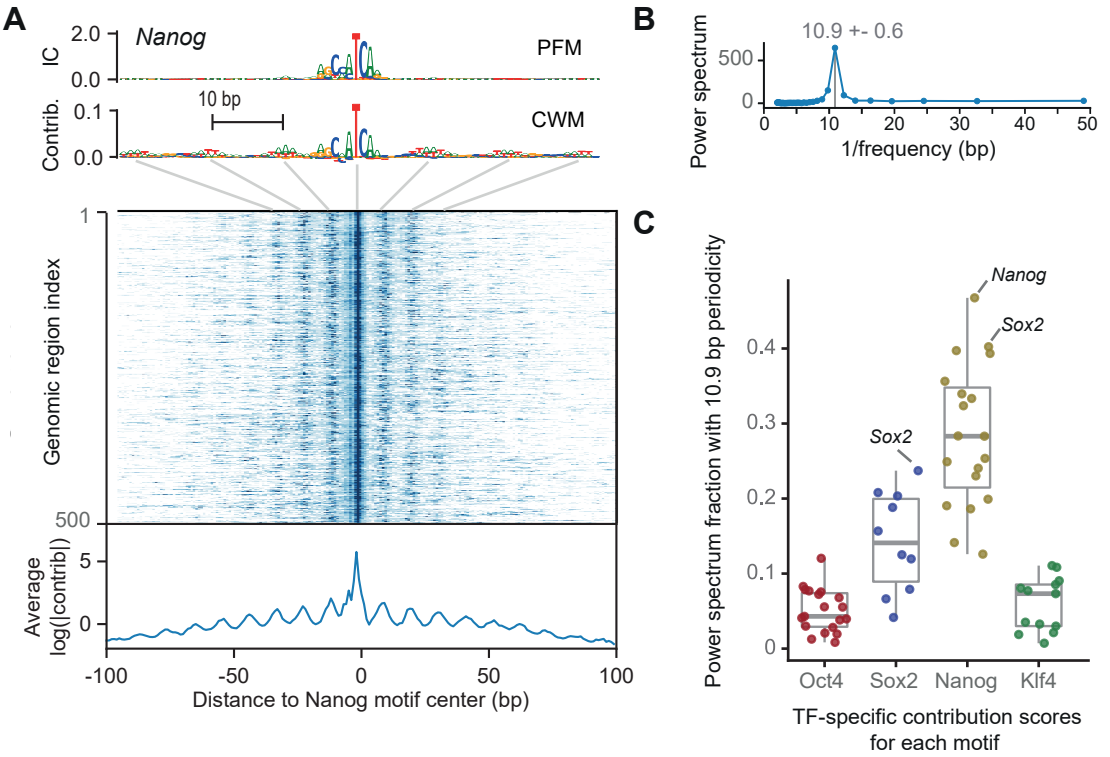


Figure 4.8: Periodicity of the Nanog motif. A) The PFM and CWM for Nanog (top). CWM has nucleotides in the flanks that follow a periodic pattern, which is also observed in the contribution scores of the individual Nanog instances (shown as heat map in the middle) or the average log contribution score (bottom). **B)** Fourier power spectrum of the average log contribution score (after subtracting the smoothed signal) reveals an average periodicity of 10.9 bp (±0.6). **C)** Fraction of the power spectrum with 10.9 bp periodicity for all discovered motifs for different TFs. Marked points correspond to the canonical Nanog or Sox2 motifs. The periodicity is highest for motifs that contribute to Nanog binding, followed by the motifs contributing to Sox2 binding.

(Figure 4.8C). This analysis revealed that genomic instances of the main Nanog motif have the strongest periodicity, and that other motifs that are predicted to contribute to Nanog binding, including another Sox2, also showed strong periodicity (Figure 4.8C). Motifs important for Sox2 binding also showed some moderate periodicity, while motifs contributing to Oct4 and Klf4 binding had minimal periodicity (Figure 4.8C). These results are consistent with recent *in vitro* studies which have shown that homeodomain TFs (Nanog and Sox2) often bind in a 10 bp periodic pattern to nucleosomes [196].

So far, the periodic pattern was observed in the contribution scores derived from BPNet and was not strongly linked to a periodic pattern in the DNA sequence. However, if Nanog preferentially binds in a periodic pattern, one might expect that different instances of Nanog motifs in the genome might also show this preferential spacing to each other.

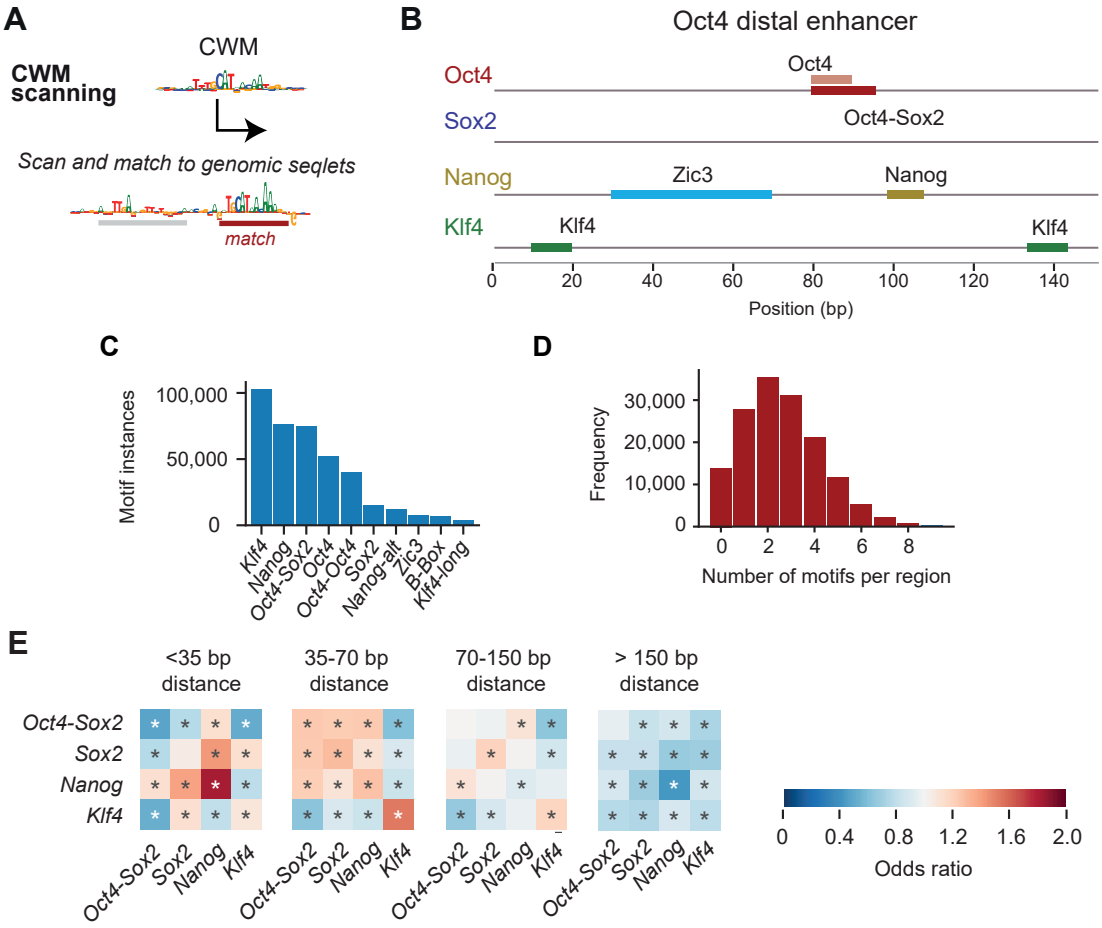


Figure 4.9: Determining motif instances in the genome by CWM scanning. A) Motif instances are mapped back to the genome by scanning the contribution scores with the CWM obtained for each motif (Section 4.2.3). **B)** Motif instances obtained by CWN scanning for the Oct4 distal enhancer shown in Figure 4.3. Each motif scanned the contribution scores for the corresponding TF (e.g. Oct4 for the Oct4-Sox2 motif). **C)** Total number of motif instances found in the peak regions for the main short motifs as shown in Figure 4.6. **D)** Number of mapped motif instances found per 1 kb wide peak region. **E)** Odds by which two motifs are found within a specified distance from each other divided by the odds the two motifs would be found in the proximity by chance (observed by permuting the region index). * denotes p-value $<$ 10-5 using Pearson's Chi-squared test (Section A.3.3).

In order to analyze the pairwise spacing of motifs in the genome, we developed CWM scanning, a method to determine the motif instances in the genome using the contribution scores (Figure 4.9A, Section 4.2.3). In brief, we used the trimmed CWM of each motif, scanned the contribution scores in ChIP-nexus peaks and computed the similarity between the CWM and the contribution scores (Section 4.2.3). Nucleotides in the genome received a motif label if they showed a high contribution score and were suffi-

ciently similar to a discovered motif. For example, 6 motif instances were identified in the distal Oct4 enhancer described above (Figure 4.9B). The TGAT sequence highlighted by DeepLIFT (Figure 4.3) was labeled as a Nanog motif (Figure 4.9B).

Overall, CWM scanning annotated more than 390,000 instances of the 10 selected motifs in the 150,908 ChIP-nexus peaks, with Klf4 motifs being the most frequent one (Figure 4.9C). Importantly, 72,696 (48.1%) have at least three mapped motifs and 20,352 number have at least 5 motifs (Figure 4.9D). Such precise maps of motifs opens the path towards analyzing motif composition and spacing at enhancers.

We analyzed which motifs co-occur more frequently than expected by chance (compared to permuted motif positions in the regions). We focused on the pairwise interactions between the motifs most strongly bound by Oct4, Sox2, Nanog and Klf4, which are the Oct4-Sox2, Sox2, Nanog, and Klf4 motifs, respectively, as shown in Figure 4.6. The results show that the Nanog motifs are most strongly overrepresented at short distances to Nanog and Sox2 motifs (<35 bp). At intermediate distances (35-70 bp), the Oct4-Sox2, Sox2 and Nanog motifs all preferentially co-occur, while the Klf4 motif only co-occurs more frequently with other Klf4 motifs. At nucleosome-range distances (70-150 bp), the Oct4-Sox2 motif still co-occurs with Nanog, while Sox2 and Klf4 motifs co-occur with itself. Strikingly, beyond nucleosome distance (>150 bp), motif pairs are no longer over-represented, suggesting that longer enhancer sequences or motif interactions are not frequent.

Next, we analyzed the distance distribution between motif instance pairs. Specifically, we focused on motif pairs that co-occurred frequently. We observed that Nanog, Sox2, Oct4-Sox and Zic3 were preferentially spaced at a multiple of 10-11 bp from the Nanog motif for all motif orientations (Figure 4.10A-D). The most striking signal was observed for the Nanog<>Nanog motif pair (Figure 4.10A). Interestingly, the Nanog motif was not spaced at the multiple of 10-11 bp for all motif partners. For example, the Nanog<>Klf4 pair did not show any positional preference (Figure 4.10E). In addition to the periodic preferential spacing, we observed that some pairwise positions were unexpectedly frequent, such as the Klf4<>Klf4 pair with ++ strand orientation spaced at 60 bp (Figure 4.10E). Motif instances with that particular spacing are likely part of a transposable element. We note that a large fraction of such motif pairs was already removed by excluding positions mapped to long motifs discovered by TF-MoDISco (Figure 4.7).

These results suggest that Nanog has a broad tendency to bind DNA cooperatively if motifs are spaced with ∼10 bp periodicity. In support of a cooperative binding mechanism, Nanog motifs in pairwise combination with Nanog, Sox2 or Oct4-Sox2 show higher average Nanog ChIP-nexus binding when the motifs are spaced at the preferred distance (Figure 4.10G-I). Thus, partner motifs that enhance Nanog binding can either be a few A or T nucleotides (as found in the Nanog motif flanks), another Nanog motif or motifs of other TFs.

This broad preference of Nanog to bind with ∼10 bp periodicity is interesting since such spacing between motifs has been regarded as an important element of the cis-regulatory code. There are a number of instances with experimental support [197, 198, 199, 200, 201, 196, 202], as well as computational evidence [203]. However, genome-wide

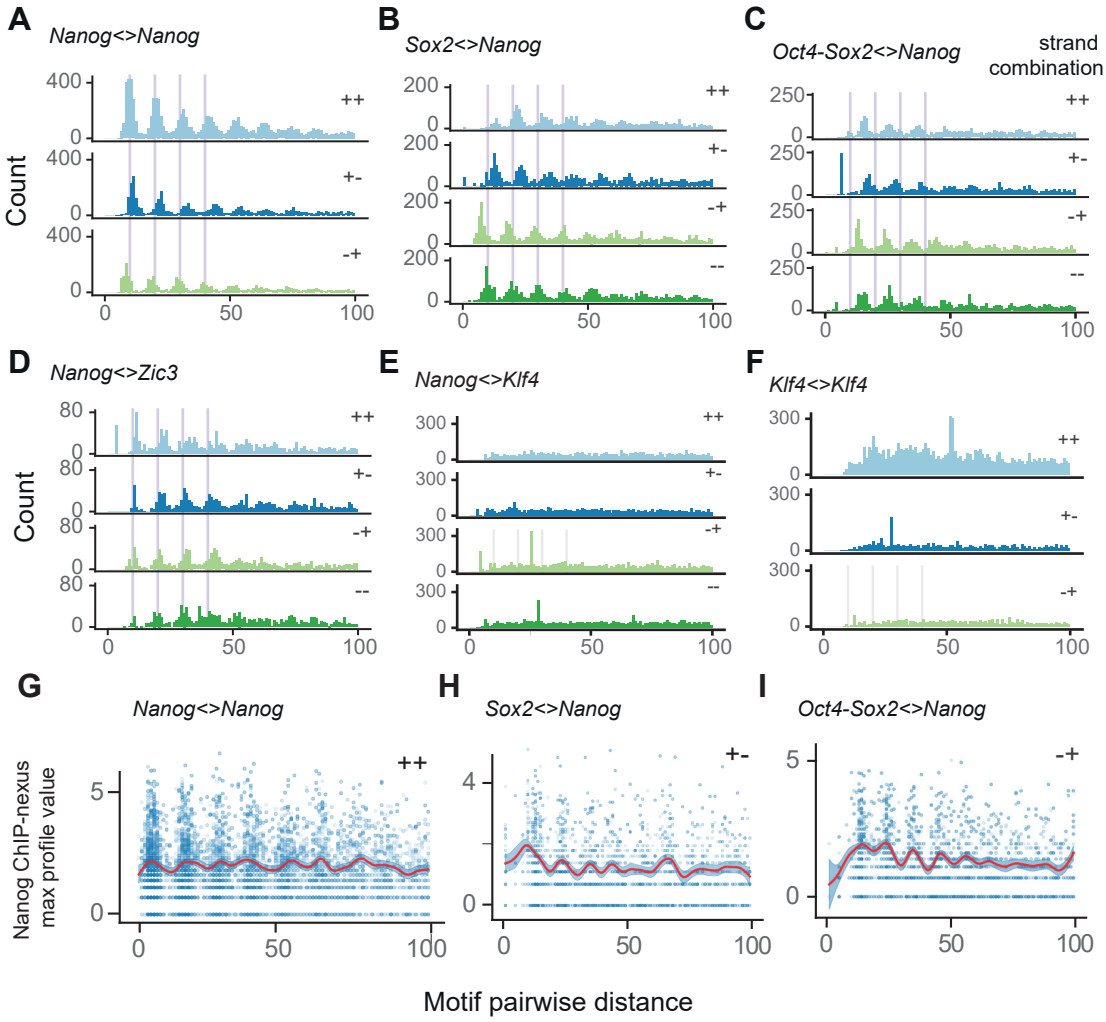


Figure 4.10: Pairwise spacing of CWM scanning motif instances. A-F) Pairwise spacing of motif instances in the genome for different pairs in all possible orientations. Distance between two motifs is always kept positive by placing the second motif in the pair downstream of the first motif in the pair. All 4 motif orientations are considered: + denotes the motif lies on the forward strand and - denotes the motifs on the reverse strand. Motif pairs overlapping long motifs discovered by TF-MoDISco (Figure 4.7) were excluded. Vertical grey lines are placed at 10, 20, 30 and 40 bp. **G-I)** Observed ChIP-nexus profile height in the genome for different motif pairs. Nanog binding to a Nanog motif is higher when another motif such as Sox2 or Oct4-Sox2 is located nearby with the preferred periodic spacing.

evidence for such broad TF binding preference has not been observed and is therefore intriguing.

4.3.4 Interrogating BPNet in silico reveals short-range and long-range cooperate interactions between motifs

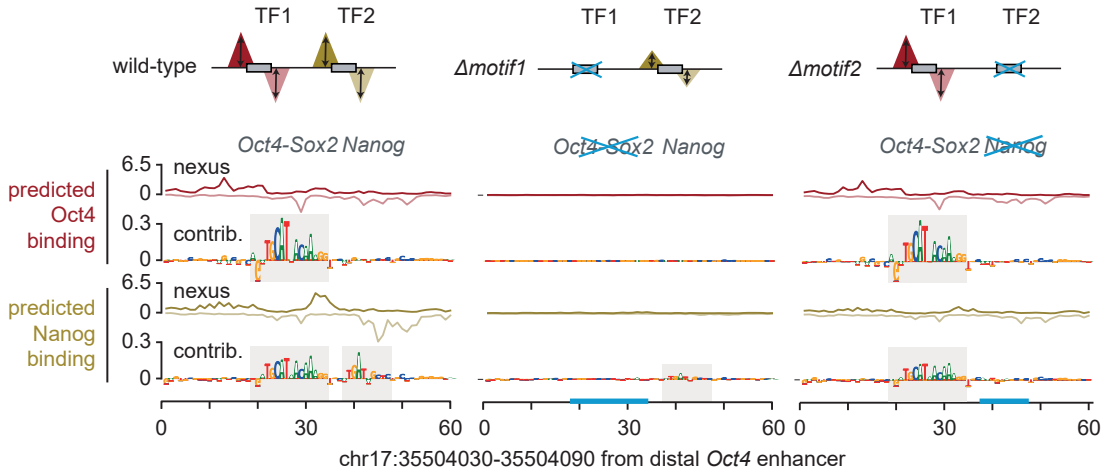


Figure 4.11: BPNet learns directional effect between motifs. Oct4-Sox2 and Nanog motif in the distal Oct4 enhancer. BPNet predictions and contribution scores suggest that Oct4-Sox2 and Nanog motifs are bound by Oct4 and Nanog respectively (left). After the Oct4-Sox2 motif is replaced with a random sequence, binding on both motifs is lost (center). By contrast, only Nanog binding is lost after replacing the Nanog motif with a random sequence while Oct4-Sox2 motif remains bound by Oct4 (right).

Since we observed evidence for cooperative interactions involving Nanog, we set out to develop methods to systematically extract cooperative interactions between motifs from the BPNet model. Specifically, we wanted to measure whether binding of a TF to its motif was enhanced in the presence of another motif, and how this change in binding might depend on the distance between the motifs. The advantage of having a trained BPNet model is that it can be applied at large-scale to any sequences, thus it should be possible to extract cooperativity between motifs *in silico*, as long as such dependencies were learned by BPNet. By contrast, studying the observed ChIP-nexus counts in the genome as shown in Figure 4.10 is limited by the number of bound genomic sequences and such analysis might also be confounded by the presence of other binding sites.

To see whether BPNet has learned any dependencies between motifs, we investigated the Oct4 distal enhancer. We observed that perturbing the Oct4-Sox2 motif (replacing it with random sequence) resulted in a loss of Nanog binding (Figure 4.11 center). By contrast, perturbing the Nanog motif did not cause any change in Oct4 binding at the Oct4-Sox2 motif (Figure 4.11 right). This suggests that the Oct4-Sox2 motif is important for Nanog binding and not the other way around. It also suggests that BPNet has learned this directional effect from the data and is not simply considering motifs as independent when making predictions.

To study the directionality of motifs more broadly, we considered two types of sequences containing motif pairs: synthetic random sequences and genomic sequences

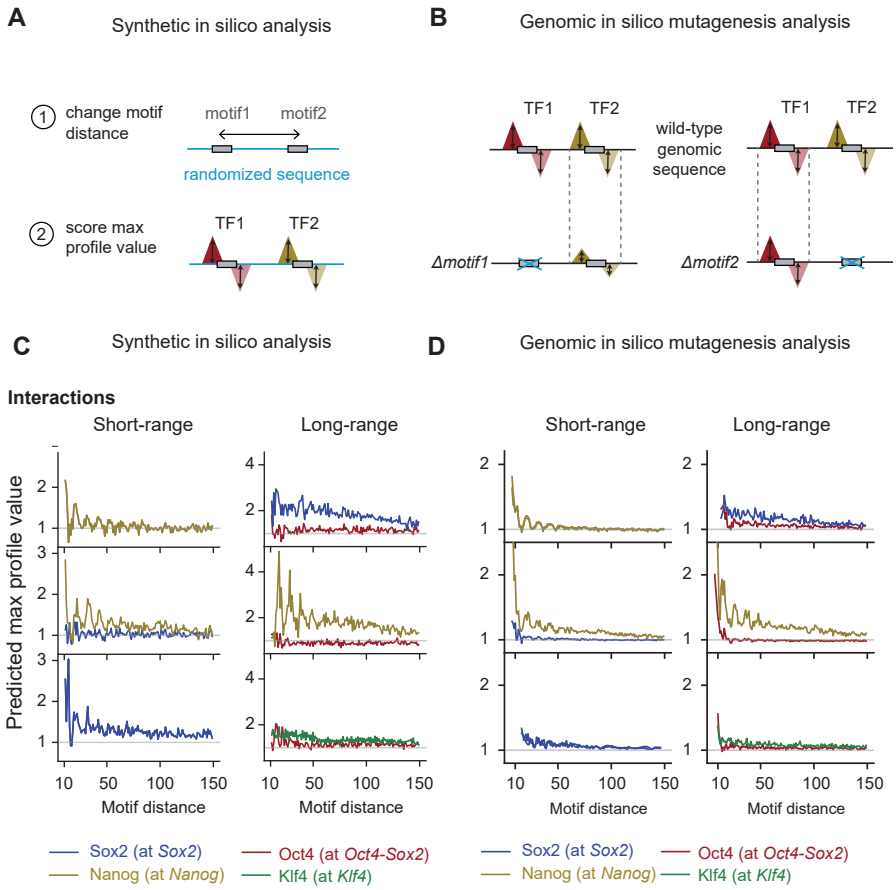


Figure 4.12: In silico analysis of motif interactions. A) Synthetic *in silico* analysis tests whether two TFs (TF1, TF2) influence each other's binding to their respective binding motif. The two motifs are embedded into 128 random sequences at a specific distance from each other. After BPNet predicts the average TF binding profile for each distance, TF binding levels are scored by measuring the maximum peak height at the expected position on each strand (based on the position of the observed average ChIP-nexus footprint). **B)** Genomic *in silico* motif interaction analysis measuring the change in TF binding in a genomic sequence after a nearby motif is mutated. For each motif pair (TF1, TF2), TF binding levels (measured as maximum peak height as above) are compared between the wild-type genomic sequence and the corresponding mutated sequence where the motif for the other transcription factor is replaced by random sequence. For both, **A** and **B**, the predicted ChIP-nexus profile originating from the neighbour motif was subtracted from the profile height (Section A.3.2). **C)** Examples from the synthetic *in silico* analysis as outlined in **A** showing either short-range interactions involving Nanog and Sox2 (left) or long-range interactions exerted by the Oct4-Sox2 motif on the binding of Sox2, Nanog or Klf4, respectively (right). **D)** The genomic *in silico* mutagenesis analysis yields similar results for the same motif pairs as shown in **C**.

identified by CWM scanning (Figure 4.12A,B). In the first approach, the two motifs of interest are embedded in many random DNA sequences, and the TF binding is then measured as a function of distance between the motifs (Figure 4.12A). Such an approach is not feasible experimentally since synthetic sequences may harbor cryptic binding motifs for TFs (even after excluding known motifs), and therefore the number of sequences tested would have to be large in order to obtain an average binding signal for each motif distance. In the *in silico* approach, however, we can easily use more than 100,000 random sequences as the synthetic sequence context, thereby averaging out any indirect effects. Furthermore, since BPNet predicts the high-resolution TF ChIP-nexus footprint, we are able to measure the TF binding signal around the position of the footprint's summit on each strand and to subtract indirect binding from the footprint's shoulder (Section A.3.2). This ensures maximum specificity of the measured TF binding signal.

When we used this synthetic approach on the BPNet model, we found again that Nanog binding is strongly increased when another Nanog motif is nearby, but only if the motif was spaced at a multiple of 10-11 bp (Figure 4.12C). This interaction was strongest at close distance and faded into background signal at around ∼70-75 bp. A similar enhanced binding of Nanog in a periodic pattern was observed when the Sox2 motif was nearby (Figure 4.12C). This was not true the other way around since Sox2 binding was not enhanced in the presence of Nanog. Sox2 binding was however enhanced in the presence of another Sox2 motif, but in a less periodic fashion (Figure 4.12C). We refer to these interactions as short-range interactions since they are strongest when very close and decay rapidly in a non-linear fashion with further away distances.

Unlike these short-range interactions, we also observed long-range interactions that were mediated by the Oct4-Sox2 motif. In the presence of Oct4-Sox2 motif, the binding of Sox2 and Nanog was strongly enhanced and this effect was apparent even at distances up to 150 bp (Figure 4.12C right). This long-range effect was also observed for the binding of Klf4, although the overall effect on binding was much smaller. In all cases, this effect was directional since the Oct4-Sox2 motif strongly affected the binding of the other TFs, while the motifs of the other TFs did not substantially affect the binding of Oct4.

In the second approach, we searched for motif pairs in the original genomic sequences and then tested whether a motif enhances the binding of another TF by replacing it with a random sequence (Figure 4.12B). The advantage of this approach is that we can directly compare predicted patterns to the experimentally observed *in vivo* binding data. Using this approach, we observed similar interaction patterns as for the synthetic sequences, albeit of lower magnitude (Figure 4.12D). The smaller effect sizes might be due to the imperfect binding motifs present in the genome since the synthetic *in silico* approach used the motif consensus sequences. It is also possible that perturbation of binding motifs can be buffered by additional binding motifs that are present in the genomic sequences, but not in the synthetic context.

In summary, both our approaches yielded similar results and pointed us to two interesting observations (Figure 4.13). First, we observed a difference between short-range and long-range interactions by the way the motif interaction strength decays with further distances (compare <35 bp with 70-150 bp in Figure 4.13). Interestingly, such distinction

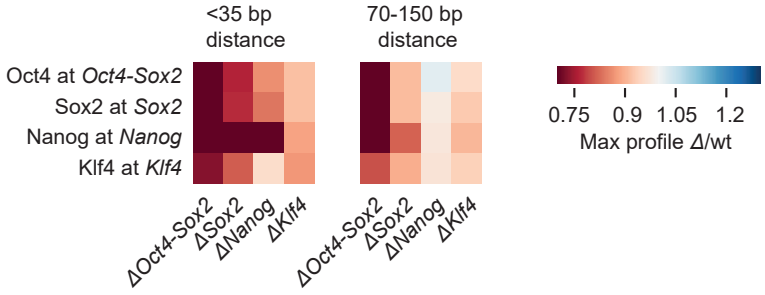


Figure 4.13: Motif hierarchy. Quantification of the results shown in Figure 4.12D as a heat map displaying the median effect across all sites with specific spacing distances. Motif pairs spaced at < 35 bp are shown as representative for short-range interactions, while motif pairs spaced 70-150 bp are shown as representative for long-range interactions, which mostly occur within nucleosome distance.

correlates well with known mechanisms by which TFs interact. Short-range interactions likely correspond to interactions that are mediated by protein-protein interactions or DNA allostery [27, 204], while long-range interactions are likely to be mediated by pioneer TFs acting on nucleosome. Indeed, Oct4 and Sox2 have both been observed to have properties of pioneer TFs [155]. Our results suggest that it is the bipartite Oct4-Sox2 motif that has the strongest effect up to a distance of 150 bp, consistent with an Oct4-Sox2 heterodimer having the strongest pioneering activity, allowing other TFs to bind within nucleosome distance.

Second, we observed a strong asymmetry in the pairwise interactions between motifs. In both the short-range and long-range interactions, one motif usually had a stronger effect on the binding of the other TF to the its motif (Figure 4.13). For example, the Nanog motif was strongly influenced by nearby Oct4-Sox2 and Sox2 motifs, but Oct4 and Sox2 binding was not much influenced by the Nanog motif. This asymmetry suggests a hierarchical enhancer model in which some TFs need to bind first before other TFs can efficiently bind to the enhancer. Such hierarchy has been observed for pioneer TFs [205], but our results suggest that directional interactions are much more common and extend beyond pioneer TFs. Based on these observations, we propose that directional interactions are an important part of the cis-regulatory code.

4.3.5 Comparison to state-of-the-art methods

4.3.5.1 ChExMix and PWM scanning

To evaluate the extent and quality of motifs discovered by BPNet in the light of previous methods, we compared our approach to ChExMix [164]. ChExMix is a state-of-the-art motif discovery and TF binding event calling method for ChIP-exo and ChIP-nexus data. We ran ChExMix (v0.3) with default parameters on each of the studied ChIP-nexus data for each TF (Oct4, Sox2, Nanog and Klf4). We used the pooled BAM file containing the reads of all the replicates for the corresponding TF. We blacklisted the

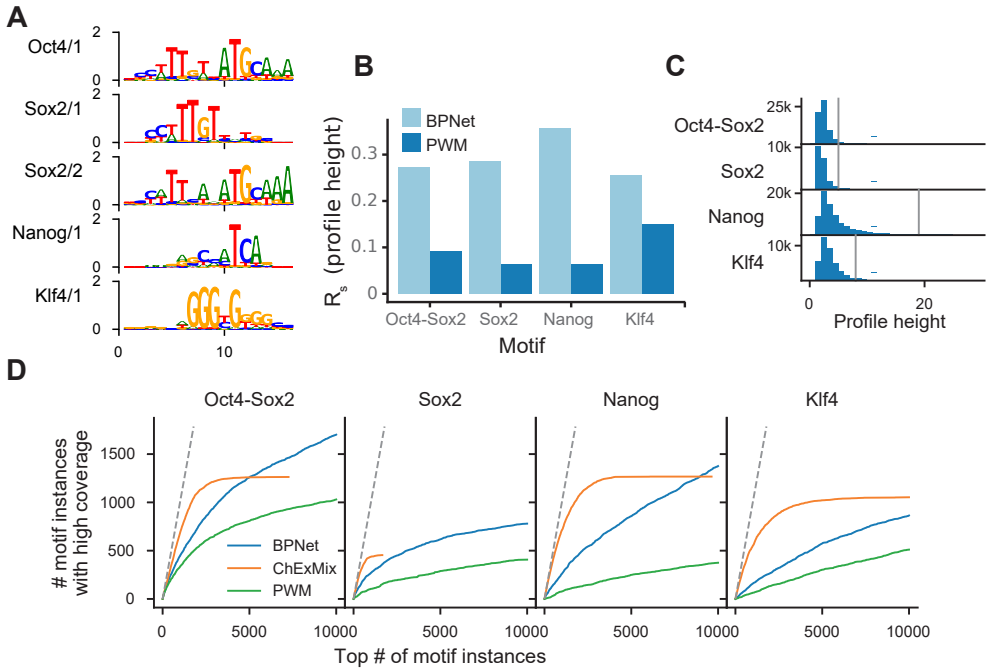


Figure 4.14: BPNet and TF-MoDISco discover more motifs than ChExMix and map motifs with greater accuracy than PWM scanning. A) Motifs discovered by ChExMix from Oct4, Sox2, Nanog and Klf4 ChIP-nexus data. **B)** Spearman correlation of the motif instance score (motif contribution score for BPNet and PWM score for PWM) with the profile height as measured by the maximal number of ChIP-nexus aligned reads at positions within 35bp from the motif center. **C)** ChIP-nexus profile height distribution at BPNet motif instances for different TFs. Vertical grey line denotes the 90th percentile which is used as a stringent threshold for determining motif instances showing a ChIP-nexus footprint. **D)** Number of motif instances showing a footprint (y-axis) as measured by the ChIP-nexus profile height larger than the threshold defined in **C** within the top-N motif instances prioritized by the corresponding method (x-axis). High motif contribution score was used to prioritize motif instances for BPNet, high PWM score for PWM and high profile score for ChExMix. Note that BPNet and PWM methods do not use the profile information whereas ChExMix is already using the read distribution at the motif instance to determine the profile score. Motif instances overlapping the peak regions as called by MACS2 from the held-out test chromosomes (1, 8 and 9) were used for **B**, **C** and **D**.

same regions as for the ChIP-nexus pipeline and used the provided mm10 background file[1].

ChExMix discovered 5 motifs in total (Figure 4.14A). These were the cognate motifs for each of the TFs: Oct4-Sox2, Sox2, Nanog and Klf4. Motifs with fuzzy indirect footprints such as Zic3, B-Box, Essrb, or dimer-motifs such as Oct4-Oct4 or other motif variants were not discovered. We speculate that these motifs were missed because of

[1] `http://lugh.bmb.psu.edu/software/chexmix/backgrounds/mouse.back`

lower number of reads and inconsistent profile shapes (especially for the fuzzy footprints). ChExMix also did not discover long TE motifs. Although changing the parameters in the motif discovery step of ChExMix may allow the discovery of some TEs, the dependence on these parameters makes it difficult for ChExMix to discover TEs alongside short motifs in a flexible manner. We conclude that the motifs discovered by BPNet extend well beyond the motifs discovered by ChExMix.

To evaluate the quality of the called motif instances in the genome, we compared the BPNet approach of CWM scanning to classical position weight matrix (PWM) scanning. Unlike ChExMix (see below), both CWM scanning (BPNet) and PWM scanning only use the sequence information to identify motif instances and thus can be directly compared. To score the quality of the identified motif instances, we determined their ChIP-nexus profile heights, as measured by the number of ChIP-nexus reads at the maximum position in the motif vicinity ($\pm$35bp from the motif center). The results show that the contribution scores of (BPNet) CWM motif instances correlated much more strongly with maximum profile heights than the PWM affinity score of instances identified by PWM scanning (Figure 4.14B). This implies that the contribution scores are a better proxy for TF occupancy than the PWM score. This makes sense since contribution scores consider the entire sequence context of the motif within the 1 kb region and thus allow to integrate more information relevant for TF binding. By contrast, the PWM is limited to the local sequence context ($<$20 bp) and does not consider the possibly synergistic interactions between nucleotides.

Since the contribution scores correlated much better with the ChIP-nexus profile height than the PWM score, we asked whether this approach also improved the often criticized high false positive rate of motif instances obtained by PWM scanning. To determine the false positive rate of the motif instances in the test chromosome, we considered sites with the ChIP-nexus profile height above the 90th percentile as true binding sites (Figure 4.14C). Since the number of binding sites depends on the used cutoff, we treated the evaluation as a ranking or prioritization problem. Indeed, motif instances derived by CWM scanning prioritized more binding sites with high ChIP-nexus counts and hence a lower false positive rate compared to PWM scanning (Figure 4.14D). This difference is especially profound for the short Nanog motif. Even though the CWM has the same length as the PWM, the contribution scores scanned by the CWM already consider the context of the motif. Hence, the Nanog motif can get a higher contribution score if it is present in the vicinity of other $\sim$10bp spaced Nanog motifs. Hence, our approach of scanning the contribution scores using the CWM (instead of the raw sequence using the PWM) greatly reduces the false positive sites while still following the familiar scanning procedure as with PWMs.

We also compared the motif instance scoring to ChExMix, which directly uses the profile information from the ChIP-nexus data to determine motif instances. As expected, ChExMix recalls more binding sites with high ChIP-nexus counts. However, we note that this comparison is circular since the profile information used to evaluate the motif instances is also used to call them. By contrast, the CWM scanning of BPNet relies only on the DNA sequence since BPNet has never seen the ChIP-nexus data from the held-out chromosomes. Interestingly, we observe that ChExMix saturates at a range from

500 to 1200 called motif instances with high profile scores, whereas the CWM canning is able to recall more binding sites in total. Low statistical power of peak callers such as ChExMix leads to a lower number of binding events and is caused by the limited amount of mapped reads despite the overall high sequencing depth (>100M).

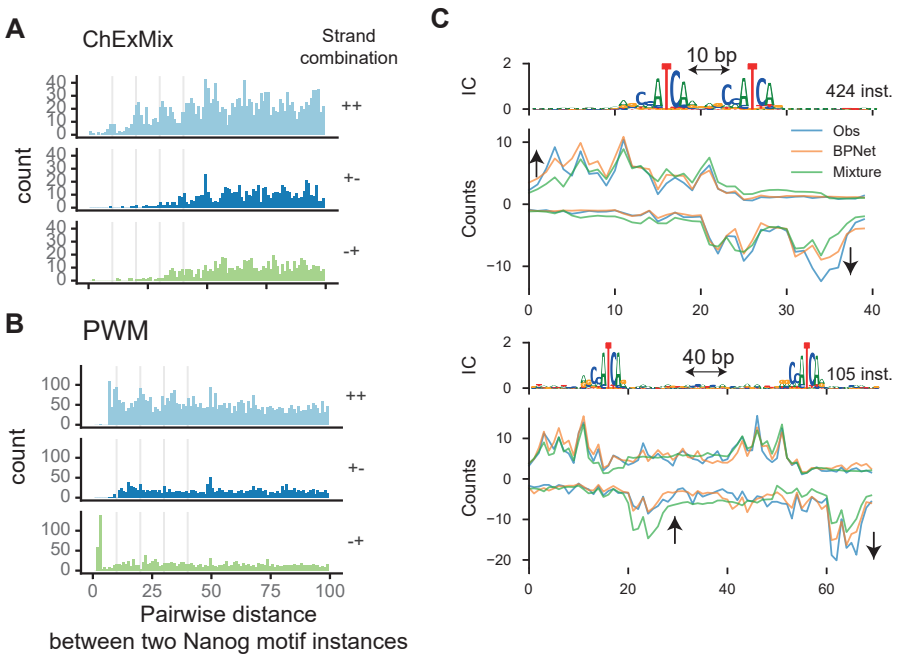


Figure 4.15: Motif instances mapped by ChExMix and PWM scanning do not show 10 bp periodicity of Nanog. A,B) Pairwise spacing of motif instances called by ChExMix (top) and PWM scanning (bottom) in the genome for Nanog-Nanog pairs. **C)** PFM and the aggregate ChIP-nexus footprint (Obs in blue) for all Nanog-Nanog pairs with the same orientation spaced at 10bp (top) or 40bp (bottom) as discovered by BPNet. Average BPNet prediction (orange) and the mixture of two individual average Nanog footprints (green) were scaled to have the same number of total counts as the observed profile.

Next, we asked whether the higher false positive rate of PWM scanning or the limited number of motif instances discovered by ChExMix impair the discovery of 10bp Nanog-Nanog spacing in the genome as discovered by BPNet. Indeed, we found that the Nanog-Nanog pairwise spacing histograms showed only weak signs of 10bp periodicity for both methods (Figure 4.15A,B). For PWM scanning, the high false discovery rate of motif instances likely prevents the detection of Nanog's 10bp periodicity. For ChExMix, we observed a depletion of instances below 40bp where most of the spatially constrained Nanog instances were seen in the Figure 4.8. This depletion of motif instances at close proximity could be due to two reasons. First, the optimized likelihood of ChExMix is non-convex and hence the global optimum might be difficult to find and the final discovered solution is determined by the initial conditions. Second, the key assumption of ChExMix is that the tag distribution (i.e. the average profile) is constant. This is an

oversimplification since ChIP-nexus profiles change their form if TFs are cooperatively bound on the DNA. For example, Figure 4.15C shows the difference between the observed average footprint of two nearby Nanog motifs (blue) and the mixture of the individual two motifs (green). If the two Nanog motifs are frequently co-bound as a homo-dimer, the inner parts of the dimer will be less accessible by exonuclease resulting in lower number of cut sites in the inner peaks compared to the outer peak sites. Hence, the ChIP-nexus profiles of co-bound TFs can be less accurately represented by the mixture of the two independent ChIP-nexus profiles as modelled by ChExMix. Interestingly, BPNet (shown in orange) does not simply model the data as a mixture and correctly captures the described phenomena.

4.3.5.2 Contribution scores of binary classification deep neural networks show lower motif instance mapping accuracy than BPNet

Similarly to comparing the binary classification to the profile model in terms of the bottleneck layer predictiveness, we asked whether the contribution scores of the profile regression model highlight additional motifs compared to the binary classification model. We computed the DeepLIFT contribution scores for each TF (pre-sigmoid activation) and ran TF-MoDISco in the same regions with the same hyper-parameters as previously done for BPNet. As expected, TF-MoDISco using the contribution scores of the binary classification model discovered a subset (16/20) of motifs with high seqlet support compared the profile regression model (Figure 4.16A). The 4 missed motifs, Oct4 monomer, Nanog mixture, B-Box and one TE, are hence not frequently used by the model to predict the presence or absence of the peak as they might co-occur with other more predictive motifs. We note that high reproducibility of the discovered motifs using two different models trained on similar but different data demonstrate the robustness of TF-MoDISco.

To compare the accuracy of motif instances calling in the genome for the 4 cognate motifs (Oct4-Sox2, Sox2, Nanog and Klf4) discovered by TF-MoDISco for both models, we performed the instance ranking analysis as for ChExMix considering sites with high ChIP-nexus profile as valid binding sites. The contribution scores of both models yielded a similar recall of Oct4-Sox2 and Sox2 motifs with high ChIP-nexus profiles (Figure 4.16B). We speculate that since the two motifs are linked to the pioneering activity, the binding sites will be important for binary classification and will hence not be missed by the binary classification model. Strikingly, the BPNet contribution scores of motif instances recalled a much higher fraction of Nanog motifs with high ChIP-nexus profiles (Figure 4.16B). Since Nanog is frequently co-bound either as a homo-dimer or as a hetero-dimer with Sox2, the profile shape of ChIP-nexus contains rich information about this binding event. Hence, the BPNet model trained using profile regression is able to yield much more accurate contribution scores. Altogether, we observe that using the profile regression model can be trained three times faster, it discovers more motifs with strong seqlet support and it calls motif instances more accurately. Moreover, since the profile regression model predicts a rich ChIP-nexus profile, it provides much more

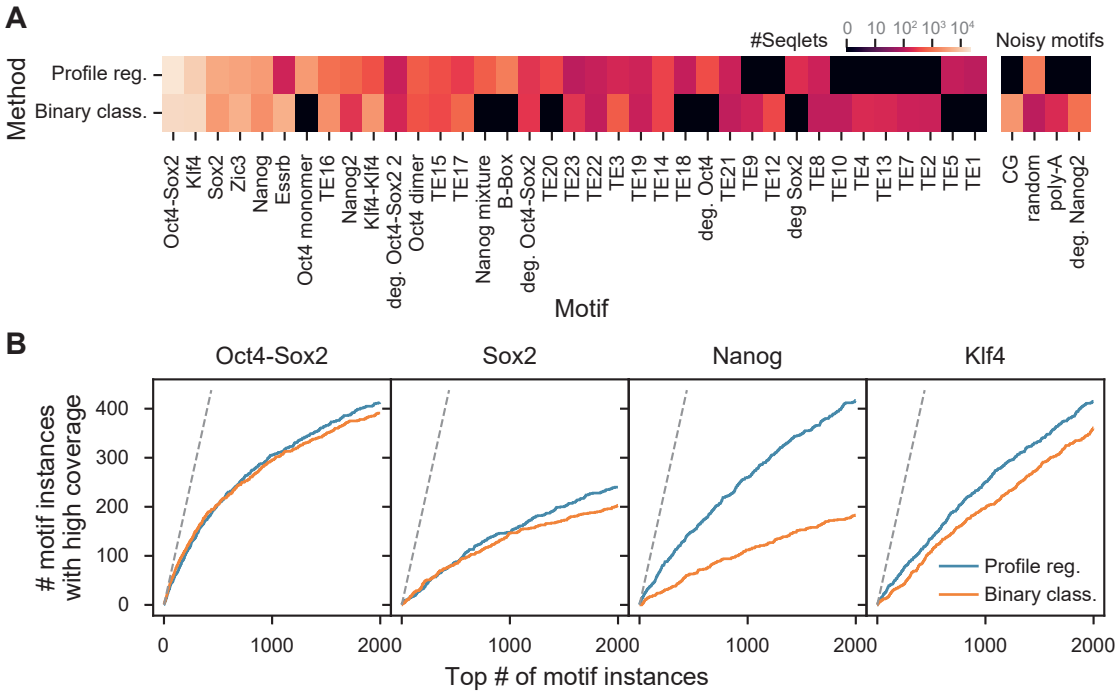


Figure 4.16: BPNet trained to predict the ChIP-nexus profile yields more accurate motif instances than a binary classification model. **A)** Detected motifs by TF-MoDISco using the contribution scores in ChIP-nexus peaks of BPNet (profile reg.) or the binary classification model (binary class.). Light color denotes a high number of seqlets for each motif. Motifs not discovered or motifs supported by less than 100 seqlets are shown in black. Potentially erroneous or noisy motifs are displayed separately on the right. **B)** Number of motif instances showing a ChIP-nexus footprint (y-axis) within the top N motif instances with highest contribution scores (x-axis) from the held-out (test) chromosomes 1, 8 and 9. A site was considered to show a ChIP-nexus footprint if the number of reads at the position where the aggregate footprint peaks (averaged across both strands) is higher than the 90% percentile value of all motif instances detected by the profile regression model for the corresponding TF.

information to study the directionality of TF binding and its grammar as shown in the Figure 4.12.

4.3.6 BPNet trained on ChIP-seq data yields a large fraction of motifs found by ChIP-nexus albeit at lower motif instance precision

As we have seen in the previous chapter, the BPNet model can be readily applied to ChIP-seq. Here we apply our interpretation workflow using DeepLIFT and TF-MoDISco to the BPNet model trained on ChIP-seq. First, we investigated the contribution scores in the known Oct4 enhancer computed in the exact same manner as for the ChIP-nexus model. Intriguingly, we found that the contribution scores also precisely highlighted the

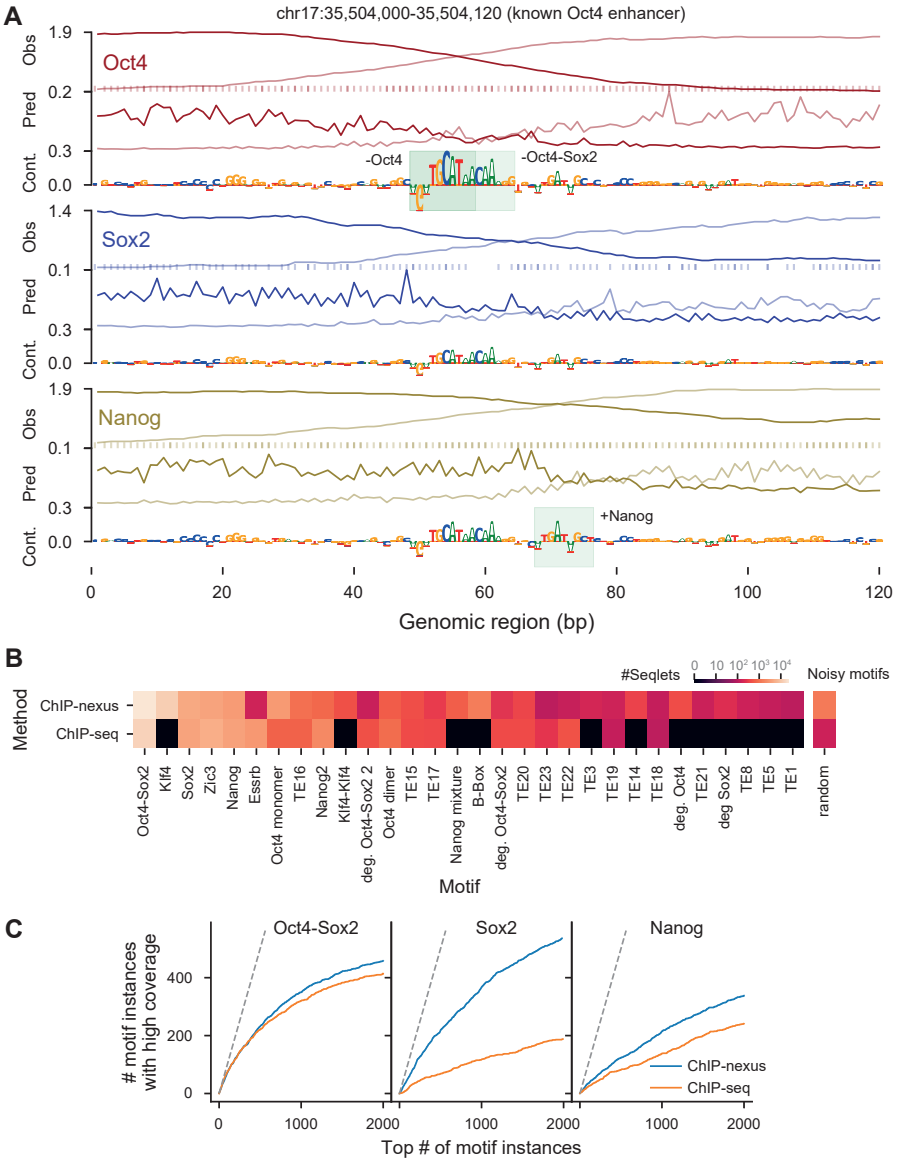


Figure 4.17: BPNet interpreted by DeepLIFT and TF-MoDISco is directly applicable to ChIP-seq. A) Observed and predicted read counts as well as the contribution scores of BPNet for the known Oct4 enhancer. The observed read counts are shown both as smoothed (line) and as raw counts (points at y=0). Motif instances derived by CWM scanning are highlighted with a green box. **B)** BPNet applied to ChIP-seq discovers the majority of the motifs as BPNet applied to ChIP-nexus data (see Figure 4.16A caption for detailed description). **C)** Motif instance localization has higher accuracy for BPNet trained on ChIP-nexus data than BPNet trained on ChIP-seq data (see Figure 4.16B caption for detailed description).

Oct4-Sox2 motif in the center and the Nanog motif on the side (Figure 4.17A). Hence, we were able to directly apply the BPNet model to ChIP-seq data and have obtained accurate predictions as well as contribution scores highlighting the expected regulatory motifs.

To test which motifs were learned by BPNet applied to ChIP-seq, we used TF-MoDISco with the same hyper-parameters as before. We observed that TF-MoDISco discovered a high fraction of the exact same motifs and only missed some motifs (Figure 4.17B). This suggests our entire BPNet workflow, which includes BPNet training, motif discovery with TF-MoDISco, and determining motifs instances using CWM scanning, can be readily applied to ChIP-seq data. To determine the quality of the motif instances derived by BPNet (example motif instances shown in Figure 4.17A), we performed the same motif instance prioritization analysis as before in Figure 4.14D and Figure 4.16B. We observed that ChIP-nexus recalled a higher fraction of motifs with high profile compared to ChIP-seq.

Altogether, we find that BPNet applied to ChIP-seq is almost as good for motif discovery as BPNet applied to ChIP-nexus data. However the motif instances from BPNet applied to ChIP-seq are not as good as those of BPNet applied to ChIP-nexus data. Moreover, determining whether the motif is directly or indirectly bound is hard if not impossible with ChIP-seq due to a much lower resolution.

4.4 Discussion

We created a versatile tool for regulatory code discovery. Highly accurate BPNet model together with the deep learning interpretation toolbox discover a broad range of motifs including TEs and map the individual motif instances to the genome at an unprecedented accuracy. The motif instances contain, in addition to the sequence mismatch score, a contribution score which is much more predictive for TF occupancy of the motif than the PWM match. This allows us to study the architecture of the enhancers as well as the mechanism controlling binding and hence also transcriptional activation.

Our computational approach follows a radically different paradigm. We first regress directly to raw experimental data (i.e. ChIP-nexus profile) using a flexible predictive model (convolutional neural network) and then define a custom contribution score that captures the measured quantity of interest (e.g. TF binding occupancy via spikiness of the ChIP-nexus profile). Doing so, the predictive model can learn subtle variation in the raw experimental data which are difficult to capture using hand-crafted rules as used by peak-callers. This results in a richer set of discovered motifs and greatly reduced false discovery rate of motif instances in the genome. Finally, since no explicit assumptions are made about the profile shape, and since experimental biases can be easily controlled for by taking the control experiments (e.g. input control in ChIP-seq) into account, our approach is also applicable to other genomics assays that contain profile information, including ChIP-seq, ATAC-seq, DNase-seq or CUT&RUN.

A critical assumption of using the model as an 'oracle' to study the regulatory grammar is that the model captures the causal relationship between sequence and the ChIP-nexus

profile. While this assumption should always be questioned and experimentally tested, we argue here that in the context of our data and model training, this is a fairly valid assumption. We predict the profile of high-resolution ChIP-nexus data, which should depend exclusively on the local sequence seen by the model. Since we use a large amount of widely varying local sequences to train the model, possible confounding factors are captured in the model input, and thus the learned model should directly represent a causal relationship between sequence and binding profile. In contrast to the profile predictions, the predictions of the total amount of counts in the region are confounded by other factors such as DNA accessibility which can not be fully accounted for using the local sequence. Chromatin accessibility is not only dependent on the local sequence but also on the larger chromatin context. As the output depends on a larger and larger sequence window, this becomes a bigger and bigger issue. For example, in genome-wide association studies (GWAS) where a single value is measured for the whole genome, the learned models can be confounded by other variants in linkage. Altogether, the high resolution and locality of ChIP-nexus profile shapes make it attractive to learn near-causal models which can be effectively interpreted to reveal the cis-regulatory code.

5 Kipoi: Platform for exchanging predictive models in genomics

The results and argumentation presented in this chapter are based on the manuscript 'The Kipoi repository accelerates community exchange and reuse of predictive models for genomics' [1]. All parts of the manuscript except figure captions have been reformulated and significantly extended to provide more technical details. Author contributions for the original publication are described in Section .

5.1 Motivation

Predictive models are key to understand the regulatory DNA sequences in the genome. As outlined in the introduction, predictive models have been successfully used to predict various molecular phenotypes from DNA sequence including transcription factor binding, chromatin accessibility, and splicing efficiency [88, 62, 63, 75, 89, 90, 91]. Predictive models have been also used to call variants from whole genome sequencing data [206, 207], and estimate the CRISPR guide activity [208, 209]. Once trained, such models allow to probe the learned relationships among data modalities *in silico*, which among other applications, enables interpreting functional non-coding variants and rationalizes the design of synthetic genes.

However, despite the success and importance of predictive models in genomics, the lack of standards and limited centralized access to trained models have hampered their practical impact. Trained predictive models are made available by the authors in various formats via scattered channels, including supplementary material of articles, author-maintained web pages and code repositories. Additionally, the provided code is often not tested which introduces additional barriers for the users. By contrast, there are well established standards and centralized repositories for sharing code such as Bioconductor [210] or repositories for sharing raw data such as Gene Expression Omnibus (`https://www.ncbi.nlm.nih.gov/geo/`), ArrayExpress (`https://www.ebi.ac.uk/arrayexpress`), and European Nucleotide Archive (`https://www.ebi.ac.uk/ena`). These repositories have greatly improved the productivity of the entire scientific community. To fully leverage the potential of predictive models build by the community, sharing predictive models should follow the same FAIR principle as originally proposed for sharing data [211]. Namely, trained models should be Findable, Accessible, Interoperable, and Reusable (FAIR).

Repositories of trained models in other fields such as computer vision and natural language processing have enabled users to rapidly develop accurate models on small new datasets via transfer learning (Section 2.3.7). However, the software infrastructure

of these repositories has to be re-designed to fulfil the needs of predictive models in genomics. Specifically, the model repository in genomics needs to be easy to use and well documented to serve practitioners without expert knowledge in machine learning. Additionally, the repository should support not just a single machine learning framework, but any machine learning framework including custom code. Furthermore, data-loading and pre-processing should be part of the model. Finally, downstream functionalities such as variant effect prediction or computing the feature contribution scores should be available for the majority of models in the repository.

Here, we present Kipoi (Greek for gardens, pronounced "Kípi"), a platform for sharing trained models in genomics that follows the FAIR principle and thereby fosters their re-use.

5.2 Kipoi infrastructure

5.2.1 Overview

Kipoi consists of three main components: model definition (Figure 5.1, left), model repository (Figure 5.1, center) and the API to access and use models (Figure 5.1, right). Model definition describes two key components: data-loader and predictive model. The role of the data-loader is to extract data from raw files and to pre-process them. Predictive model, which is a parameterized function, takes as input the output of the data-loader and makes predictions. The Kipoi model definition further specifies required dependencies, provides information about the original publication, the distribution license, and describes the model inputs and outputs. Model repository stores the trained models and makes them easily accessible. The Kipoi repository currently contains over 2,000 individual trained models from 23 distinct studies covering key predictive tasks in genomics from DNA sequence such as transcription factor binding, alternative splicing, and chromatin accessibility. The models can be browsed and searched on Kipoi's web site: `www.kipoi.org`. Finally, Kipoi provides an API (Figure 5.1, right) to load the models from the repository and to use them for downstream applications such as comparing models or transfer learning.

Components of Kipoi such as the API or the repository leverage standard software development tools such as i) Conda–popular package manager for any programming language with a rich repertoire of bioinformatics packages thanks to the Bioconda distribution[212], ii) GitHub-a software repository with issue tracking, iii) CircleCI-continuous integration platform for testing models and iv) Zenodo–file archival service for storing model parameters. By using GitHub and Zenodo, the contributed models are tracked, indexed, and archived to facilitate reproducible science [213].

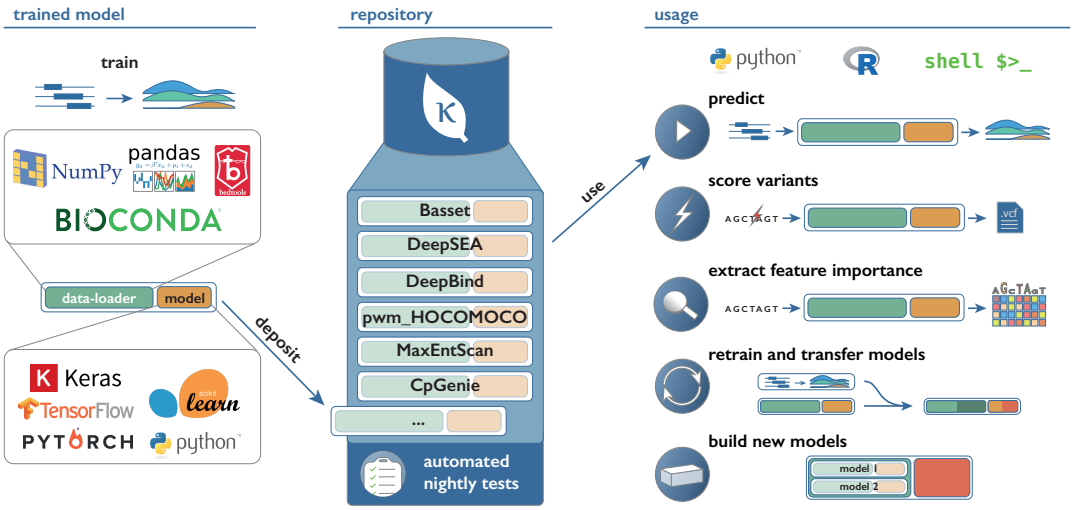


Figure 5.1: Overview of Kipoi. From left to right: At its core, Kipoi defines a programmatic standard for data-loaders and predictive models. Data-loaders translate genomics data types into numeric representation that can be used by machine learning models. Kipoi models can be implemented using a broad range of machine learning frameworks. The Kipoi repository allows users to store and retrieve trained models together with associated data-loaders. Kipoi models are automatically versioned, nightly tested and systematically documented with examples for their use. Kipoi models can be accessed through unified interfaces from python, R, and command line. All models and their software dependencies can be installed in a fully automatic manner. Kipoi streamlines the application of trained models to make predictions on new data, to score variants stored in the standard genetic variant file format, and to assess the effect of variation in the input to model predictions (feature contribution score). Moreover, Kipoi models can be adapted to new tasks by either retraining them, or by building new composite models that combine existing ones. Newly defined models can be deposited in the repository. Taken from [1].

5.2.2 Model definition

5.2.2.1 Model class

Model in Kipoi is a parameterized function specified as a python class containing the `predict_on_batch(x)` method. The `predict_on_batch` should take as input and also return a single numpy array, a list of numpy arrays or a dictionary of numpy arrays. This class can be directly implemented by the contributor. Additionally, a pre-made model can be used for models implemented in machine learning frameworks including Keras, TensorFlow, PyTorch and Scikit-learn. The pre-made models for deep learning frameworks provide two additional methods: `predict_activation_on_batch(x, layer, ...)`, which returns the feature activation map of an intermediary layer (useful for transfer learning) and `input_grad(x, ...)`, which returns the gradient of the input with respect to model's predictions (useful for feature contribution scores). New pre-made

models for other machine learning frameworks can be easily added by implementing the aforementioned class.

5.2.2.2 Data-loader class / function

The role of the data-loader is to iteratively extract data from files, process them and return a nested dictionary containing numpy arrays with three keys at the root: inputs, targets (optional), and metadata (optional). Inputs are consumed by the model to make predictions, targets can be optionally used to train or benchmark the model, and metadata provide additional information about the extracted samples such as the genomic interval of the extracted sequence.

Data-loader can be implemented in multiple differnet ways. The canonical way is to implement a Dataset class containing the `__len__` method returning the dataset length and the `__getitem__` method returning either a single data example or a batch of examples. This implementation allows to load the data in parallel using multiple processes by leveraging the Data-Loader class implemented in PyTorch. Other implementation ways include a python function loading the whole dataset or a python generator/iterator loading either batches or single examples of the dataset (`http://kipoi.org/docs/contributing/04_Writing_dataloader.py/`). Regardless of the implementation, the following methods will be available for each data-loader:

- `batch_iter` returning batches of data (dictionary with inputs, targets and metadata keys)
- `batch_train_iter` returning batches of data indefinitely as a tuple of inputs and targets (directly useful with the Keras' `fit_generator`),
- `batch_predict_iter` returning batches of inputs and `load_all` returning the whole dataset.

Similar to pre-made models, we provide efficient pre-made data-loaders for sequence based models in the Kipoiseq python package (`https://github.com/kipoi/kipoiseq/`). The data-loaders in Kipoiseq are implemented using simple building blocks such as *extractors*, which extract data from files, and *transforms*, which transform differnet objects (e.g. one-hot-encode DNA sequence or resize a genomic interval). These components are also available in Kipoiseq. This modularity allows users to easily build new data-loaders by mix and matching the available components.

5.2.2.3 model.yaml

In addition to defining the model class, other elements such as required dependencies, test datasets or parameter files are specified in the `model.yaml` file. The following code snipped shows one such `model.yaml` file.

```
1  defined_as: kipoi.model.KerasModel
2  args:
3      weights:
```

```
4           url: https://zenodo.org/record/1452399/files/model.weights.h5?download=1
5           md5: 2a0ae0a29337eb8106d65e1baeda85d1
6       arch:
7           url: https://zenodo.org/record/1452399/files/model.arch?download=1
8           md5: 6903bcab337a6753ad010f43f208df42
9       backend: tensorflow
10      image_dim_ordering: tf
11
12  default_dataloader:
13      defined_as: kipoiseq.dataloaders.SeqIntervalDl
14      default_args:
15          auto_resize_len: 1000
16
17  # required pip and conda dependencies
18  dependencies:
19    conda:
20      - python>=3.5
21      - h5py
22      - bioconda::pyfaidx
23    pip:
24      - tensorflow <=1.4.1
25      - keras==1.2.2
26      - kipoiseq
27
28  test:
29    expect:
30      url: https://s3..../kipoi-models/predictions/.../<model>/predictions.h5
31      md5: 62da0ac731f323ea54ee6e30c38e0722
32    precision_decimal: 5
33
34  schema:
35      inputs:
36          shape: (1000,4)
37          doc: 1000 base pair sequence of one-hot encoding ACGT
38      targets:
39          shape: (421,)
40          doc: Probability for chromatin accessibility [0,1]
41          column_labels: task_names.txt
42
43  # information about the model. Rendered on kipoi.org
44  info:
45    authors:
46        - name: My Name
47          github: myname
48    name: MyKerasModel
49    doc: >
50
51      Description of the model shown on kipoi.org.
52    cite_as: https://doi.org/x/y   # link to the paper, blog post, ...
53    tags:  # under which category dos this model fall
54      - DNA accessibility
```

Listing 5.1: Example model.yaml definition.

There are 6 key components in the model.yaml file over which we will go in detail next: model definition, data-loader definition, dependencies, test file, input/output schema, and general information.

Model definition

```
1   defined_as: kipoi.model.KerasModel
2   args:
3       weights:
4           url: https://zenodo.org/record/1452399/files/model.weights.h5?download=1
5           md5: 2a0ae0a29337eb8106d65e1baeda85d1
6       arch:
7           url: https://zenodo.org/record/1452399/files/model.arch?download=1
8           md5: 6903bcab337a6753ad010f43f208df42
9       backend: tensorflow
10      image_dim_ordering: tf
```

Listing 5.2: Model definition.

First is the definition of the model specified by `defined_as` and `args` fields. The `defined_as` can either refer to a pre-made model (such as `kipoi.model.KerasModel`) or to a custom model implemented by the user (e.g. `model.MyModel` implemented in `model.py`). The arguments to that model (`args`) can be either directly specified in the yaml file (`backend: tensorflow`) or it can contain a URL to the file. For example, the argument `weights` contains a publicly accessible URL pointing to model parameters file (`model.weights.h5`) hosted on a file-sharing platform providing a stable URL such as Zenodo. Upon model request, these files get downloaded and validated using the specified MD5 hash (`md5`).

Data-loader definition

```
default_dataloader:
    defined_as: kipoiseq.dataloaders.SeqIntervalDl
    default_args:
        auto_resize_len: 1000
```

Listing 5.3: Data-loader definition.

Data-loader can be either implemented by the user or a pre-made data-loader from Kipoiseq can be used. In this example, a `SeqIntervalDl` data-loader from Kipoiseq is shown. `SeqIntervalDl` requires two files to load the data: a BED3 file containing genomic intervals and a FASTA file of the reference genome. It returns one-hot-encoded DNA sequences of length 1000 under the `inputs` key in the returned dictionary.

Dependencies

```
dependencies:
  conda:
    - python>=3.5
    - h5py
    - bioconda::pyfaidx
  pip:
    - tensorflow <=1.4.1
    - keras==1.2.2
    - kipoiseq
```

Listing 5.4: Dependency definition.

The package dependencies required by the model and the data-loader can be installed either through the Conda package manager (`https://conda.io`) or through the pip package manager (`https://pypi.org/project/pip/`). Specific version restrictions can be specified (e.g. `tensorflow<=1.4.1`). Packages from specific Conda channels such as Bioconda (`https://bioconda.github.io/`) or conda-forge (`https://conda-forge.org/`), can be easily specified by prefixing the package name with `<<channel>>::` (e.g. `bioconda::pyfaidx`). Thanks to these package channels, a large set of dependencies including the frequently used bioinformatics packages (e.g. pyfaidx, pysam, pybedtools) are easily installable. Since Conda is a package any programming language (not just python), models implemented in other languages can be easily installed. Moreover, since any command-line tool can be called from python, a model in Kipoi can also wrap existing command-line tools implemented in other languages. One such example is

lsgkm-SVM (v0.0.1) from the Bioconda channel which is also wrapped as a Kipoi model (`lsgkm-SVM`).

Test file

```
test:
  expect:
    url: https://s3..../kipoi-models/predictions/.../<model>/predictions.h5
    md5: 62da0ac731f323ea54ee6e30c38e0722
  precision_decimal: 5
```

Listing 5.5: Test file specification.

To test whether the predictions indeed match the expected ones, a URL to a test HDF5 file can be specified (test.expect). This file contains the inputs and expected model predictions. When testing the model, the predictions should match the expected predictions for at least `precision_decimal` significant digits.

Input/output schema

```
schema:
    inputs:
        shape: (1000,4)
        doc: 1000 base pair sequence of one-hot encoding ACGT
    targets:
        shape: (421,)
        doc: Probability for chromatin accessibility [0,1]
        column_labels: task_names.txt
```

Listing 5.6: Input/output schema definition.

For transparency and testing purposes, a model.yaml should also specify the shapes of the input numpy arrays and the predicted numpy arrays. In the example above, the model takes as input a numpy arrays of shape (B, 1000,4) and returns a numpy array of shape (B, 412), where B is the batch size (32 by default, can be specified when making model predictions).

General information

```
# information about the model. Rendered on kipoi.org
info:
  authors:
      - name: My Name
        github: myname
  name: MyKerasModel
  doc: >

    Description of the model shown on kipoi.org.
  cite_as: https://doi.org/x/y   # link to the paper, blog post, ...
  tags:  # under which category dos this model fall
    - DNA accessibility
```

Listing 5.7: General information.

The general information about the model include the list of authors, short documentation, DOI link to the publication featuring this model and different tags for easier navigation.

5.2.3 Model repository

The `model.yaml` files are stored in the git repository at `https://github.com/kipoi/models`. A folder containing the model.yaml file is considered as a single model. In model's directory, additional files such as `model.py` for custom model implementation, `dataloader.py` for custom data-loader implementation or `labels.txt` annotating model predictions can be added. As mentioned earlier when describing the model definition, the required files such as model weights should be accessible via stable publicly available URL link. To assure reproducibility at all times, we encourage the contributors to host the files on archiving services such as Zenodo (`https://zenodo.org/`) or Figshare (`https://figshare.com`). All updates of the model repository are tracked through git and the repository is regularly archived to Zenodo (`https://zenodo.org/record/1637796`).

5.2.3.1 Specifying multiple models via templates

To simultaneously specify multiple very similar models (say the same model trained for different transcription factors), we allow contributors to implement a `model-template.yaml` file and `models.tsv`. `models.tsv` lists all the models and provides variables that change across the differnet models (say a URL link to model weights). For each row in the `models.tsv` file, a `model.yaml` is created by populating the `model-template.yaml` with variables specified in the `models.tsv` file. One example is the CpGenie model group containing models in trained on the data from different cell-lines:

```
type: keras
args:
  weights:
    url: {{ weights_url }}
    md5: {{ weights_md5 }}
```

Listing 5.8: model-template.yaml

```
model                  weights_url                                  weights_md5
A549_ENCSR000DDI       https://.../A549_ENCSR000DDI.h5              6d3a971ce766128ca444dd70ef76df70
BE2C_ENCSR000DEB       https://.../BE2C_ENCSR000DEB.h5              919b2f7f675bebb9217d95021d92af74
```

Listing 5.9: models.tsv

5.2.3.2 Using custom model repositories

In addition to the default model repository or source (`https://github.com/kipoi/models`), the user can host and seamlessly use other model sources. These are specified in the `~/.kipoi/config.yaml` file and are treated equivalently to the default model source.

```yaml
model_sources:
  kipoi:
    type: git
    remote_url: git@github.com:kipoi/models.git
    local_path: /home/avsec/.kipoi/models/

  my_git_models:
    type: git
    remote_url: git@github.com:asd/other_models.git
    local_path: ~/.kipoi/other_models/

  my_local_models:
    type: local
    local_path: /data/mymodels/
```

Listing 5.10: Kipoi global config file example located by default at ∼/.kipoi/config.yaml.

5.2.3.3 Depositing or updating models

Since model definitions—`model.yaml` files—are stored in a git repository, models are added or updated by issuing a pull request to the Kipoi model repository `https://github.com/kipoi/models`. The non-source files should be uploaded to the file-sharing platforms such as Zenodo or Figshare. The URLs of the uploaded files are specified in the `model.yaml` file. Newly added models are reviewed by the member of the Kipoi core team to ensure model quality, naming guidelines scope, and appropriate specification of the distribution license.

5.2.3.4 Testing models

Newly added or updated models via GitHub pull-requests are tested using the CircleCI continuous integration service. Every day, all model groups are tested in the repository's master branch by running tests on one or more selected models for every model group. For each tested model, the following steps will be performed:

1. Check that the yaml files are correctly formatted and contain all necessary fields.
2. Install a new Conda environment with all required dependencies as specified in `model.yaml`.
3. Run the whole model prediction pipeline for the example files specified together with the data-loader.
4. Run model predictions for the test file containing inputs and expected predictions as described in Section 5.2.2.3. Compare predictions to expected predictions.

5.2.4 API

To access models stored in the model repository consisting of `model.yaml` files, Kipoi provides a python API. The python API can be installed as a python package using pip or Conda package manages (`pip install kipoi` or `conda install -c bioconda kipoi`). Here are the main commands provided by the Kipoi python API:

```python
import kipoi

kipoi.list_models() # list available models

model = kipoi.get_model("Basset") # load the model

model = kipoi.get_model(  # load the model from a past commit
    "https://github.com/kipoi/models/tree/<commit>/<model>",
    source='github-permalink'
)

# main attributes
model.model # wrapped model (say keras.models.Model)
model.default_dataloader # dataloader
model.info # description, authors, paper link, ...

# main methods
model.predict_on_batch(x) # provided by all models
model.pipeline.predict(dict(fasta_file="hg19.fa",
                            intervals_file="intervals.bed"))
# runs: raw files -[dataloader]-> numpy arrays -[model]-> predictions
```

Listing 5.11: Kipoi's python API.

The python API can be also accessed from the R programming language by using the `reticulate` R package (`https://rstudio.github.io/reticulate/`). In addition to the python API, Kipoi also provides a command line interface exposing the `kipoi` command highlighted in the next section (Results).

5.2.4.1 Dependency installation on the target machine

To assure that the users will be able to easily run model predictions without any installation hurdles, Conda virtual environments can be created for each model using the `kipoi env create` command. In addition to installing a Conda environment for each model, Kipoi provides shared Conda environments covering multiple models. Currently, we provide and test two shared environments which cover 19/23 model groups. After installing the model environments (shared or individual), the right environment for each model can be queried using the `kipoi env get <model>` command.

As an alternative to Conda environments, Kipoi also provides a Singularity container [214, 215] and a Docker container with all necessary Conda environments installed. Model predictions can be executed inside the Singularity container by adding the `--singularity` flag to the `kipoi predict` or `kipoi veff score_variants` command. When using the Singularity container, the user only has to install kipoi and the Singularity command line tool (version $\geq$2.5) to run predictions for any model. The Docker container is available at `https://hub.docker.com/r/kipoi/models/`. Using the Docker container, users can also make predictions on the Windows operating system which is currently not supported by the Kipoi core team due to the limited availability of Conda packages built for Windows.

5.2.4.2 Variant effect prediction and model interpretation plugins

In addition to the core Kipoi package providing tools to access models and make predictions, we provide additional plugins implemented as python packages. Specifically, we provide the variant effect prediction plugin (`https://github.com/kipoi/kipoi-veff`) performing in-silico mutagenesis (ISM) (Figure 2.6B) and a model interpretation plugin for computing feature contribution scores (`https://github.com/kipoi/kipoi-interpret`) described in Section 2.3.8.

5.3 Results

5.3.1 Benchmarking alternative models predicting transcription factor binding

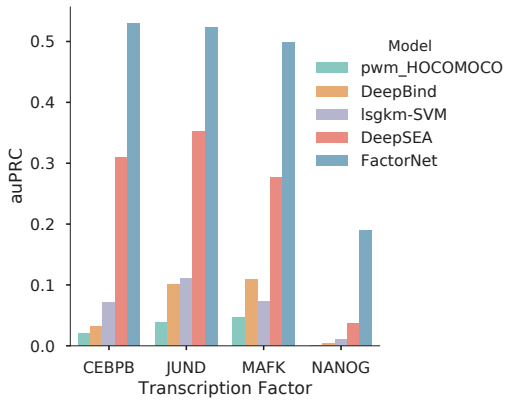
Kipoi provides access to a broad spectrum of models through a common API thereby facilitating systematic comparative evaluation of models for the same prediction task. We used the Kipoi API to benchmark several popular models for predicting genome-wide DNA binding profiles of transcription factors (TFs). As mentioned in the introduction, accurate model prediction are desired for various downstream tasks including variant effect prediction or imputation.

Majority of TF binding models in the literature focus on learning DNA sequence preferences of TFs. The position weight matrix (PWM) is the simplest and most popular model that records the position-specific preference score of a TF across a predefined length of sequence for every nucleotide at each position. A PWM can be used to score the affinity of a TF to any subsequence matching the length of the PWM as the sum of the PWM's position-specific preference scores for every observed base in the subsequence. Several generative and discriminative models have been developed to learn PWMs from in vitro and in vivo TF binding experiments [13]. We ported PWMs for 600 human TFs from the HOCOMOCO database [93] into the Kipoi repository. More recently, supervised machine learning approaches have been used to learn more complex non-linear models of TF sequence preferences. lsgkm-SVM is a support vector machine classifier that predicts the probability of binding from DNA sequence inputs represented as a collection of gapped k-mers, i.e. words of a predefined length k allowing for gaps [94]. The SVM model is trained on TF ChIP-seq data to discriminate high confidence bound regions in the genome from unbound background regions. We replicated lsgkm-SVM models for several TFs from the original publication and deposited these in the Kipoi repository. Convolutional neural networks (CNNs) have also been used to learn discriminative sequence models of TF binding from ChIP-seq data. We ported all 927 single-task DeepBind CNN models [63] into the model model repository. We also ported a multi-task CNN model called DeepSEA [63], which can jointly predict binding of multiple TFs to input DNA sequences. The above mentioned models use only DNA sequence as inputs. Because the DNA sequence is the same across different cellular contexts, these sequence-only models of TF binding cannot predict different in vivo TF binding landscapes in new cell types not used during training. More recently, a multi-

a

Model	Publication	Type	Input modalities	Framework	Size	Prediction time (n=256)	Input seq. length
pwm_HOCOMOCO	Kulakovskiy et al 2015	Position weight matrix scan	DNA sequence	Keras	16KB	1.2ms	101bp
DeepBind	Alipanahi et al 2015	Convolutional neural network	DNA sequence	Keras	36KB	2.4ms	101bp
lsgkm-SVM	Ghandi et al 2014	Support vector machine	DNA sequence	LS-GKM	4MB	10s	101bp
DeepSEA	Zhou *et al 2015*	Convolutional neural network	DNA sequence	PyTorch	211MB	94ms	1000bp
FactorNet	Quang *et al* 2017	Convolutional and recurrent neural network	DNA sequence + DNase + annotation	Keras	3-13MB	118ms	1002bp

b



c

```
# Create and activate a new
# conda environment with all
# model dependencies installed
kipoi env create <Model>
source activate kipoi-<Model>

# Run model prediction and save
# the results sequentially into
# an hdf5 file
kipoi predict <Model> \
  --dataloader_args='{
  "intervals_file": "intervals.bed",
  "fasta_file": "hg38.fa"}' \
  -o '<Model>.preds.h5'
```

Figure 5.2: Applying and benchmarking alternative Kipoi models for transcription factor binding prediction. (a) Five models for predicting transcription factor binding based on alternative modeling paradigms: i) predefined position weight matrices contained in the HOCOMOCO database [93]; ii) lsgkm-SVM [94], a support vector machine classifier; iii) the convolutional neural network DeepBind [62]; iv) the multi-task convolutional neural network DeepSEA [63]; v) FactorNet [75], a multimodal deep neural network with convolutional and recurrent layers that further integrates chromatin accessibility profile and genomic annotation features. Models differ by i) the size of genomic input sequence, where DeepSEA [63] and FactorNET [75] consider ∼1 kb sequence inputs, whereas other models are based on ∼100 bp, and ii) parametrization complexity with the total size of model parameters ranging from 16 kB (pwm_HOCOMOCO) to 211 MB (DeepSEA). **(b)** Performance of the models in **a** for predicting ChIP-seq peaks of four transcription factors on held-out data (chromosome 8), quantified using the area under the precision-recall curve. More complex models yield more accurate predictions than the simpler models such as the commonly used PWMs. **(c)** Example use of Kipoi from the command line to install software dependencies, download the model, extract and pre-process the data, and write predictions to a new file. Results as shown in **b** can be obtained for all Kipoi models listed in **a** using these generic commands by varying the placeholder <<Model>>. Taken from [1].

modal neural network model called FactorNet [75] was developed to predict genome-wide cell-type specific in vivo TF binding maps by jointly modeling DNA sequence with

chromatin accessibility (DNase-seq) profiles and gene expression (RNA-seq) data from the target cell type. FactorNet uses convolutional layers and recurrent layers in the neural network model. These five models were evaluated on different test sets in their respective publications. Hence, a direct comparison of their prediction performance on identical benchmarking test data sets has been lacking.

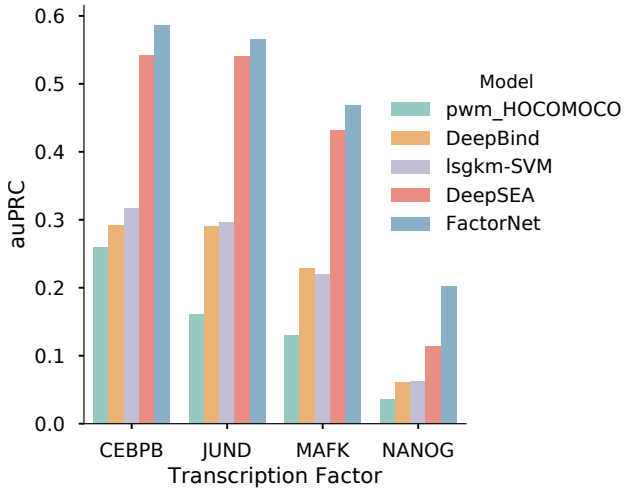


Figure 5.3: Performance of models highlighted in Figure 5.2 for predicting ChIP-seq peaks of four transcription factors on held-out data (chromosome 8) restricted to accessible chromatin regions as measured by DNase (Section A.4.2.1). This shows that DeepSEA and FactorNet perform similarly when model evaluation is restricted to bound and unbound regions that strictly overlap accessible chromatin regions. Taken from [1].

Since all five models were ported to the Kipoi repository, we were able to use the generic kipoi predict command to obtain predictions on a common set of benchmark test datasets (Figure 5.2). We used four benchmark TF ChIP-seq datasets: CEBPB in HeLa-S3, JUND in HepG2, MAFK in K562, and NANOG in H1-hESC. All models except lsgkm-SVM were obtained directly from their respective publications and hence trained by the authors. With the exception of the PWMs, all other models were trained using supervised learning algorithms to discriminate bound bins from different types of sampled background bins. The models also used different lengths of sequence inputs (Figure 5.2A). To test these models, the human genome was binned into contiguous, fixed-sized bins. Ground truth labels were obtained from ChIP-seq experiments targeting a TF in a specific cellular context. The ChIP-seq dataset was processed to identify high confidence binding sites in the genome thereby associating every genomic sequence bin with a binary label representing a bound or unbound state (Section A.4.4.2). We used all bins in chromosome 8 as our held-out test set because it was not used to train any of the supervised models. We used the area under the precision-recall curve (auPRC) as our evaluation measure to account for the significantly skewed class imbalance.

PWMs generally performed poorly across all TFs (Figure 5.2B). The poor performance is likely due to the simplicity of the PWM model and its inability to account for sequence features such as binding sites of other cooperating and competing TFs factors that affect in vivo binding of the target TF. DeepSEA consistently outperformed DeepBind and lsgkm-SVM. These results suggest that the models with greater capacity are beneficial. Further, longer input sequences and a better sampling strategy of unbound training sequences from accessible chromatin regions also appear to improve prediction performance. FactorNet obtains the highest chromosome-wide performance across all TFs highlighting the importance of explicitly integrating target cell-type specific chromatin accessibility profiles with DNA sequence for predicting in vivo TF binding (Figure 5.2B). This conclusion is also supported by the observation that DeepSEA and FactorNet perform similarly when model evaluation is restricted to bound and unbound bins that strictly overlap accessible chromatin regions (Figure 5.3).

Without Kipoi's unified API, comparing these disparate models would be a cumbersome task. The models are implemented using different software framework and formats, require different input file formats and return predictions in a different formats. Hence, model-specific data pre-processing code is required to standardize the evaluation. Additionally, installing the appropriate package dependencies for each model can be difficult and time consuming. Here, all the steps of the evaluation pipeline for the disparate models and benchmark datasets were executed using three simple commands (Figure 5.2C). These commands install a new Conda environment containing the required package dependencies, download the model from the repository, extract and pre-process the test data, make the model predictions, and write the results to a new file. Since data loading and model prediction are continuously executed on mini-batches of data, the predictions can easily scale to large datasets that might not entirely fit into the computer's main memory. Moreover, the majority of data-loaders in Kipoi support parallel data loading, which can drastically speed-up prediction time and allow for optimal utilization of graphical processor units (GPUs). Moreover, since these three commands are common to all models and since the predictions are stored in a standardized format, such model comparison can be easily scripted using workflow management tools such as Snakemake [216].

5.3.2 Improving predictive models of chromatin accessibility using transfer learning

Transfer learning is a machine learning approach where a model trained on one prediction task is reused as an initialization for a model that is to be optimized on a different but related task (Section 2.3.7). Transferred models typically learn new tasks more rapidly, require less data to train and generalize better to unseen data than models trained from scratch [217]. In biological image analysis, pre-trained models from the ImageNet competition [139] were successfully adapted to classify skin lesions [218], perform morphological profiling [219] and analyze *in situ* hybridization (ISH) images[220, 221]. In genomics, Kelley *et al.* [64] demonstrated the utility of transfer-learning for sequence-based predictive models of chromatin accessibility. They trained the multi-task Basset

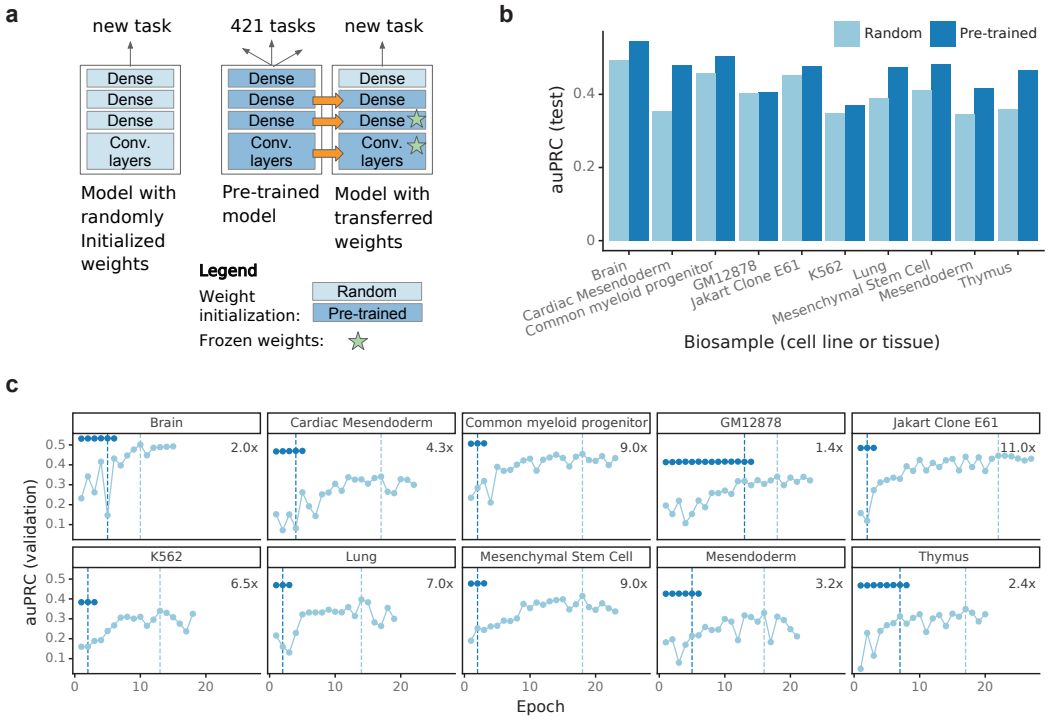


Figure 5.4: Adapting existing models to new tasks (transfer learning). **(a)** Architecture of alternative models for predicting chromatin accessibility from DNA sequence. Model parameters are either randomly initialized (left) or transferred from an existing neural network pre-trained on 421 other biosamples (cell lines or tissues, right). **(b)** Predictive performance measured using the area under the precision-recall curve, comparing randomly initialized (light blue) versus pre-trained (dark blue) models. Shown is the performance on held-out data (chromosomes 1, 8 and 21) for 10 biosamples that were not used during pre-training. **(c)** Training curves, showing the area under the precision-recall curve on the validation data (chromosome 9) as a function of the training epoch. The dashed vertical line denotes the training epoch at which the model training is completed. Pre-trained models require fewer training epochs than randomly initialized models and achieve more accurate predictions. Taken from [1].

model for predicting binary chromatin accessibility profiles of 149 cell types. Next, they trained single-task models of chromatin accessibility in 15 other cell types using weights from the multi-task model for initialization. The predictive performance was higher for models initialized with transferred weights compared to models initialized with random weights.

Here, we revisited the Basset transfer learning example on a larger dataset of chromatin accessibility profiles for 431 biosamples (cell types or tissues). We trained a genome-wide multi task model predicting chromatin accessibility for 421 biosamples

while holding out 10 biosamples. For the 10 held-out biosamples, we transferred the weights of the network to a new model and replaced the final fully connected layer with a randomly initialized one (Figure 5.4A). The transferred single-task model was trained once for each of the 10 held-out biosamples. The weights of all layers except the last two fully connected layers were kept fixed during re-training. The model initialized with transferred weights yielded improved predictive accuracy for all biosamples, with 15.1% auPRC improvement on average, compared to the model initialized with random weights (Figure 5.4B). In addition to better predictive performance, the transferred models trained much faster. On average, training the randomly initialized model to optimal performance took 17.3 epochs, resulting in more than one day of training for each of the 10 biosamples, while training the transferred model took 5.3 epochs on average resulting in 7 hours of training on average on the same Titan X GPU (Figure 5.4C). This reduction of training time can allow researchers to train more accurate models on smaller datasets while saving time and compute costs.

Transfer learning is facilitated by Kipoi in three ways. First, it provides direct access to a comprehensive collection of state-of-the-art models in genomics. Models can be easily browsed on the kipoi.org website by an informative name, tag such as 'DNA accessibility' or machine learning framework name. This allows users to find a model trained on the most similar task. Second, each model can be easily installed and already contains a tested data-loader which can be often directly used for model retraining. Third, the `kipoi predict` command offers an option to store the activation of a desired intermediate layer instead of the final prediction. These activations of an intermediate layer can be used as input features for a new model. Since they are already pre-computed, the training process can be substantially sped up without hurting predictive performance.

5.3.3 Predicting the molecular effects of genetic variants using interpretation plugins

Two important applications of trained models in genomics, with translational relevance in human genetics and cancer research, are predicting the effects of genetic variants on molecular phenotypes and dissecting the contribution of individual nucleotides in the scored sequence via feature contribution scores [62, 63, 96]. Variant effect prediction has been implemented individually by a subset of published sequence-based predictive models such as DeepBind [62], DeepSEA [63], and CpGenie [96]. Kipoi implements a plugin (kipoi-veff python package) that allows annotating variants obtained from the variant call format (VCF) files using any DNA sequence based model. Performing in-silico mutagenesis (ISM), the module computes transformations of model predictions for sequence containing i) the reference and ii) the alternative allele (Figure 5.5A). The module distinguishes and handles two classes of models. First, if the model can be applied across the entire genome, such as chromatin accessibility models, sequences centered on the query variants are generated (top row, Figure 5.5B). Second, if the model can only be applied to regions anchored at specific genomic locations, such as splicing models at intron-exons junctions, the valid regions are overlapped with the variants of interest (bottom row, Figure 5.5B). A uniform handling of these two scenarios (Figure 5.5C)

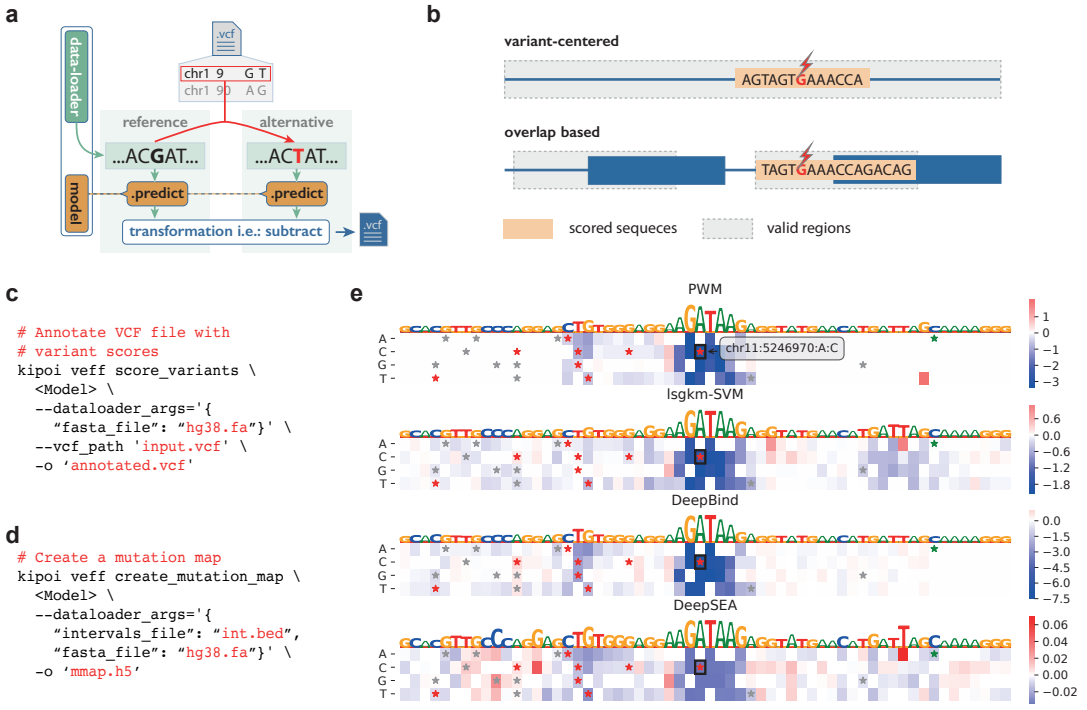


Figure 5.5: Variant effect prediction and feature contribution scores. **(a)** Schema of variant effect prediction using in-silico mutagenesis. Model predictions calculated for the reference allele and the alternative allele are contrasted and written into an annotated copy of the input variant call format file (VCF). **(b)** Kipoi uniformly supports variant effect prediction for models that can make predictions anywhere in the genome (top) and also for models that can make predictions only on predefined regions such as exon boundaries (bottom). **(c)** Generic command for variant effect prediction. **(d)** Generic command to compute the contribution scores using in-silico mutagenesis. **(e)** Feature contribution scores visualized as a mutation map (heatmap, blue negative effect, red positive effect) for variant rs35703285 and the predicted GATA2 binding difference between alleles for four different models. The black boxes in the mutation maps highlight the position and the alternative allele of the respective variant. Additionally, stars highlight variants annotated in the human variant database ClinVar with red: (likely) pathogenic, green: likely benign, grey: uncertain or conflicting significance, other. Taken from [1].

greatly simplifies their application. Altogether, the variant effect prediction module allows integrating a broad range of regulatory genomics predictive models into personal genome annotation pipelines and is trivially extended with newly added models.

Feature contribution scores highlight the parts of a given input that are most influential for the model prediction (Section 2.3.8). As shown in the previous section, feature contribution scores can be used to localize cis-regulatory elements such as transcription

factor binding sites. Kipoi implements a plugin (kipoi-interpret) which can compute ISM, saliency maps [222, 110], and DeepLIFT [110] for the majority of models in Kipoi (saliency maps and DeepLIFT are limited to models implemented in specific deep learning frameworks). It also provides visualization tools such as mutation maps [62] shown in Figure 2.6E.

To demonstrate the use of feature contribution scores for inspecting genomic regions containing the variant of interest, we computed the ISM contribution scores for the first four models from Figure 5.2 at the GATA1 binding site overlapping a pathogenic variant rs35703285 [223] linked to beta Thalassemia (MedGen:C0005283). ISM mutation maps showed that all four models agree on the effect and its direction. It also shows that a similar loss of GATA2 binding can be expected from other variants in the region overlapping the GATA2 motif. Interestingly, the three most complex models (lsgkm-SVM, DeepBind, and DeepSEA) predict effects of similar strength further away from the core motifs, reflecting that they can model more complex regulatory structure than the sole core motif captured by the position weight matrix approach. There is increasing evidence that the sequence context of binding sites plays a major role in gene regulation [13]. eQTL variants often localize near but not at transcription factor binding sites, advocating for using advanced regulatory models when interpreting personal genomes.

5.3.4 Predicting pathogenic splice variants by combining models

State of the art variant pathogenicity scores achieve higher accuracy by combining predictions from multiple models. By combining multiple models, these scores can cover multiple biological processes as well as average out conflicting predictions for the same biological process. Kipoi greatly facilitates the process of building new variant pathogenicity scores thanks to the standardization of models and the variant effect prediction plugin introduced in previous Section 5.3.3. To demonstrate this, we integrated four predictive models covering complementary aspects of splicing into a single variant pathogenicity score applicable to variants located near splice sites (Figure 5.6A).

Splicing is a multi-step process and genetic variants disrupting any of the splicing steps may cause genetic diseases. In fact, variants disrupting splicing are one of the most frequent cause of genetic disease [224]. There are three locations on the RNA important for splicing: the donor site, the acceptor site and the branchpoint. To evaluate the quality of all three sites, we integrated complementary models trained on different types of data focusing on differnet aspects such as i,ii) 5' and 3' MaxEntScan [89], a probabilistic model scoring donor and acceptor sites that was trained on genome annotation, iii) HAL [90] model scoring donor sites that was trained on a massively parallel reporter assay [90], and iv) Labranchor, a deep-learning model scoring the region upstream of the acceptor site for possible branchpoint locations that was trained from experimentally mapped branchpoints [91].

We trained and evaluated the composite model on two different datasets of splice variants classified either as pathogenic or benign (dbscSNV and ClinVar, Section A.4.5) and evaluated the performance of models using different subsets of the features. First, we incrementally added features corresponding to variant effect predictions of four splicing

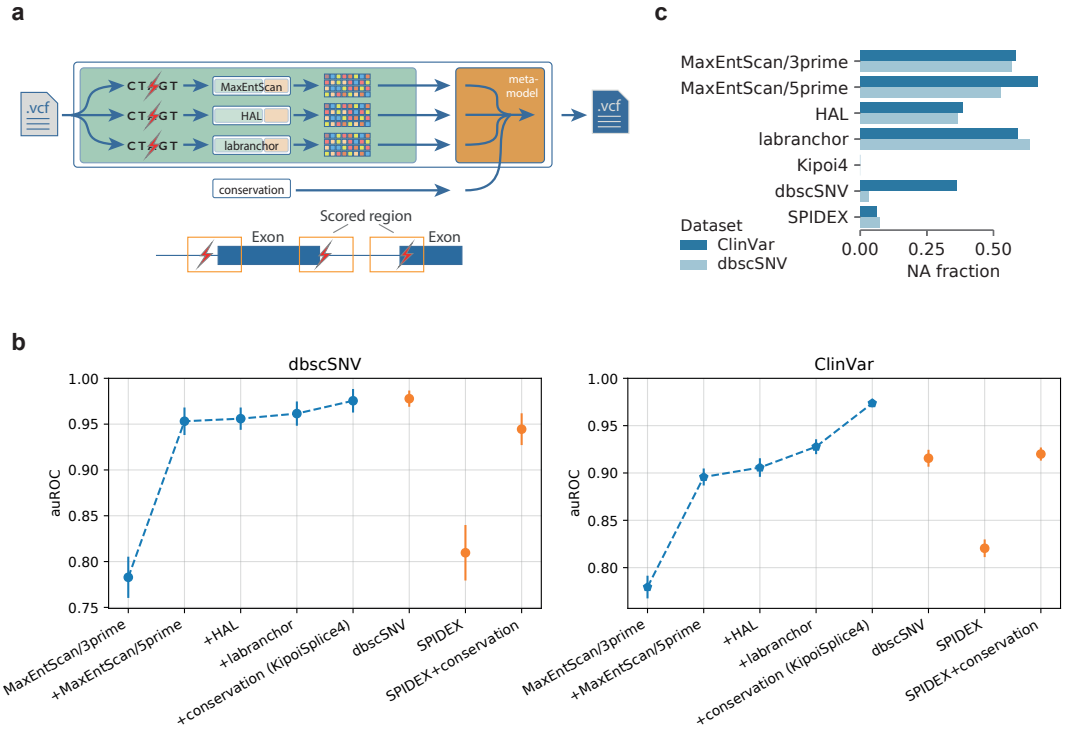


Figure 5.6: Composite models using Kipoi for improved pathogenic splice variant scoring. **(a)** Illustration of composite modelling for mRNA splicing. A model trained to distinguish pathogenic from benign splicing region variants is easily constructed by combining Kipoi models for complementary aspects of splicing regulation (MaxEntScan 3' models acceptor site, MaxEntScan 5' and HAL model donor sites, labranchor models the branchpoint) and phylogenetic conservation. These variant scores are combined by logistic regression to predict the variant pathogenicity (orange box). **(b)** Different versions of the ensemble model were trained and evaluated in 10-fold cross-validation for the dbscSNV and ClinVar datasets (Section A.4.5). The four leftmost models are incrementally added to the composite model in chronological order of their publication: the leftmost point only uses information from the MaxEntScan/3prime model, while '+conservation (KipoiSplice4)' uses all four models and phylogenetic conservation. These performances were compared to a logistic regression model using state-of-the-art splicing variant effect predictors (SPIDEX, SPIDEX+conservation, dbscSNV). KipoiSplice4 achieves state-of-the-art performance on the dbscSNV dataset and outperforms alternative models on ClinVar which contains a broader range of variants. **(c)** Fraction of unscored variants for different models in the dbscSNV and ClinVar datasets. Taken from [1].

models in Kipoi in the chronological order of model publication. We show that with an increasing number of splicing models in the ensemble, the performance increased for both, dbscSNV and ClinVar datasets (Figure 5.6B, four left-most methods). We refer to the logistic regression model using features of all four splicing models as KipoiSplice4. Next,

we evaluated the performance of two state-of-the-art splicing scores: the random forest score (dbscSNV) [225] and SPIDEX [226]. Since dbscSNV uses conservation scores while the Kipoi models do not, we trained dbscSNV without the conservation scores (dbscSNV) and added the same four conservation scores (Section A.4.5) to SPIDEX (SPIDEX w/ cons.) as well as to the ensemble of four Kipoi models (KipoiSplice4 w/ cons.). Among the methods not using conservation scores, dbscSNV and KipoiSplice4 show similar performance on the dbscSNV dataset. On the ClinVar dataset, dbscSNV w/o cons. could not be computed and the KipoiSplice4 showed the best performance. Comparing methods that used conservation scores, the performance of the Kipoi ensemble was similar to the dbsSNV model on the dbscSNV dataset, while it outperformed it on the ClinVar dataset. One reason for better performance of 'KipoiSplice4' is that it scores more variants (Figure 5.6C). SPIDEX and dbscSNV models do not explicitly model the splicing branchpoint, while KipoiSplice4 does so using Labranchor.

Finally, we wrapped the individual models (MaxEntScan, HAL and Labranchor) into a data-loader and made the ensemble model KipoiSplice4 (with and without the conservation scores) available in Kipoi. This allows users to predict effects of *de novo* variants in splice sites. Building on top of this work, we have further improved the individual modules focusing on differnet aspects of splicing and developed a new model termed MMSplice [4, 7] which won the CAGI5 exon skipping prediction challenge. MMSplice is also available in Kipoi.

5.4 Discussion

We have developed Kipoi, a platform to exchange predictive models in genomics consisting of three key components: definition of the trained model, central repository containing over 2,000 trained models, and a unified API to load and use the trained models. Through rigorous testing of models and dependency installation, Kipoi overcomes the barriers of model deployment, improves model dissemination, and ultimately facilitates reproducible research. We have demonstrated that Kipoi not only provides the parameters of trained models useful for transfer learning as done by traditional model repositories, but also greatly facilitates making predictions (and hence model comparison) on new data through a unified prediction API. Additionally, it allows sequence-based models to be directly used for scoring genetic variants stored in a VCF file. Finally, it provides a plugin to interpret these models using various feature contribution scores.

The Kipoi model repository is dedicated to trained models with applications in genomics. Specifically, we request at least one input data modality to be derived either from DNA sequence (which includes amino acid sequences), or from a functional genomics assay such as ChIP-seq or protein mass-spectrometry. With the increased adoption, we plan to include also other types of models. Additionally, because the Kipoi API is agnostic to input or output data types as well as machine learning frameworks, and because creating a new Kipoi model repository (i.e. git repository) is simple, it could easily be adapted to other domains beyond genomics.

Another popular strategy for implicitly sharing models is to share the pre-computed model predictions for a large set of inputs or to expose the model through a web-server. For example, variant scoring models such as CADD provide a large dataset of predictions for all possible single-nucleotide variants in the human genome as well as a web-tool to annotate variants. However, such approach has important limitations. The dataset of precomputed predictions only covers a small fraction of all possible genetic variants. Due to the exponential growth of possible sequence variants such as indels, storing predictions for all possible variants (say indels of length < 20 bp) is impossible with today's compute resources. While web servers can potentially score any variant in the genome including indels, they are impractical when many predictions have to be made or when the data are under privacy protection. Hence, sharing predictive models is necessary in genomics.

Standardization of predictive models (and software in general) has numerous benefits. By standardizing components, a consistent API can be developed and used for many downstream applications. APIs foster code modularity, remove code redundancy, and allow developers to focus on more important tasks than re-implementing the same functionality over and over again. For example, the variant effect prediction functionality is only possible in Kipoi due to the standardization of models. By adhering to the model standard, the contributors can directly use the variant effect prediction plugin with their model. We acknowledge that strict standards can introduce some overhead initially and restrict the number of use-cases. Hence, there should be a compromise between strict standards and no standards. The Bioconductor package platform and the GEO data archive platform are nice examples of striking the right balance between the two. These platforms successfully compromise between strict standards and no standards. With Kipoi, we tried to balance the two by being strict on some aspects such as testing and transparent input-output specification, while being loose on other aspects such as the choice of the machine learning framework. Altogether, we are convinced that a short-term investment in the standardization of models as done in Kipoi will bring large long-term benefits.

6 Conclusions

Thanks to the advances in next generation sequencing technologies, many genome-wide assays measuring molecular phenotypes (i.e. functional properties of the genome) became available in the recent years. This opened a path to understand the human genome by relating DNA sequence to molecular phenotypes using predictive models. In parallel, powerful modeling techniques such as deep neural networks began to emerge. This thesis leveraged the advances in deep neural networks and further developed them to make them more accurate for genomics problems (BPNet and spline transformation) and improved their interpretability (DeepLIFT for sequence-to-profile models, CWM scanning, motif interactions). Additionally, it established a platform to share predictive models developed by the community (`http://kipoi.org`).

We have developed a new neural network layer–spline transformation–to flexibly and robustly model distances to various genomics landmarks (description and results available in [2]). Extending the sequence-based model predicting *in vivo* RNA-binding protein binding sites with spline transformations to model distances to various genomic landmarks increased the prediction accuracy for 120 out of 123 proteins. We demonstrated that spline transformation yields better predictive performance, trains faster and is more robust than the piecewise linear transformation, as obtained by a composition of rectified linear units.

Additionally, we have created a versatile tool for regulatory code discovery by combining an accurate sequence-to-profile model–BPNet with advanced model interpretation techniques also further developed in this thesis. BPNet consists of three key elements: model architecture maintaining the single-nucleotide resolution, loss function separately modeling the profile and total counts, and a flexible way to control for assay specific biases. We showed that BPNet can predict ChIP-nexus data at base-pair resolution with accuracy comparable to experimental replicates. We also showed that by learning to predict the profile, the model learns better representations of TF binding compared to the binary classification models. The binary classification models are trained to predict the presence or absence of read coverage peaks and thereby discard valuable information about TF binding present in read coverage profiles. Better representations obtained by predicting the profile resulted in more accurate binary classification predictions as well as better enhancer activity predictions.

Despite the complex model with many parameters, we were able to extract motifs from the model, map them back to the genome much more accurately than a PWM, and study the cooperativity between motifs. Regulatory grammar of TF binding is currently poorly understood due to the lack of computational methods. Simple models like PWMs can not capture the complexity of TF binding while the parameters of complex models such deep neural networks are difficult to interpret. We applied BPNet to ChIP-nexus

data of the four well-studied pluripotency TFs Oct4, Sox2, Nanog and Klf4 in mouse embryonic stem cells (ESCs). By interpreting BPNet, we discovered motifs both directly and indirectly bound by TFs, clarified previously conflicting motifs for Nanog, and discovered many transposable elements bound by TFs. By mapping motifs back to the genome and studying their interactions, we observed a strong preference for Nanog to bind in a 10 bp periodic pattern corresponding to the helical periodicity of the DNA. Finally, we extracted rules of cooperative binding between the four TFs, which uncovered significant soft preferences of specific motif pairs to interact over short-range and long-range distances. These interactions are consistent with previously studied protein-protein interactions [182, 189] or roles of pioneer transcription factors [227, 155], thereby linking preferential motif spacing in the genome with known biological mechanisms. While ChIP-nexus was the primary assay for which BPNet and the interpretation tools were initially developed, we show that they can be directly applied to other assays such as ChIP-seq. In summary, we have created a highly versatile computational method for the discovery of cis-regulatory code from TF binding data, opening the path for the systematic discovery of cis-regulatory code in experimentally accessible cell types.

Finally, we have developed Kipoi, a platform to exchange predictive models in genomics consisting of three key components: definition of the trained model, central repository containing over 2,000 trained models and a unified API to load and use the trained models. We have demonstrated that Kipoi not only provides the parameters of trained models to perform transfer learning as done by traditional model zoos, but also greatly facilitates making predictions (and hence model comparison) on new data through a unified prediction API, it allows sequence-based models to be directly used for scoring genetic variants stored in a VCF file, and provides a plugin to interpret model predictions using various feature contribution scores. Through rigorous testing of models and dependency installation, Kipoi overcomes the barriers of difficult model deployment, improves model dissemination, and ultimately facilitates reproducible research.

6.1 Outlook

Thanks to both, experimental and computational advances, the genomics field is entering an exciting era of new discoveries. Here I will pinpoint some directions of how the work presented in this thesis could be extended.

6.1.1 Extending spline transformation with attention mechanism

Spline transformation can not only used to model distances to external genomic landmarks, but also to the position within the sequence. For example, the `SplineWeight1D` layer in the concise package uses spline transformation to up or down weight positions in the 1D activation map. This layer was successfully used in the binary classification version of the fully-convolutional BPNet architecture. The disadvantage of such approach is that the positional weights are the same for all inputs. To make them dynamic and allow them to change their 'attention' to differnet parts depending on the input, the attention mechanism [228] could be used with spline transformation. By using splines

as the basis functions, the required memory of the attention mechanism, which grows quadratically with sequence length, could be kept small enough and hence be practical for genomics applications which require to process long sequences.

6.1.2 Extending the BPNet framework to new data modalities

We have shown that BPNet and the interpretation tools are also applicable to ChIP-seq. BPNet should work with any sequencing-based protocol where the local enrichment of reads and their spatial distribution (profile) reflects the measured quantity. Hence, BPNet could be used with assays such as CUT&RUN, ATAC, scATAC, DNase, CAGE, eClip and iClip. There are two key challenges when doing so. First, a reliable track of assay specific biases should be obtained for the assay of interest. For some assays such as DNase, ATAC or scATAC, the choice for the right bias track is still a subject of debate. Second, defining the profile contribution score based on 'spikiness' of the profile might not work for some assays. Hence alternative ways of aggregating the contribution scores across multiple positions should be explored. We note that regardless of the weighting scheme, the trick exploiting the linearity of DeepLIFT could still be used to compute the contribution scores using a single backpropagation pass. Altogether, by applying BPNet to these other assays, we could build more accurate models sequence-based models for the measured molecular phenotypes beyond TF binding and derive novel biological insights.

6.1.3 Understanding the binding grammar of all TFs with BPNet

BPNet coupled with the interpretation techniques could be applied to many more publicly available datasets. For example, consortia such as ENCODE or Roadmap Epigenomics provide thousands of genome-wide datasets probing TF binding. These could be used to systematically discover the cis-regulatory code and uncover the hierarchy as well as spatial constraints of TF binding at enhancers. We are convinced that such 'tour-de-force' approach is the only way to decipher TF binding and transcriptional activation due to the biological complexity. In addition to applying BPNet to more datasets, effective strategies to deal with multi-tasking are needed. It is unclear, how to best apply BPNet to 100 or 1000 experimental tracks simultaneously or whether such extreme multi-tasking is useful at all. Additionally, we designed the BPNet architecture to be simple to specify and to have only few hyper-parameters. We think that there is still plenty of room for improvement in the architecture especially thanks to the incredible progress of the deep learning community. Given enough computational resources, the best architecture could be even automatically derived using the neural architecture search (NAS) [229, 230].

6.1.4 Extending the Kipoi platform and leveraging it for key genomics applications

Kipoi could be extended to include models based on other data-modalities such as single-cell RNA expression or protein expression. The infrastructure is general enough to support the needs of these models as long as re-training of models is not required. We have primarily focused on sequence-based models since sequence can be almost perfectly

measured and does not change much with new sequencing technologies. By contrast, models which take as input the experimental measurements from some high-throughput assay such as single-cell expression are rapidly changing due to the rapid improvements of these assays. Hence, models taking as input the output of these assays will be only temporarily useful to the community as the assay might not be around in five years of time. Nevertheless, as these technologies mature and better techniques for normalizing the data across multiple experimental platforms become available, we foresee Kipoi as the central medium to exchange these models.

One immediate application of the variant scoring functionality in Kipoi for many models would be to link the available 5,000 variant effect scores to higher phenotypes such as disease. This can be done in the context of differnet applications including GWAS causal variant fine-mapping, computing gene burden scores for association studies in rare Mendelian diseases, and improving the polygenic risk scores for common diseases. Additionally, variant pathogenicity scores like CADD could be extended to use the features available in Kipoi. After establishing some of these applications, Kipoi models could be integrated into the standard pipelines for variant interpretation in the clinics to give additional insights especially for the non-coding variants.

Finally, the Kipoi core team could partner with organizers of machine learning challenges in genomics such as DREAM[1] or CAGI[2] to coordinate the development of accurate predictive models in the genomics community. That way, the development of models would be modularized into three steps: i) designing training datasets and evaluation metrics done by challenge organizers, ii) training the best model on a sound datasets and metric done by challenge competitors and, iii) making the models available for the whole community done by the Kipoi core team. Such division of tasks would allow each group to focus on a single task and leverage their expertise. Such modularization would also lower the entry barrier for new students or machine learning experts lacking domain expertise. Additionally, the joint effort would be effectively utilized since the end product—trained predictive model—would be easy to use for everyone and could hence best serve its purpose.

[1] `http://dreamchallenges.org/`
[2] `https://genomeinterpretation.org/`

A Appendix

A.1 ChIP-nexus data and pre-processing

This section corresponds to chapters 3 and 4. The following three sub-sections A.1.1, A.1.2, and A.1.3 were written by Sabrina Krueger at the Stowers Institute for Medical Research. They are included in this thesis for the sake clarity and completeness. For detailed author contributions see pages 37 and 53.

A.1.1 Cell culture

Mouse R1 ESCs were cultured on 0.1% gelatin-coated plates without feeder cells. Mouse ESC medium was prepared by supplementing N2B27 medium (1:1 mix of DMEM/F12 with GlutaMax supplemented with N2 and Neurobasal medium supplemented with B27, Invitrogen) with 2 mM L-Glutamine (Stemcell Technologies), 1x 2-Mercaptoethanol (Millipore), 1x NEAA (Stemcell Technologies), 3 μM CHIR99021 (Stemcell Technologies), 1 μM PD0325901 (Stemcell Technologies), 0.033% BSA solution (Invitrogen) and 1e7 U/ml LIF (Millipore).

A.1.2 ChIP-nexus

For each ChIP experiment, 107 mouse ESCs were used. Cells were washed with PBS and cross-linked with 1% formaldehyde (Fisher Scientific) in PBS for 10 min at room temperature. The reaction was quenched with 125 mM glycine. Fixed cells were washed with cold PBS, scraped, centrifuged, resuspended in cold lysis buffer (15 mM HEPES (pH 7.5), 140 mM NaCl, 1 mM EDTA, 0.5 mM EGTA, 1% Triton X-100, 0.5% N-lauroylsarcosine, 0.1% sodium deoxycholate, 0.1% SDS), incubated for 10 min on ice and sonicated with a Bioruptor Pico for four cycles of 30 s on and 30 s off. ChIP-nexus procedure and data processing were performed as previously described (He et al., 2015) except that the ChIP-nexus adaptor mix contained four fixed barcodes (ACTG, CTGA, GACT, TGAC). For each ChIP, 5 μg antibody was coupled to 50 μl of Dynabeads Protein A or Protein G (Invitrogen). The following antibodies were used: α-Oct3/4 (Santa Cruz, sc-8628), α-Sox2 (Santa Cruz, sc-17320), α-Nanog (Santa Cruz, sc-30328), α-Klf4 (R&D Systems, AF3158), α-Klf4 (Abcam, ab106629), α-Esrrb (Abcam, ab19331), α-Pbx 1/2/3 (Santa Cruz, sc-888), and α-Zic3 (Abcam, ab222124). At least two biological replicates were performed for each factor. Single-end sequencing of 75 bp was performed using an Illumina NextSeq 500 instrument according to manufacturer's instructions.

A.1.3 PAtCh-Cap

For each PAtCh-cap experiment, 10% of sheared chromatin sample volume from 107 mouse ESCs was used as input. Chromatin was prepared as described for ChIP-nexus. PAtCh-Cap was performed as previously described (Terooatea et al., 2016). Processing pipeline

A.1.4 Processing pipeline

Random barcodes and fixed barcodes were trimmed off the reads and reassigned to FASTQ labels using nimnexus (v0.1.1). The adapters were then trimmed using cutadapt (v1.8.1). Next, the reads were aligned with BWA (v0.7.13) using the command `bwa aln -q 5 -l 32 -k` to the mouse genome assembly mm10. Mapping stats were computed using SAMtools flagstat (v1.2). Reads were filtered using SAMtools view to remove unmapped reads and mates, non-primary alignments, reads failing platform or vendor quality checks, and PCR or optical duplicates (`-F 1804`). Low quality reads (MAPQ < 30) were also removed. Reads aligned to the same position with the same barcode, CIGAR string and the SAM flag were de-duplicated using `nimnexus dedup` (v0.1.1). The total number of final (filtered) aligned reads was 243M for Oct4, 140M for Sox2, 214M for Nanog and 176M for Klf4. The final filtered BAM file was converted to tagAlign format (BED 3+3) using bedtools 'bamtobed' (v2.26). Cross-correlation scores were obtained for each file using phantompeakqualtools (v1.2). BigWig tracks containing the number of aligned 5' read ends mapped on the corresponding strand (pooled across all replicates) were generated using `bedtools genomecov -5 -bg -strand +/-`, followed by bedGraph to BigWig conversion using UCSC bedGraphToBigWig.

Peaks were called using MACS2 (v2.1.1.20160309) by shifting and extending the reads to obtain coverage tracks similar to ChIP-seq (shift=-75, extsize=150). Peaks overlaping the blacklisted regions listed in `http://mitra.stanford.edu/kundaje/akundaje/release/blacklists/mm10-mouse/mm10.blacklist.bed.gz` were excluded. Peak calling was performed on different combinations of aligned reads from different replicate experiments as described in X and as used in the ENCODE ChIP-seq pipeline. Finally, 1kb regions centered at peak summits from the "optimal set" of IDR peaks passing the FDR threshold of 0.05 were used as the peak regions for the corresponding TF. We observed 25,849 peaks for Oct4, 10,999 for Sox2, 56,459 for Nanog and 57,601 for Klf4.

A.1.4.1 Code availability

Nim-nexus code is available at `https://github.com/Avsecz/nimnexus/`. ChIP-nexus pipeline performing the described steps (i.e. turning the raw reads in the FASTQ format to BigWig coverage tracks and the called peaks) is available at `https://github.com/kundajelab/chip-nexus-pipeline`. Detailed pipeline specification is available at `https://docs.google.com/document/d/1h9lZ0GyVWd02RCmtaFWSaSFzrcNHoH_OgyPHMpU7b04`.

A.2 Motif validation and analysis for Section 4.3.2.1

This section corresponds to chapter 4, section 4.3.2.1. It was written by Julia Zeitlinger at the Stowers Institute for Medical Research. Analysis was performed by Melanie Weilert. The section is included in this thesis for the sake clarity and completeness. For detailed author contributions see page 53.

First, we analyzed Nanog binding in an attempt to address and resolve previous conflicting reports. In addition to the main, most frequent Nanog motif, we identified two additional motifs, Nanog-mix and Nanog-alt, with sharp and specific Nanog ChIP-nexus footprints (Figure A.1A). It has been reported that Nanog is able to form homodimers *in vitro* and *in vivo* [231], but the underlying sequences that are bound by Nanog homodimers are not known. There is also evidence that Nanog forms a heterodimer with Sox2 [182, 189], but whether these putative Nanog-Sox2 heterodimer motifs are bound *in vivo* is not clear.

We therefore analyzed whether the three Nanog motifs may represent different modes of binding. All three motifs contain a TCA core where Nanog binding is the strongest (main footprint in Figure A.1A). Consistent with direct binding, a sequence containing this core (TGATGGC), has been shown to be bound by Nanog in an EMSA gel shift assay [183]. Next to the main binding footprint on the left, these three motifs show additional binding footprints with subtle differences from each other (asterisked in Figure A.1A). Strikingly, the Nanog-mix and Nanog-alt motifs show in their CWM a sequence that corresponds exactly to the sequence to which a monomeric Nanog is bound in a crystal structure (AATGGGC) [232]. Our Nanog binding footprint therefore likely represents a footprint where Nanog makes two direct DNA contacts, consistent with Nanog binding as homodimers.

The Nanog-alt motif contains an additional GG to the left that is predicted to strongly contribute towards Nanog binding. This raises the possibility that Nanog binds to the third motif with a partner. However, Sox2 was not bound to this motif (Figure 4.5). In fact, we did not identify any motif to which Nanog was likely bound as heterodimer with Sox2. There was no motif with sharp footprints of both Nanog and Sox2, and the previously identified putative Nanog-Sox2 motif was not bound in our ChIP-nexus data (Figure 4.5). These results argue against Nanog and Sox2 binding together as a stable heterodimer in ESCs.

To confirm that the TF partner motifs we discovered were indeed bound by the expected TF, we performed more ChIP-nexus experiments and observed a sharp footprint of Zic3 on the discovered motif (Figure A.1B). Likewise, we found that the Esrrb motif showed a footprint in Esrrb ChIP-nexus data. These results confirm the identity of the Zic3 and Esrrb motifs.

Finally, we analyzed the TFIIIC B-box motif in more detail since we found it highly unusual to identify a Pol III motif. We found that the motif was found inside a set of ∼280 tRNA genes which are transcribed by Pol III. These overlapping sites were highly conserved across vertebrates with diverse amino acid anti-codons (Figure A.1C). Since the TFIIIC B-box motif is palindromic, we then computationally oriented the

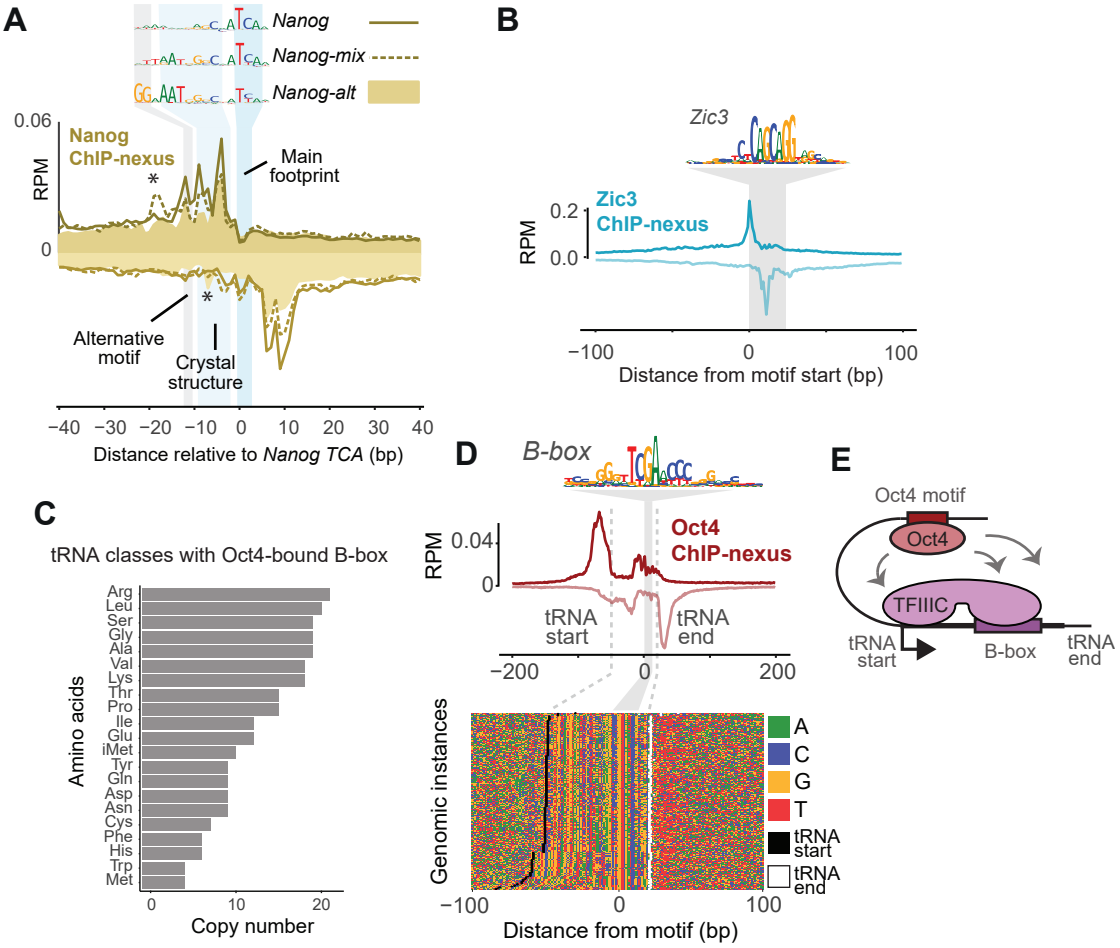


Figure A.1: Validation of motifs discovered by TF-MoDISco. A) Three varying Nanog motifs were identified with sharp Nanog binding. Upon alignment to the collective TCA main motif component, which shows the strongest ChIP-nexus footprint, each Nanog motif displays subtle and distinct Nanog binding differences. The Nanog-mix and Nanog-alt motif sequences contain contributing nucleotides in their CWM that match the sequence AATGGGC, which is bound by Nanog in a crystal structure. The CWM of Nanog-alt motif has GG on the left and is considered an alternative motif whose binding specificity is unclear. **B)** A strong ChIP-nexus footprint of Zic3 is found across the TF-MoDISco Zic3 motif, confirming the identity of the Zic3 motif. **C)** The TFIIIC B-box motif was found to overlap with 283 tRNA genes with a diverse set of amino acid classifications. **D)** tRNA-overlapping TFIIIC B-box motif instances were reoriented to match tRNA gene transcriptional direction and ordered by tRNA gene start proximity. This reveals Oct4 binding across both the TFIIIC B-box and tRNA gene start/stop sites. **E)** Model representing the relationship between the TFIIIC B-box motif to tRNA genes and Oct4.

motifs based on the transcription direction. This resulted in a much clearer ChIP-nexus binding footprint of Oct4, revealing that Oct4 binds upstream and downstream of tRNAs in addition to the enclosed TFIIIC B-box (Figure A.1D,E). Interestingly, Oct1, a TF containing a similar binding motif to Oct4, has previously been shown to regulate Pol III-transcribed genes in HeLa cells [233]. Furthermore, modulation of Pol III transcription is critical for maintaining the pluripotency of ESCs [234]. Taken together, our results raise the possibility that the Oct4 binding we observe at the TFIIIC B-box motif is functionally important to regulate tRNA production in ESCs.

A.3 Relationship between the Poisson log-likelihood, mean-squared error and multinomial log likelihood

Let's start first by writing down the negative log-likelihood for the Multinomial distribution. Let L be the sequence length, N the total number of events (i.e. total number of read counts in the region) and p_i the probability of obtaining the outcome i (e.g. the read gets aligned to position i). Then, the negative log likelihood can be written as

$$\begin{aligned} NLL_{Mult}(k_1, \ldots, k_L | N, \mathbf{p}) &= -\log \frac{N!}{k_1! \ldots k_L!} \prod_{i=1}^{L} p_i^{k_i} \\ &= -\sum_{i=1}^{L} \log k_i \log p_i + M \ . \end{aligned}$$

Note that gathered all the terms independent of $p_i \ \forall i$ to constant M. Let's assume the read counts at each genomic location k_i are distributed according to the Poisson distribution. The Poisson log likelihood for the sequence region of length L reads

$$\begin{aligned} \sum_{i=1}^{L} NLL_{Poiss}(k_i, \boldsymbol{\mu}) &= -\sum_{i=1}^{L} logP_{Poiss}(k_i | \mu_i) \\ &= -\sum_{i=1}^{L} \log e^{-\mu_i} \frac{\mu_i^{k_i}}{k_i!} \\ &= \sum_{i=1}^{L} (\mu_i - k_i \log \mu_i) + P \ . \end{aligned}$$

If we replace μ_i with $N_p p_i$, where N_p is the predicted number of total counts and use $\sum_{i=1}^{L} p_i = 1$, $\sum_{i=1}^{L} k_i = N$, we obtain:

$$\begin{aligned}\sum_{i=1}^{L} NLL_{Poiss}(k_i, \boldsymbol{\mu}) &= \sum_{i=1}^{L}(N_p p_i - k_i \log N_p - k_i \log p_i) + P \\ &= N_p \sum_{i=1}^{L} p_i - \log N_p \sum_{i=1}^{L} k_i - \sum_{i=1}^{L} k_i \log p_i) + P_2 \\ &= N_p - N \log N_p - \sum_{i=1}^{L} k_i \log p_i + P_2 \ ,\end{aligned}$$

We observe that the second term equals to the multinomial negative log-likelihood. If we set $N_p = e^{\log N_p}$, $N = e^{\log N}$ and perform a Taylor expansion[1] up to the squared term for variable $\log N_p$ around $\log N$ using, we obtain:

$$\begin{aligned}N_p - N \log N_p &= e^{\log N_p} - e^{\log N} \log N_p \\ &\approx N(1 - \log N) + \frac{N}{2}(\log N_p - \log N)^2 \ .\end{aligned}$$

This means that we can approximate the Poisson log-likelihood by a sum of mean-squared errors and the multinomial loss function where the predicted log of total counts $\log N_p$ is close to the true total counts $\log N$.

$$NLL_{Poiss}(\mathbf{k}|N_p, \mathbf{p}) \approx NLL_{Mult}(\mathbf{k}|N, \mathbf{p}) + \frac{N}{2} MSE(\log N, \log N_p) \ . \tag{A.1}$$

We simplify the expression further by replacing the N in front of MSE with $\alpha \bar{N}$, where $\bar{N}$ is the average (or median) value of N across the dataset and α is the tuning parameter which allows to up or down-weight the importance of total count prediction:

$$NLL_{Poiss}(\mathbf{k}|N_p, \mathbf{p}) \approx NLL_{Mult}(\mathbf{k}|N, \mathbf{p}) + \alpha \frac{\bar{N}}{2} MSE(\log N, \log N_p) \ . \tag{A.2}$$

If $\alpha = 1$, the multinomial loss and the mean squared error loss are balanced according to the Poisson log-likelihood.

A.3.1 Transposable element analysis

Transposable element analysis RepeatMasker annotations for mm10 obtained from `http://www.repeatmasker.org/genomes/mm10/RepeatMasker-rm405-db20140131/mm10.fa.out.gz` were used to compute the overlap of seqlets with transposable elements (TEs). A seqlet was considered to overlap a TE if it was fully contained with at least one element

[1] $$f(x) = f(a) + f'(a)(x-a) + \frac{1}{2} f''(x)(x-a)^2 + O((x-a)^3)$$

defined in Repeat Masker annotation. Kimura 2-parameters distance (Kimura, 1980) between the seqlet sequence and the consensus sequence of the motif was used to sort the seqlets in Figure 4. This distance metric was re-implemented in Python and is equivalent to dist.dna function from R's APE package with the `model='K80'` parameter (`https://www.rdocumentation.org/packages/ape/versions/5.2/topics/dist.dna`).

A.3.2 Pairwise motif interaction analysis

We studied the pairwise interaction between the following motifs discovered by TF-MoDISco:

- Oct4-Sox2 (pattern 0 from Oct4, consensus=TTTGCATAACAA),
- Sox2 (pattern 1 from Sox2, consensus=GAACAATGG),
- Nanog (pattern 1 from Nanog, consensus=AGCCATCA),
- Klf4 (pattern 0 from Klf4, consensus=CCACGCCC).

We considered motif instance pairs (A, B) spaced at some distance d $<$ 160 bp and compared BPNet ChIP-nexus profile predictions between 4 cases: where either motif A or B was replaced by a random sequence, where both were replaced by a random sequence or where both were left intact. Motif instance pairs were either simulated in synthetic sequences or were detected by CWM scanning in ChIP-nexus peaks.

A.3.2.1 Synthetic sequences

For synthetic sequences, we first created 128 random sequences of 1 kb in length by sampling the base at each position with equal probability. Next, we replaced the central bases by the consensus sequence of motif A and similarly inserted motif B d-bases downstream of A (d is the distance between motif centers). We used BPNet to predict the stranded ChIP-nexus profile for the primary TF of motif A (e.g. Oct4 for the Oct4-Sox2 motif and Nanog for the Nanog motif). We averaged the predictions across the 128 random sequences to obtain the marginal predicted profile P_{AB}. We repeated the same procedure by i) inserting only the motif A in the center (P_A), ii) inserting only the motif B d-bases downstream of the center, and iii) not inserting any motif and hence only averaging the predictions across random sequences ($P_\emptyset$). We used the predicted profile P_A to determine the predicted summit location within +- 35 bp of the motif A center for each strand. The stranded summit location at motif A was then used to determine the profile height in all 4 scenarios averaged across the two strands. We denote average predicted profile height of the 4 different predicted profiles (P_A, P_B, P_{AB} and $P_\emptyset$) by h_A, h_B, h_{AB} and $h_\emptyset$ correspondingly. We define the directional score of motif B to motif A as: $(h_{AB} - (h_B - h_\emptyset))/h_A$.

Directional score of 1 denotes that profile height of A is the same whether or not motif B is present in the vicinity of A. If the directional score is higher than one, then the profile of A is higher compared to the case where B is absent. We note that the second

term in the numerator (h_B - $h_\emptyset$) corrects for the tail of the motif B profile which occurs when motif A and B are close to each other.

We performed the analysis for all motif pairs, strand orientations and possible pairwise distances ranging from 11 bp to 160 bp.

A.3.2.2 Genomic sequences

To compute the directional effect of motifs in genomic sequences, we first obtained motifs instance locations in 1kb ChIP-nexus peak regions using CWM scanning. We discarded motif instances from duplicated peak regions overlapping other peak regions by more than 200 bp as well as motif instances overlapping TEs (discovered by TF-MoDISco and mapped back to the genome using CWM scanning). Also, Sox2 motif instances overlapping the Oct4-Sox2 motif were discarded. For each motif pair, 4 model predictions were made:

- P_{AB}: the reference sequence of the whole interval in which the motifs were present
- P_A: motif instance B replaced by random sequence
- P_B: motif instance A replaced by random sequence
- $P_\emptyset$: motif instances A and B replaced with random sequence

We computed the profile heights at motif A profile summit locations in the same manner as for the synthetic sequences yielding 4 profile heights: h_A, h_B, h_{AB} and $h_\emptyset$. We added "pseudo counts" defined as the 20th percentile of the considered quantity to the tail-corrected profile height of the reference sequence: $h_{AB} - (h_B - h_\emptyset) + PC_{AB}$ as well as the profile height of the A-only sequence: $h_A + PC_A$. Next, we kept only the motif pairs where the tail-corrected profile height of the motif was in the top 20% for both motifs. This ensured that only motif pairs showing a footprint were used. Finally the directionality score was computed for each motif instance pair as:

$$(h_{AB} - (h_B - h_\emptyset) + PC_{AB})/(h_A + PC_A);. \tag{A.3}$$

We note that there are three main differences between the synthetic and genomic sequences. First, in genomic sequences, the background sequences were not random and may contain other motifs. Second, the "perfect" consensus sequence was used for injecting motifs in synthetic sequences, whereas for genomic sequences the motif sequence rarely matched the consensus. Third, the distribution of motif pairwise distances in genomic sequences is not perfectly uniform as for the synthetic case, hence some pairwise distances might be under-represented.

A.3.3 Motif pairwise spacing analysis

We obtained and filtered motif instances as described in previous section using CWM scanning. We discarded Sox2 sites overlapping the Oct4-Sox2 motif. To compute

whether motif A is located close to B more frequently than expected by chance, we counted i) the number of times a motif instance A is close to motif instance B and ii) the number of times motif instance A is close to motif instance B if we shuffle all motif instances between peaks while maintaining the relative location within the peak. Next, we constructed the following 2-by-2 contingency matrix:

```
cm = [[ # A not close to B (shuffled), # A not close to B],
      [ # A close to B (shuffled),       # A close to B]]
```

We applied the Pearson's Chi-square test (`chi2_contingency` from `scipy.stats`) to obtain the p-value quantifying whether the odds-ratios (A close vs not close to B) between the observed and shuffled motif instances are significantly different. Finally, we used the odds-ratio as a summary statistic to visualize whether A is closer to B than expected by chance:

$$\frac{\#\text{ A close to B}}{\#\text{ A not close to B}} \Big/ \frac{\#\text{ A close to B (shuffled)}}{\#\text{ A not close to B (shuffled)}} \,. \tag{A.4}$$

A.4 Additional method description for the section describing Kipoi

Methods described in this section are part of the Kipoi manuscript [1]. They are included in this thesis for the sake clarity and completeness. For author contributions see page vii.

A.4.1 Models

A.4.1.1 pwm_HOCOMOCO

Position frequency matrices (PFM) for all 600 human transcription factors in HOCOMOCO v10 were downloaded from `http://hocomoco10.autosome.ru/final_bundle/HUMAN/mono/HOCOMOCOv10_pcms_HUMAN_mono.txt` and transformed to position weight matrices (PWM) using the pseudocount probability of 0.001. Scanning the DNA sequences using the PWM matrix is implemented as a Keras model consisting of a single convolutional layer with one filter whose weights are set to the PWM, followed by global max pooling. The model operates on one-hot-encoded DNA sequence.

A.4.1.2 DeepBind

Original weights and architecture were obtained from supplementary material of the original publication6 and were converted to Keras 2.0 models (code: `https://github.com/kundajelab/DeepBindToKeras`).

A.4.1.3 DeepSEA

The DeepSEA model was converted from the original Torch7 model [63] (v0.94b) to a PyTorch model using a modified version of the script `https://github.com/clcarwin/convert_torch_to_pytorch`. Since prediction of model tasks and variant effect prediction use different handling of reverse-complement sequences there are two models in the Kipoi model zoo dedicated to the two different use cases in order to replicate results from the original model exactly. Implementations of reverse-complement handling were taken from .lua files provided in the software package in the publication [63]. Predictions of the models and variant effects produced by the models in the Kipoi repository match the predictions produced by the website `http://deepsea.princeton.edu/job/analysis/create/`.

A.4.1.4 FactorNet

FactorNet models were obtained from `https://github.com/uci-cbcl/FactorNet/tree/bef6f6b38e81d362162a106dc8a726ecae910138`. In addition to the models available in the github repository, Daniel Quang kindly provided the trained models for CEBPB and MAFK, which were part of the internal evaluation round in the ENCODE-DREAM in vivo transcription factor binding prediction challenge (`http://synapse.org/encode`). The models were converted to Keras 2.0 supporting the tensorflow backend.

A.4.1.5 MaxEntScan

We used MaxEntScan implemented in the maxentpy package (`https://github.com/kepbod/maxentpy`, v0.0.1) provided through the Bioconda channel. We implemented a data-loader that takes the reference genome FASTA file and the genome annotation GTF file as input and returns sequences of all regions [-3nt,5nt] w.r.t. the annotated 5' splice sites for the 5' model and sequences of all regions [-3nt,20nt] w.r.t. the annotated 3' splice sites for the 3' model.

A.4.1.6 HAL

The HAL model was adapted from `https://github.com/Alex-Rosenberg/cell-2015/tree/ca54d1117fd28375260bfde3d1b46f3d6074f306` by implementing the identical 5' splice-site scoring function into a Kipoi's model class with the `predict_on_batch` function. Model weights were obtained from the same repository and directly applied. We implemented a data-loader that takes the reference genome FASTA file and the genome annotation GTF file as inputs and returns k-mer counts of sequences from all regions [-80nt, 80nt] w.r.t. the annotated 5' splice sites.

A.4.1.7 Labranchor

The Labranchor model was obtained from `https://github.com/jpaggi/labranchor/tree/d0e232413cf39afdad7e438bef93f3cae6b816e1`. The Keras model implementa-

tion provided by the authors could directly be used for the Kipoi model. We implemented a data-loader that takes the reference genome FASTA file and the genome annotation GTF file as inputs and returns one-hot-encoded sequences of all regions [-70, 0] nt relative to the annotated 3' splice sites. Benchmarking transcription factor binding prediction models The benchmark was scripted with Snakemake v5.3.0 [216]. The complete Snakefile for the analysis described in this section is available at: `https://github.com/kipoi/manuscript/blob/master/src/tf-binding/Snakefile`.

A.4.2 Data and prediction command

The test set for transcription factor binding models was generated using 101bp contiguous intervals throughout chromosome 8 in the human genome assembly hg19. Each interval was labeled based on majority overlap with transcription factor ChIP-seq high-confidence peaks (IDR< 0.05) from the ENCODE-DREAM in-vivo transcription factor binding site prediction challenge (`http://synapse.org/encode`). Intervals in the hg19 blacklist regions (`https://www.encodeproject.org/annotations/ENCSR636HFF/`) were removed. CEBPB was evaluated in the HeLa-S3 (ENCFF002CSA), JUND in HepG2 (ENCSR000EEI), MAFK in K562 (ENCFF812QPN) and NANOG in H1-hESC (ENCFF379EPK) cell type. The additional files required by FactorNet (like the DNase accessibility track) were obtained from the URLs listed in `https://github.com/uci-cbcl/FactorNet/tree/bef6f6b38e81d362162a106dc8a726ecae910138/data#bigwig-files`. For models that require sequence lengths of more than 101 bp, we increased the size of labeled intervals. For example, to provide 1002 bp intervals for FactorNet, we subtracted 450 bp from start coordinates and added 451 bp to end coordinates. All model predictions were obtained by running the `kipoi predict` command in the individual conda environment for each model.

A.4.2.1 Accessible-only regions

In addition to chromosome-wide evaluation, the auPRC was computed only for regions overlapping DNase-seq signal peak regions in the corresponding cell-type by more than 50%. DNase-seq peaks were obtained from the relaxed peaks provided by the ENCODE-DREAM in-vivo transcription factor binding challenge (`http://synapse.org/encode`).

A.4.3 lsgkm-SVM training

lsgkm-SVM (v0.0.1) from Bioconda (`bioconda::ls-gkm=0.0.1`) was used for model training and prediction. The model was retrained on ENCODE datasets using files downloaded from the same source as mentioned in the publication [94]. Preprocessing of training was performed using the gkmSVM R-package (v0.79.0) using default parameters `genNullSeqs(..., nMaxTrials=20, xfold=1, genomeVersion='hg19',..)`. For training the 322 datasets with the most peaks were chosen, similar to the lsgkm-SVM publication. Training was performed with the parameters `gkmtrain -l 11 -d 3 -c 1 -T 16 -m 5120 -v 3`. For the final model chromosome 8 and 9 were held out from training to enable model

benchmarking comparable with the DeepSEA models. Trained models reached area under the receiver operating curve (auROC) similar to the original publication [94]:

	auROC	**# seqs**	**# seqs without 'N'**	**# seqs test set**
Mean	0.95	22699	22698	1854
Std	0.038	13499	13499	1210
Min	0.803	6067	13499	391
1st quartile	0.931	11211	11211	861
Median	0.970	19398	19396	1424
3rd quartile	0.982	34510	34510	2878
Max	0.996	71537	71535	6342

Table A.1: Predictive performance of the re-trained lsgkm-SVM model.

A.4.4 Transfer learning

A.4.4.1 Peak File Acquisition

We downloaded DNase files for 431 biosamples (cell lines or tissues) from Roadmap (`http://www.ncbi.nlm.nih.gov/geo/roadmap/epigenomics/`) and ENCODE (`https://www.encodeproject.org/`), and processed them separately to obtain a final dataset of binary labels (0/1) indicating chromatin accessibility in each interval of the combined accessibility region for each biosample. We processed the raw data as follows: The fastq files were aligned with BWA aln (v0.7.10), where all datasets were treated as single-end. Dynamic read trimming was set to 5, the seed length was 32, and 2 mismatches maximum were allowed in mapping. After mapping, reads were filtered to remove unmapped reads and mates, non-primary alignments, reads failing platform/vendor quality checks, and PCR/optical duplicates (`-F 1804`). Low quality reads (`MAPQ < 30`) were also removed. Duplicates were marked with Picard MarkDuplicates (v1.126) and removed. The final filtered file was converted to tagAlign format (BED 3+3) using bedtools' `bamtobed` (v2.27.1). Cross-correlation scores were obtained for each file using phantompeakqualtools (v1.1).

All files with a cross-correlation quality tag below 0 were discarded. For the ENCODE data generated from the Stam Lab protocol, the datasets were trimmed to 36 bp and technical replicates were combined. After removing mitochondrial and ambiguously mapped reads, the reads were randomly subsampled to a total of 50 million reads per sample. For the ENCODE data generated from the Crawford Lab protocol, the same procedure as above was performed, except reads were trimmed to 20 bp due to the different library generation protocol. For the Roadmap data, which was all generated by the Stam Lab protocol, the same procedure as above was performed with trimming to 36 bp. Reads from multiple files were combined and subsampled to 50 million reads in case the total number of reads was more than 50 million.

These trimmed, filtered, subsampled tagAlign files were then used to generate signal tracks and call peaks. Signal tracks and peaks were called with a loose threshold ($p <$

0.01) with MACS2 (v2.1.0) to generate bigwig files (fold enrichment and p-value) and Narrow Peak files, respectively. To obtain final peak sets, we performed pseudoreplicate subsampling on the pooled reads across all replicates (taking all reads from the final tagAligns and splitting in half by random assignment to two replicates) and running IDR (v2.0.3) with a p-value threshold of < 0.1 to get a consensus region set for each DNase experiment.

A.4.4.2 Data Preprocessing

We divided the genome into intervals of width 1000 bp using a stride of 200 bp. For each interval, we use the hg19 reference genome to extract the DNA sequence and assign a binary label of 0 (negative) or 1 (positive) for each of the 431 biosamples if the central 200 bp of the interval overlapped at least 50% of the accessibility IDR peak or if the accessibility IDR peak overlapped at least 50% of the the central 200 bp of the interval. This resulted in the 16,551,625 intervals and 431 binary labels per interval for each of the biosamples. We use data from chromosomes 1, 8, and 21 for testing, data from chromosome 9 for validation, and the remaining data for training the models.

We selected 10 biosamples to benchmark our transfer learning procedure by performing hierarchical clustering and randomly selecting one biosample from each of the 10 clusters. Selected biosamples were: common myeloid progenitor, GM12878, Jurkat clone E61, K562, mesendoderm, mesenchymal stem cell, cardiac mesoderm, thymus, lung, and brain.

A.4.4.3 Model Architecture

We trained 3 types of models predicting chromatin accessibility given DNA sequence: one multi-task model with randomly initialized weights predicting accessibility for 421 cell-types, and two types of single-task models trained on the remaining 10 cell-types: a model with randomly initialized weights and a model with weights transferred from the multi-task model. All models were convolutional neural networks (CNN) with the BASSET [64] architecture and were implemented in Keras version 1.2 using tensorflow-gpu version 1.0.0 backend.

A.4.4.4 Transfer Learning

We used the trained multi-task model and transferred the weights from all but the final classification layer to the transferred single-task architecture. We froze the weights of all layers but the final two, and replaced the final classification layer with a layer outputting a single prediction, instead of 421.

A.4.4.5 Model Training and Evaluation

Randomly initialized models and transferred models were trained using a categorical or binary cross-entropy loss, batch size of 256, epoch size of 2,500,000 and the ADAM optimizer50 with a learning rate of 0.0003. These hyper-parameters were manually selected

and were not optimized due to computational constraints. Early stopping monitoring auPRC on the validation set was used with patience of 4 epochs for models with randomly initialized weights and monitoring the validation cross-entropy loss with patience of 1 epoch for models with transferred weights. Models for individual cell types and transferred models for individual cell types were evaluated on the same test set for a given biosample (chromosomes 1, 8 and 21). For example in the GM12878 biosample, the test set contains 135,630 positives and 2,330,052 negatives, and the validation set contains 35,526 positives and 647,116 negatives. Predicting the molecular effects of genetic variants using interpretation plugins The presented variants were selected from the ClinVar release from April 2018. The selection involved performing variant effect prediction for all variants in the DeepSEA model and selecting the variant with the strongest negative predicted effect in GATA2 model outputs respectively. Mutation maps centered on those two variants were generated using the mutation map commands displayed in Figure 5.5D and implemented in the kipoi-veff plugin.

A.4.5 Predicting pathogenic splice variants by combining models

A.4.5.1 Data: ClinVar

The ClinVar release from April 2018 based on the reference genome GRCh37 was used (`ftp://ftp.ncbi.nlm.nih.gov/pub/clinvar/vcf_GRCh37/clinvar_20180429.vcf.gz`). Only variants in the range [-40, 10] nt around the splicing acceptor or variants in the range [-10, 10] nt around the splice donor of a protein coding gene (ENSEMBL GRCh37 v75 annotation) were used. The positive set comprises of variants classified as "Pathogenic" (6,310 variants) and the negative set comprises of variants classified as "Benign" (4,405 variants). Variants causing a premature stop codon were discarded. Per-variant pathogenicity/conservation scores (`CADD_raw`, `CADD_phred`, `phyloP46way_placental`, `phyloP46way_primate`) and the dbscSNV score were obtained by VEP [235] (v92). Spidex scores were obtained from ANNOVAR (`http://www.openbioinformatics.org/annovar/spidex_download_form.php`).

A.4.5.2 Features

Kipoi features: For specific Kipoi models the following features were produced:

- MaxEntScan/3prime, MaxEntScan/5prime, HAL
  - `<model>_ref`: Model prediction for the reference allele
  - `<model>_alt`: Model prediction for the alternative allele
- Labranchor
  - `labranchor_logit_ref`: (optional) Model prediction for the reference allele on the logit scale
  - `labranchor_logit_alt`: (optional) Model prediction for the alternative allele on the logit scale

- All models
  - `<model>_is_na`: 1 if model prediction is unavailable for the variant and 0 otherwise

These features were obtained by running the `kipoi veff score_variants` command on the variants table with `-s logit_ref logit_alt ref alt logit diff` formatted as a vcf file and then parsing the returned vcf files using `kipoi_veff.parsers.KipoiVCFParser`.

dbscSNV features:

- `dbscSNV_rf_score` - dbscSNV random forest score obtained with VEP
- `dbscSNV_rf_score_isna` - 1 if `dbscSNV_rf_score` is unavailable for the variant and 0 otherwise

SPIDEX features:

- `dpsi_max_tissue`, the maximum mutation-induced change in percentage-spliced in (PSI) across 16 tissue
- `dpsi_max_tissue_isna`, 1 if `dpsi_max_tissue` is unavailable for the variant and 0 otherwise
- `dpsi_zscore`, z-score transformed `dpsi_max_tissue`
- `dpsi_zscore_isna`, 1 if `dpsi_zscore` is unavailable for the variant and 0 otherwise

Conservation features: All obtained using VEP

- `CADD_raw`, Combined Annotation–Dependent Depletion score as described in [225]
- `CADD_phred`, CADD phred-like rank score based on whole genome CADD raw scores
- `phyloP46way_placental`, phyloP (phylogenetic p-values) conservation score based on the multiple alignments of 33 placental mammal genomes including human as described in [225].
- `phyloP46way_primate`, phyloP (phylogenetic p-values) conservation score based on the multiple alignments of 10 primate genomes including human.

NA values were zero-imputed and each feature was standardized to have mean of zero and variance of one.

Response variable: `ClinicalSignificance` was transformed into a binary classification variable with `Pathogenic` corresponding to class 1 and `Benign` corresponding to class 0.

A.4.5.3 Data: dbscSNV

Table S2 from the supplementary material of [89] was used to train and evaluate models in a 10-fold cross validation (2,959 variants, 1,164 from the positive class).

dbscSNV features (without conservation, described in [89])

- `PWM_ref`, `PWM_alt`,
- `MES_ref`, `MES_alt`,
- `NNSplice_ref`, `NNSplice_alt`,
- `HSF_ref`, `HSF_alt`,
- `GeneSplicer_ref`, `GeneSplicer_alt`,
- `GENSCAN_ref`, `GENSCAN_alt`,
- `NetGene2_ref`, `NetGene2_alt`,
- `SplicePredictor_ref`, `SplicePredictor_alt`

Kipoi model, conservation and SPIDEX features were the same as for the ClinVar dataset. Response variable: `Group` variable in the original table - `'Positive'==1` and `'Negative'==0`.

A.4.5.4 Meta-model and evaluation

Logistic regression implemented in scikit-learn with default parameters was used to build the meta model using different feature subsets. 10-fold cross-validation was used (implemented in `sklearn.model_selection.cross_validate`) to evaluate models using the auROC metric.

List of Figures

List of Tables

References

[1] Avsec, Ž. *et al.* The kipoi repository accelerates community exchange and reuse of predictive models for genomics. *Nature Biotechnology* (2019). URL `http://dx.doi.org/10.1038/s41587-019-0140-0`.

[2] Avsec, Ž., Barekatain, M., Cheng, J. & Gagneur, J. Modeling positional effects of regulatory sequences with spline transformations increases prediction accuracy of deep neural networks. *Bioinformatics* **34**, 1261–1269 (2017). URL `http://dx.doi.org/10.1093/bioinformatics/btx727`.

[3] Eraslan, G., Avsec, Ž., Gagneur, J. & Theis, F. J. Deep learning: new computational modelling techniques for genomics. *Nature Reviews Genetics* (2019). URL `http://dx.doi.org/10.1038/s41576-019-0122-6`.

[4] Cheng, J. *et al.* Mmsplice: modular modeling improves the predictions of genetic variant effects on splicing. *Genome Biology* **20** (2019). URL `http://dx.doi.org/10.1186/s13059-019-1653-z`.

[5] Brechtmann, F. *et al.* Outrider: A statistical method for detecting aberrantly expressed genes in rna sequencing data. *The American Journal of Human Genetics* **103**, 907–917 (2018).

[6] Shrikumar, A. *et al.* TF-MoDISco v0.4.2.2-alpha: Technical Note (2018). `1811.00416v2`.

[7] Cheng, J., Çelik, M. H., Nguyen, Y. D., Avsec, Ž. & Gagneur, J. Cagi5 splicing challenge: Improved exon skipping and intron retention predictions with mmsplice. *Human Mutation* (2019). URL `http://dx.doi.org/10.1002/humu.23788`.

[8] Cheng, J., Maier, K. C., Avsec, Ž., Rus, P. & Gagneur, J. Cis-regulatory elements explain most of the mrna stability variation across genes in yeast. *RNA* **23**, 1648–1659 (2017). URL `http://dx.doi.org/10.1261/rna.062224.117`.

[9] Ait-El-Mkadem, S. *et al.* Mutations in mdh2, encoding a krebs cycle enzyme, cause early-onset severe encephalopathy. *American journal of human genetics* **100** (2016).

[10] Shendure, J., Findlay, G. M. & Snyder, M. W. Genomic medicine–progress, pitfalls, and promise. *Cell* **177**, 45–57 (2019). URL `http://dx.doi.org/10.1016/j.cell.2019.02.003`.

[11] Harrow, J. *et al.* GENCODE: the reference human genome annotation for The ENCODE Project. *Genome research* **22**, 1760–1774 (2012).

[12] Soutourina, J. Transcription regulation by the mediator complex. *Nature Reviews Molecular Cell Biology* **19**, 262 (2018).

[13] Slattery, M. *et al.* Absence of a simple code: how transcription factors read the genome. *Trends in biochemical sciences* **39**, 381–399 (2014).

[14] Long, H. K., Prescott, S. L. & Wysocka, J. Ever-changing landscapes: Transcriptional enhancers in development and evolution. *Cell* **167**, 1170–1187 (2016). URL `http://linkinghub.elsevier.com/retrieve/pii/{S009286741631251X}`.

[15] Reiter, F., Wienerroither, S. & Stark, A. Combinatorial function of transcription factors and cofactors. *Current Opinion in Genetics & Development* **43**, 73–81 (2017). URL `http://dx.doi.org/10.1016/j.gde.2016.12.007`.

[16] Spitz, F. & Furlong, E. E. M. Transcription factors: from enhancer binding to developmental control. *Nature Reviews. Genetics* **13**, 613–626 (2012). URL `http://dx.doi.org/10.1038/nrg3207`.

[17] Stormo, G. D. DNA binding sites: representation and discovery. *Bioinformatics* **16**, 16–23 (2000).

[18] Weirauch, M. T. *et al.* Evaluation of methods for modeling transcription factor sequence specificity. *Nature Biotechnology* **31**, 126–134 (2013). URL `http://dx.doi.org/10.1038/nbt.2486`.

[19] Bussemaker, H. J., Foat, B. C. & Ward, L. D. Predictive modeling of genome-wide mrna expression: From modules to molecules. *Annual Review of Biophysics and Biomolecular Structure* **36**, 329–347 (2007). URL `http://dx.doi.org/10.1146/annurev.biophys.36.040306.132725`.

[20] Badis, G. *et al.* Diversity and complexity in dna recognition by transcription factors. *Science* **324**, 1720–1723 (2009). URL `http://dx.doi.org/10.1126/science.1162327`.

[21] Stormo, G. D. & Zhao, Y. Determining the specificity of protein–dna interactions. *Nature Reviews Genetics* **11**, 751–760 (2010). URL `http://dx.doi.org/10.1038/nrg2845`.

[22] Jolma, A. *et al.* DNA-binding specificities of human transcription factors. *Cell* **152**, 327–339 (2013). URL `http://dx.doi.org/10.1016/j.cell.2012.12.009`.

[23] White, M. A., Myers, C. A., Corbo, J. C. & Cohen, B. A. Massively parallel in vivo enhancer assay reveals that highly local features determine the cis-regulatory function of chip-seq peaks. *Proceedings of the National Academy of Sciences* **110**, 11952–11957 (2013). URL `http://dx.doi.org/10.1073/pnas.1307449110`.

[24] Zambelli, F., Pesole, G. & Pavesi, G. Motif discovery and transcription factor binding sites before and after the next-generation sequencing era. *Briefings in bioinformatics* **14**, 225–237 (2012).

[25] Meijsing, S. H. *et al.* Dna binding site sequence directs glucocorticoid receptor structure and activity. *Science* **324**, 407–410 (2009). URL `http://dx.doi.org/10.1126/science.1164265`.

[26] Rohs, R. *et al.* The role of dna shape in protein–dna recognition. *Nature* **461**, 1248–1253 (2009). URL `http://dx.doi.org/10.1038/nature08473`.

[27] Kim, S. *et al.* Probing allostery through dna. *Science* **339**, 816–819 (2013). URL `http://dx.doi.org/10.1126/science.1229223`.

[28] Watson, L. C. *et al.* The glucocorticoid receptor dimer interface allosterically transmits sequence-specific dna signals. *Nature Structural & Molecular Biology* **20**, 876–883 (2013). URL `http://dx.doi.org/10.1038/nsmb.2595`.

[29] Slattery, M. *et al.* Cofactor binding evokes latent differences in dna binding specificity between hox proteins. *Cell* **147**, 1270–1282 (2011). URL `http://dx.doi.org/10.1016/j.cell.2011.10.053`.

[30] Siggers, T., Duyzend, M. H., Reddy, J., Khan, S. & Bulyk, M. L. Non-dna-binding cofactors enhance dna-binding specificity of a transcriptional regulatory complex. *Molecular Systems Biology* **7**, 555–555 (2014). URL `http://dx.doi.org/10.1038/msb.2011.89`.

[31] Panne, D. The enhanceosome. *Current Opinion in Structural Biology* **18**, 236–242 (2008). URL `http://dx.doi.org/10.1016/j.sbi.2007.12.002`.

[32] Wasson, T. & Hartemink, A. J. An ensemble model of competitive multi-factor binding of the genome. *Genome Research* **19**, 2101–2112 (2009). URL `http://dx.doi.org/10.1101/gr.093450.109`.

[33] Kitayner, M., Suad, O., Rozenberg, H. & Shakked, Z. Diversity in dna recognition by p53 revealed by crystal structures with hoogsteen base pairs (p53-dna complex 1) (2010). URL `http://dx.doi.org/10.2210/pdb3igl/pdb`.

[34] Glatt, S., Alfieri, C. & Müller, C. W. Recognizing and remodeling the nucleosome. *Current Opinion in Structural Biology* **21**, 335–341 (2011). URL `http://dx.doi.org/10.1016/j.sbi.2011.02.003`.

[35] Barozzi, I. *et al.* Coregulation of transcription factor binding and nucleosome occupancy through dna features of mammalian enhancers. *Molecular Cell* **54**, 844–857 (2014). URL `http://dx.doi.org/10.1016/j.molcel.2014.04.006`.

[36] Liu, X., Lee, C.-K., Granek, J. A., Clarke, N. D. & Lieb, J. D. Whole-genome comparison of leu3 binding in vitro and in vivo reveals the importance of nucleosome occupancy in target site selection. *Genome Research* **16**, 1517–1528 (2006). URL `http://dx.doi.org/10.1101/gr.5655606`.

[37] Kaplan, N. *et al.* The dna-encoded nucleosome organization of a eukaryotic genome. *Nature* **458**, 362–366 (2008). URL `http://dx.doi.org/10.1038/nature07667`.

[38] Bai, L. & Morozov, A. V. Gene regulation by nucleosome positioning. *Trends in Genetics* **26**, 476–483 (2010). URL `http://dx.doi.org/10.1016/j.tig.2010.08.003`.

[39] Kaplan, T. *et al.* Quantitative models of the mechanisms that control genome-wide patterns of transcription factor binding during early drosophila development. *PLoS Genetics* **7**, e1001290 (2011). URL `http://dx.doi.org/10.1371/journal.pgen.1001290`.

[40] Pique-Regi, R. *et al.* Accurate inference of transcription factor binding from dna sequence and chromatin accessibility data. *Genome Research* **21**, 447–455 (2010). URL `http://dx.doi.org/10.1101/gr.112623.110`.

[41] Wang, J. *et al.* Sequence features and chromatin structure around the genomic regions bound by 119 human transcription factors. *Genome Research* **22**, 1798–1812 (2012). URL `http://dx.doi.org/10.1101/gr.139105.112`.

[42] Miller, J. A. & Widom, J. Collaborative competition mechanism for gene activation in vivo. *Molecular and Cellular Biology* **23**, 1623–1632 (2003). URL `http://dx.doi.org/10.1128/mcb.23.5.1623-1632.2003`.

[43] Mirny, L. A. Nucleosome-mediated cooperativity between transcription factors. *Proceedings of the National Academy of Sciences* **107**, 22534–22539 (2010). URL `http://dx.doi.org/10.1073/pnas.0913805107`.

[44] Lazarovici, A. *et al.* Probing dna shape and methylation state on a genomic scale with dnase i. *Proceedings of the National Academy of Sciences* **110**, 6376–6381 (2013). URL `http://dx.doi.org/10.1073/pnas.1216822110`.

[45] Berger, M. F. *et al.* Compact, universal dna microarrays to comprehensively determine transcription-factor binding site specificities. *Nature Biotechnology* **24**, 1429–1435 (2006). URL `http://dx.doi.org/10.1038/nbt1246`.

[46] Berger, M. F. & Bulyk, M. L. Universal protein-binding microarrays for the comprehensive characterization of the dna-binding specificities of transcription factors. *Nature Protocols* **4**, 393–411 (2009). URL `http://dx.doi.org/10.1038/nprot.2008.195`.

[47] Maerkl, S. J. & Quake, S. R. A systems approach to measuring the binding energy landscapes of transcription factors. *Science* **315**, 233–237 (2007). URL http://dx.doi.org/10.1126/science.1131007.

[48] Fordyce, P. M. *et al.* De novo identification and biophysical characterization of transcription-factor binding sites with microfluidic affinity analysis. *Nature Biotechnology* **28**, 970–975 (2010). URL http://dx.doi.org/10.1038/nbt.1675.

[49] Zhao, Y., Granas, D. & Stormo, G. D. Inferring binding energies from selected binding sites. *PLoS Computational Biology* **5**, e1000590 (2009). URL http://dx.doi.org/10.1371/journal.pcbi.1000590.

[50] Jolma, A. *et al.* Multiplexed massively parallel selex for characterization of human transcription factor binding specificities. *Genome Research* **20**, 861–873 (2010). URL http://dx.doi.org/10.1101/gr.100552.109.

[51] Ren, B. Genome-wide location and function of dna binding proteins. *Science* **290**, 2306–2309 (2000). URL http://dx.doi.org/10.1126/science.290.5500.2306.

[52] Johnson, D. S., Mortazavi, A., Myers, R. M. & Wold, B. Genome-wide mapping of in vivo protein-DNA interactions. *Science* **316**, 1497–1502 (2007).

[53] Rhee, H. S. & Pugh, B. F. Comprehensive genome-wide protein-DNA interactions detected at single-nucleotide resolution. *Cell* **147**, 1408–1419 (2011). URL http://dx.doi.org/10.1016/j.cell.2011.11.013.

[54] He, Q., Johnston, J. & Zeitlinger, J. Chip-nexus enables improved detection of in vivo transcription factor binding footprints. *Nature biotechnology* **33**, 395 (2015).

[55] Zhou, X. *et al.* The human epigenome browser at washington university. *Nature Methods* **8**, 989–990 (2011). URL http://dx.doi.org/10.1038/nmeth.1772.

[56] The ENCODE Project Consortium *et al.* An integrated encyclopedia of DNA elements in the human genome. *Nature* **489**, 57 (2012).

[57] Quail, M. A. *et al.* A large genome center's improvements to the illumina sequencing system. *Nature methods* **5**, 1005 (2008).

[58] Zhang, Y. *et al.* Model-based analysis of chip-seq (macs). *Genome biology* **9**, R137 (2008).

[59] Guo, Y., Mahony, S. & Gifford, D. K. High resolution genome wide binding event finding and motif discovery reveals transcription factor spatial binding constraints. *PLoS Computational Biology* **8**, e1002638 (2012). URL http://dx.doi.org/10.1371/journal.pcbi.1002638.

[60] Kharchenko, P. V., Tolstorukov, M. Y. & Park, P. J. Design and analysis of chip-seq experiments for dna-binding proteins. *Nature biotechnology* **26**, 1351 (2008).

[61] Stormo, G. D., Schneider, T. D., Gold, L. & Ehrenfeucht, A. Use of the 'perceptron' algorithm to distinguish translational initiation sites in e. coli. *Nucleic Acids Res.* **10**, 2997–3011 (1982).

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

[63] Zhou, J. & Troyanskaya, O. G. Predicting effects of noncoding variants with deep learning-based sequence model. *Nature methods* **12**, 931–4 (2015). URL `http://www.nature.com/doifinder/10.1038/nmeth.3547http://www.ncbi.nlm.nih.gov/pubmed/26301843http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4768299`.

[64] Kelley, D. R., Snoek, J. & Rinn, J. L. Basset: Learning the regulatory code of the accessible genome with deep convolutional neural networks. *Genome Research* **26**, 990–999 (2016).

[65] Pugin, J. *et al.* Cd14 is a pattern recognition receptor. *Immunity* **1**, 509–516 (1994). URL `http://dx.doi.org/10.1016/1074-7613(94)90093-0`.

[66] Berg, O. G. & von Hippel, P. H. Selection of dna binding sites by regulatory proteins: Statistical-mechanical theory and application to operators and promoters. *Journal of molecular biology* **193**, 723–743 (1987).

[67] Bailey, T. L. *et al.* MEME SUITE: tools for motif discovery and searching. *Nucleic Acids Research* **37**, W202–8 (2009). URL `http://dx.doi.org/10.1093/nar/gkp335`.

[68] Bailey, T. L. & Machanick, P. Inferring direct DNA binding from ChIP-seq. *Nucleic Acids Research* **40**, e128 (2012). URL `http://dx.doi.org/10.1093/nar/gks433`.

[69] Schneider, T. D., Stormo, G. D., Gold, L. & Ehrenfeucht, A. Information content of binding sites on nucleotide sequences. *Journal of Molecular Biology* **188**, 415–431 (1986). URL `http://dx.doi.org/10.1016/0022-2836(86)90165-8`.

[70] Schneider, T. D. & Stephens, R. Sequence logos: a new way to display consensus sequences. *Nucleic Acids Research* **18**, 6097–6100 (1990). URL `http://dx.doi.org/10.1093/nar/18.20.6097`.

[71] Salzberg, S. L., Delcher, A. L., Kasif, S. & White, O. Microbial gene identification using interpolated markov models. *Nucleic Acids Research* **26**, 544–548 (1998). URL `http://dx.doi.org/10.1093/nar/26.2.544`.

[72] Siebert, M. & Söding, J. Bayesian markov models consistently outperform pwms at predicting motifs in nucleotide sequences. *Nucleic Acids Research* **44**, 6055–6069 (2016). URL `http://dx.doi.org/10.1093/nar/gkw521`.

[73] Foat, B. C., Morozov, A. V. & Bussemaker, H. J. Statistical mechanical modeling of genome-wide transcription factor occupancy data by matrixreduce. *Bioinformatics* **22**, e141–e149 (2006). URL `http://dx.doi.org/10.1093/bioinformatics/btl223`.

[74] Riley, T. R., Lazarovici, A., Mann, R. S. & Bussemaker, H. J. Building accurate sequence-to-affinity models from high-throughput in vitro protein-dna binding data using featurereduce. *eLife* **4** (2015). URL `http://dx.doi.org/10.7554/elife.06397`.

[75] Quang, D. & Xie, X. FactorNet: a deep learning framework for predicting cell type specific transcription factor binding from nucleotide-resolution sequential data (2017).

[76] Song, L. & Crawford, G. E. Dnase-seq: a high-resolution technique for mapping active gene regulatory elements across the genome from mammalian cells. *Cold Spring Harbor Protocols* **2010**, pdb–prot5384 (2010).

[77] Buenrostro, J. D., Wu, B., Chang, H. Y. & Greenleaf, W. J. Atac-seq: a method for assaying chromatin accessibility genome-wide. *Current protocols in molecular biology* **109**, 21–29 (2015).

[78] Skene, P. J. & Henikoff, S. An efficient targeted nuclease strategy for high-resolution mapping of dna binding sites. *Elife* **6**, e21856 (2017).

[79] Nagalakshmi, U. *et al.* The transcriptional landscape of the yeast genome defined by rna sequencing. *Science* **320**, 1344–1349 (2008). URL `http://dx.doi.org/10.1126/science.1158441`.

[80] Lieberman-Aiden, E. *et al.* Comprehensive mapping of long-range interactions reveals folding principles of the human genome. *Science* **326**, 289–293 (2009). URL `http://dx.doi.org/10.1126/science.1181369`.

[81] Krakau, S., Richard, H. & Marsico, A. Pureclip: capturing target-specific protein–rna interaction footprints from single-nucleotide clip-seq data. *Genome Biology* **18**, 240 (2017). URL `https://doi.org/10.1186/s13059-017-1364-2`.

[82] Van Nostrand, E. L. *et al.* Robust transcriptome-wide discovery of RNA-binding protein binding sites with enhanced CLIP (eCLIP). *Nature methods* **13**, 1–9 (2016). URL `http://www.nature.com/doifinder/10.1038/nmeth.3810%5Cnhttp://www.ncbi.nlm.nih.gov/pubmed/27018577`.

[83] Schwanhäusser, B. *et al.* Global quantification of mammalian gene expression control. *Nature* **473**, 337–342 (2011). URL `http://dx.doi.org/10.1038/nature10098`.

[84] Schwalb, B. *et al.* Tt-seq maps the human transient transcriptome. *Science* **352**, 1225–1228 (2016). URL `http://dx.doi.org/10.1126/science.aad9841`.

[85] Melnikov, A. *et al.* Systematic dissection and optimization of inducible enhancers in human cells using a massively parallel reporter assay. *Nature Biotechnology* **30**, 271–277 (2012). URL `http://dx.doi.org/10.1038/nbt.2137`.

[86] Cong, L. *et al.* Multiplex genome engineering using crispr/cas systems. *Science* **339**, 819–823 (2013). URL `http://dx.doi.org/10.1126/science.1231143`.

[87] Mali, P. *et al.* Rna-guided human genome engineering via cas9. *Science* **339**, 823–826 (2013). URL `http://dx.doi.org/10.1126/science.1232033`.

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

[89] Jian, X., Boerwinkle, E. & Liu, X. In silico prediction of splice-altering single nucleotide variants in the human genome. *Nucleic Acids Res.* **42**, 13534–13544 (2014).

[90] Rosenberg, A. B., Patwardhan, R. P., Shendure, J. & Seelig, G. Learning the sequence determinants of alternative splicing from millions of random sequences. *Cell* **163**, 698–711 (2015).

[91] Paggi, J. M. & Bejerano, G. A sequence-based, deep learning model accurately predicts RNA splicing branchpoints. *bioRxiv* (2017).

[92] Sandelin, A. Jaspar: an open-access database for eukaryotic transcription factor binding profiles. *Nucleic Acids Research* **32**, 91D–94 (2004). URL `http://dx.doi.org/10.1093/nar/gkh012`.

[93] Kulakovskiy, I. V. *et al.* HOCOMOCO: expansion and enhancement of the collection of transcription factor binding sites models. *Nucleic Acids Res.* **44**, D116–25 (2016).

[94] Lee, D. LS-GKM: a new gkm-SVM for large-scale datasets. *Bioinformatics* **32**, 2196–2198 (2016).

[95] Kelley, D. R. *et al.* Sequential regulatory activity prediction across chromosomes with convolutional neural networks. *Genome Res.* **28**, 739–750 (2018).

[96] Zeng, H. & Gifford, D. K. Predicting the impact of non-coding variants on DNA methylation. *Nucleic Acids Res.* **45**, e99 (2017).

[97] Angermueller, C., Lee, H. J., Reik, W. & Stegle, O. DeepCpG: accurate prediction of single-cell DNA methylation states using deep learning. *Genome Biology* **18**, 67 (2017). URL `http://genomebiology.biomedcentral.com/articles/10.1186/s13059-017-1189-z`.

[98] Zhou, J. *et al.* Deep learning sequence-based ab initio prediction of variant effects on expression and disease risk. *Nat. Genet.* **50**, 1171–1179 (2018).

[99] Pan, X. & Shen, H.-B. RNA-protein binding motifs mining with a new hybrid deep learning based cross-domain knowledge integration approach. *BMC bioinformatics* **18**, 136 (2017).

[100] Lee, B., Baek, J., Park, S. & Yoon, S. deeptarget: End-to-end learning framework for microRNA target prediction using deep recurrent neural networks. In *Proceedings of the 7th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, 434–442 (ACM, 2016).

[101] Park, S., Min, S., Choi, H. & Yoon, S. deepMiRGene: Deep neural network based precursor microRNA prediction (2016). 1605.00017.

[102] Yeo, G. & Burge, C. B. Maximum entropy modeling of short sequence motifs with applications to RNA splicing signals. *J. Comput. Biol.* **11**, 377–394 (2004).

[103] Bogard, N., Linder, J., Rosenberg, A. B. & Seelig, G. Predicting the impact of cis-regulatory variation on alternative polyadenylation (2018). URL http://dx.doi.org/10.1101/300061.

[104] Sample, P. J. *et al.* Human 5' utr design and variant effect prediction from a massively parallel translation assay. *bioRxiv* 310375 (2018).

[105] Boycott, K. M. *et al.* A diagnosis for all rare genetic diseases: The horizon and the next frontiers. *Cell* **177**, 32–37 (2019). URL http://dx.doi.org/10.1016/j.cell.2019.02.040.

[106] Lappalainen, T., Scott, A. J., Brandt, M. & Hall, I. M. Genomic analysis in the age of human genome sequencing. *Cell* **177**, 70–84 (2019).

[107] Maurano, M. T. *et al.* Systematic localization of common disease-associated variation in regulatory dna. *Science* **337**, 1190–1195 (2012). URL http://dx.doi.org/10.1126/science.1222794.

[108] Young, R. A. Control of the embryonic stem cell state. *Cell* **144**, 940–954 (2011). URL http://dx.doi.org/10.1016/j.cell.2011.01.032.

[109] Breiman, L. Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Stat. Sci.* **16**, 199–231 (2001).

[110] Shrikumar, A., Greenside, P., Shcherbina, A. & Kundaje, A. Not just a black box: Learning important features through propagating activation differences (2016). 1605.01713.

[111] Murphy, K. P. *Machine Learning: A Probabilistic Perspective* (MIT Press, 2012).

[112] Zou, H. & Hastie, T. Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)* **67**, 301–320 (2005).

[113] Tibshirani, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)* **58**, 267–288 (1996).

[114] Bergstra, J. & Bengio, Y. Random search for hyper-parameter optimization. *Journal of Machine Learning Research* **13**, 281–305 (2012).

[115] Snoek, J., Larochelle, H. & Adams, R. P. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, 2951–2959 (2012).

[116] Breiman, L. Random forests. *Mach. Learn.* **45**, 5–32 (2001).

[117] Goodfellow, I., Bengio, Y. & Courville, A. *Deep Learning* (MIT Press, 2016).

[118] He, K., Zhang, X., Ren, S. & Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, 1026–1034 (2015).

[119] Glorot, X. & Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 249–256 (2010).

[120] Rumelhart, D. E., Hinton, G. E., Williams, R. J. *et al.* Learning representations by back-propagating errors. *Cognitive modeling* **5**, 1 (1988).

[121] Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization (2014). `1412.6980`.

[122] Bottou, L. Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nımes* **91**, 12 (1991).

[123] Bengio, Y. Practical recommendations for gradient-based training of deep architectures (2012). `1206.5533`.

[124] D'haeseleer, P. What are DNA sequence motifs? *Nat. Biotechnol.* **24**, 423–425 (2006).

[125] Bailey, T. L., Elkan, C. *et al.* Fitting a mixture model by expectation maximization to discover motifs in bipolymers (1994).

[126] Van Den Oord, A. *et al.* Wavenet: A generative model for raw audio. *SSW* **125** (2016).

[127] Zeiler, M. D., Krishnan, D., Taylor, G. W. & Fergus, R. Deconvolutional networks. In *Cvpr*, vol. 10, 7 (2010).

[128] Battaglia, P. W. *et al.* Relational inductive biases, deep learning, and graph networks (2018). `1806.01261`.

[129] Ioffe, S. & Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR* **abs/1502.03167** (2015). URL `http://arxiv.org/abs/1502.03167`.

[130] Salimans, T. & Kingma, D. P. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems*, 901–909 (2016).

[131] Lei Ba, J., Kiros, J. R. & Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

[132] Wu, Y. & He, K. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 3–19 (2018).

[133] He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778 (2016).

[134] Friedman, J. H. Greedy function approximation: A gradient boosting machine. *Ann. Stat.* **29**, 1189–1232 (2001).

[135] Li, H., Xu, Z., Taylor, G., Studer, C. & Goldstein, T. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems*, 6389–6399 (2018).

[136] Huang, G., Liu, Z., Van Der Maaten, L. & Weinberger, K. Q. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4700–4708 (2017).

[137] Srivastava, R. K., Greff, K. & Schmidhuber, J. Highway networks. *arXiv preprint arXiv:1505.00387* (2015).

[138] Zitnik, M. *et al.* Machine learning for integrating data in biology and medicine: Principles, practice, and opportunities. *Inf. Fusion* **50**, 71–91 (2018).

[139] Russakovsky, O. *et al.* ImageNet large scale visual recognition challenge (2014). `1409.0575`.

[140] Sundararajan, M., Taly, A. & Yan, Q. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 3319–3328 (JMLR. org, 2017).

[141] Mahony, S. & Pugh, B. F. Protein–dna binding in high-resolution. *Critical reviews in biochemistry and molecular biology* **50**, 269–283 (2015).

[142] Albert, I. *et al.* Translational and rotational settings of H2A.Z nucleosomes across the Saccharomyces cerevisiae genome. *Nature* **446**, 572 (2007). URL `http://dx.doi.org/10.1038/nature05632`.

[143] Barski, A. *et al.* High-Resolution Profiling of Histone Methylations in the Human Genome. *Cell* **129**, 823–837 (2007). URL `http://dx.doi.org/10.1016/j.cell.2007.05.009`.

[144] Mikkelsen, T. S. *et al.* Genome-wide maps of chromatin state in pluripotent and lineage-committed cells. *Nature* **448**, 553–560 (2007). URL `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2921165/`.

[145] Robertson, G. *et al.* Genome-wide profiles of STAT1 DNA association using chromatin immunoprecipitation and massively parallel sequencing. *Nature Methods* **4**, 651 (2007). URL `http://dx.doi.org/10.1038/nmeth1068`.

[146] Landt, S. G. *et al.* Chip-seq guidelines and practices of the encode and modencode consortia. *Genome research* **22**, 1813–1831 (2012).

[147] Schones, D. E. *et al.* Dynamic regulation of nucleosome positioning in the human genome. *Cell* **132**, 887–898 (2008).

[148] Buenrostro, J. D. *et al.* Single-cell chromatin accessibility reveals principles of regulatory variation. *Nature* **523**, 486 (2015).

[149] Gerstein, M. B. *et al.* Architecture of the human regulatory network derived from encode data. *Nature* **489**, 91 (2012).

[150] Kundaje, A. *et al.* Integrative analysis of 111 reference human epigenomes. *Nature* **518**, 317 (2015).

[151] Siggers, T. & Gordaˆn, R. Protein–dna binding: complexities and multi-protein codes. *Nucleic acids research* **42**, 2099–2111 (2013).

[152] Cai, H. N., Arnosti, D. N. & Levine, M. Long-range repression in the drosophila embryo. *Proceedings of the National Academy of Sciences* **93**, 9309–9314 (1996).

[153] Markstein, M. & Levine, M. Decoding cis-regulatory dnas in the drosophila genome. *Current opinion in genetics & development* **12**, 601–606 (2002).

[154] Merika, M. & Thanos, D. Enhanceosomes. *Current Opinion in Genetics & Development* **11**, 205–208 (2001). URL `http://dx.doi.org/10.1016/S0959-{437X}(00)00180-5`.

[155] Soufi, A. *et al.* Pioneer transcription factors target partial DNA motifs on nucleosomes to initiate reprogramming. *Cell* **161**, 555–568 (2015). URL `http://dx.doi.org/10.1016/j.cell.2015.03.017`.

[156] Lambert, S. A. *et al.* The human transcription factors. *Cell* **172**, 650–665 (2018). URL `http://linkinghub.elsevier.com/retrieve/pii/S0092867418301065`.

[157] Papagianni, A. *et al.* Capicua controls toll/IL-1 signaling targets independently of RTK regulation. *Proceedings of the National Academy of Sciences of the United States of America* **115**, 1807–1812 (2018). URL http://dx.doi.org/10.1073/pnas.1713930115.

[158] Starick, S. R. *et al.* ChIP-exo signal associated with DNA-binding motifs provides insight into the genomic binding of the glucocorticoid receptor and cooperating transcription factors. *Genome Research* **25**, 825–835 (2015). URL http://dx.doi.org/10.1101/gr.185157.114.

[159] Rozowsky, J. *et al.* Peakseq enables systematic scoring of chip-seq experiments relative to controls. *Nature biotechnology* **27**, 66 (2009).

[160] Guo, Y. *et al.* Discovering homotypic binding events at high spatial resolution. *Bioinformatics* **26**, 3028–3034 (2010).

[161] Zeng, X. *et al.* jmosaics: joint analysis of multiple chip-seq datasets. *Genome biology* **14**, R38 (2013).

[162] Hartonen, T., Sahu, B., Dave, K., Kivioja, T. & Taipale, J. PeakXus: comprehensive transcription factor binding site discovery from ChIP-Nexus and ChIP-Exo experiments. *Bioinformatics* **32**, i629–i638 (2016). URL https://doi.org/10.1093/bioinformatics/btw448. http://oup.prod.sis.lan/bioinformatics/article-pdf/32/17/i629/24151274/btw448.pdf.

[163] Luehr, S., Hartmann, H. & Söding, J. The XXmotif web server for eXhaustive, weight matriX-based motif discovery in nucleotide sequences. *Nucleic Acids Research* **40**, W104–W109 (2012). URL https://doi.org/10.1093/nar/gks602. http://oup.prod.sis.lan/nar/article-pdf/40/W1/W104/4936010/gks602.pdf.

[164] Yamada, N., Lai, W. K. M., Farrell, N., Pugh, B. F. & Mahony, S. Characterizing protein-DNA binding event subtypes in ChIP-exo data. *Bioinformatics* **35**, 903–913 (2019). URL http://dx.doi.org/10.1093/bioinformatics/bty703.

[165] Terooatea, T. W., Pozner, A. & Buck-Koehntop, B. A. PAtCh-cap: input strategy for improving analysis of ChIP-exo data sets and beyond. *Nucleic Acids Research* **44**, e159 (2016). URL http://dx.doi.org/10.1093/nar/gkw741.

[166] Yardımcı, G. G., Frank, C. L., Crawford, G. E. & Ohler, U. Explicit DNase sequence bias modeling enables high-resolution transcription factor footprint detection. *Nucleic Acids Research* **42**, 11865–11878 (2014). URL https://doi.org/10.1093/nar/gku810. http://oup.prod.sis.lan/nar/article-pdf/42/19/11865/14121404/gku810.pdf.

[167] Li, Q., Brown, J. B., Huang, H., Bickel, P. J. *et al.* Measuring reproducibility of high-throughput experiments. *The annals of applied statistics* **5**, 1752–1779 (2011).

[168] Chollet, F. & Others. Keras. `https://github.com/fchollet/keras` (2015).

[169] Soldner, F. & Jaenisch, R. Stem cells, genome editing, and the path to translational medicine. *Cell* **175**, 615–632 (2018). URL `https://linkinghub.elsevier.com/retrieve/pii/S0092867418311826`.

[170] Theunissen, T. W. & Jaenisch, R. Mechanisms of gene regulation in human embryos and pluripotent stem cells. *Development* **144**, 4496–4509 (2017). URL `http://dx.doi.org/10.1242/dev.157404`.

[171] Festuccia, N. *et al.* Esrrb extinction triggers dismantling of naïve pluripotency and marks commitment to differentiation. *The EMBO Journal* **37** (2018). URL `http://dx.doi.org/10.15252/embj.201695476`.

[172] Moorthy, S. D. *et al.* Enhancers and super-enhancers have an equivalent regulatory role in embryonic stem cells through regulation of single or multiple genes. *Genome Research* **27**, 246–258 (2017). URL `http://dx.doi.org/10.1101/gr.210930.116`.

[173] Novo, C. L. *et al.* Long-range enhancer interactions are prevalent in mouse embryonic stem cells and are reorganized upon pluripotent state transition. *Cell reports* **22**, 2615–2627 (2018). URL `http://linkinghub.elsevier.com/retrieve/pii/S2211124718302213`.

[174] Whyte, W. A. *et al.* Enhancer decommissioning by LSD1 during embryonic stem cell differentiation. *Nature* **482**, 221–225 (2012). URL `http://dx.doi.org/10.1038/nature10805`.

[175] King, D. M., Maricque, B. B. & Cohen, B. A. Synthetic and genomic regulatory elements reveal aspects of cis-regulatory grammar in mouse embryonic stem cells. *BioRxiv* (2018). URL `http://biorxiv.org/lookup/doi/10.1101/398107`.

[176] Dorighi, K. M. *et al.* Mll3 and mll4 facilitate enhancer RNA synthesis and transcription from promoters independently of H3K4 monomethylation. *Molecular Cell* **66**, 568–576.e4 (2017). URL `http://linkinghub.elsevier.com/retrieve/pii/S1097276517302745`.

[177] Wnuk, K. *et al.* Predicting dna accessibility in the pan-cancer tumor genome using rna-seq, wgs, and deep learning. *bioRxiv* 229385 (2018).

[178] Blondel, V. D., Guillaume, J.-L., Lambiotte, R. & Lefebvre, E. The louvain method for community detection in large networks. *J of Statistical Mechanics: Theory and Experiment* **10**, P10008 (2011).

[179] Ward, J. H. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association* **58**, 236–244 (1963). URL `http://www.tandfonline.com/doi/abs/10.1080/01621459.1963.10500845`.

[180] Bar-Joseph, Z., Gifford, D. K. & Jaakkola, T. S. Fast optimal leaf ordering for hierarchical clustering. *Bioinformatics* **17 Suppl 1**, S22–9 (2001). URL http://dx.doi.org/10.1093/bioinformatics/17.suppl_1.S22.

[181] Chen, X. *et al.* Integration of external signaling pathways with the core transcriptional network in embryonic stem cells. *Cell* **133**, 1106–1117 (2008). URL http://dx.doi.org/10.1016/j.cell.2008.04.043.

[182] Gagliardi, A. *et al.* A direct physical interaction between nanog and sox2 regulates embryonic stem cell self-renewal. *The EMBO Journal* **32**, 2231–2247 (2013). URL http://dx.doi.org/10.1038/emboj.2013.161.

[183] He, X. *et al.* A biophysical model for analysis of transcription factor interaction and binding site arrangement from genome-wide binding data. *Plos One* **4**, e8155 (2009). URL http://dx.doi.org/10.1371/journal.pone.0008155.

[184] Loh, Y.-H. *et al.* The oct4 and nanog transcription network regulates pluripotency in mouse embryonic stem cells. *Nature Genetics* **38**, 431–440 (2006). URL http://dx.doi.org/10.1038/ng1760.

[185] Mitsui, K. *et al.* The homeoprotein nanog is required for maintenance of pluripotency in mouse epiblast and ES cells. *Cell* **113**, 631–642 (2003). URL http://dx.doi.org/10.1016/S0092-8674(03)00393-3.

[186] Salmon-Divon, M., Dvinge, H., Tammoja, K. & Bertone, P. PeakAnalyzer: genome-wide annotation of chromatin binding and modification loci. *BMC Bioinformatics* **11**, 415 (2010). URL http://dx.doi.org/10.1186/1471-2105-11-415.

[187] Gordân, R., Hartemink, A. J. & Bulyk, M. L. Distinguishing direct versus indirect transcription factor-DNA interactions. *Genome Research* **19**, 2090–2100 (2009). URL http://dx.doi.org/10.1101/gr.094144.109.

[188] Mariani, L., Weinand, K., Vedenko, A., Barrera, L. A. & Bulyk, M. L. Identification of human lineage-specific transcriptional coregulators enabled by a glossary of binding modules and tunable genomic backgrounds. *Cell systems* **5**, 187–201.e7 (2017). URL http://dx.doi.org/10.1016/j.cels.2017.06.015.

[189] Mullin, N. P. *et al.* Distinct contributions of tryptophan residues within the dimerization domain to nanog function. *Journal of Molecular Biology* **429**, 1544–1553 (2017). URL http://dx.doi.org/10.1016/j.jmb.2016.12.001.

[190] Botquin, V. *et al.* New POU dimer configuration mediates antagonistic control of an osteopontin preimplantation enhancer by oct-4 and sox-2. *Genes & Development* **12**, 2073–2090 (1998). URL http://dx.doi.org/10.1101/gad.12.13.2073.

[191] Tomilin, A. *et al.* Synergism with the coactivator OBF-1 (OCA-b, BOB-1) is mediated by a specific POU dimer configuration. *Cell* **103**, 853–864 (2000). URL http://dx.doi.org/10.1016/S0092-8674(00)00189-6.

[192] Mistri, T. K. *et al.* Selective influence of sox2 on POU transcription factor binding in embryonic and neural stem cells. *EMBO Reports* **16**, 1177–1191 (2015). URL http://dx.doi.org/10.15252/embr.201540467.

[193] Arimbasseri, A. G. & Maraia, R. J. RNA polymerase III advances: Structural and tRNA functional views. *Trends in Biochemical Sciences* **41**, 546–559 (2016). URL http://dx.doi.org/10.1016/j.tibs.2016.03.003.

[194] Smit, A., Hubley, R. & Green, P. Repeatmasker open-4.0. 2013–2015 (2015).

[195] Kunarso, G. *et al.* Transposable elements have rewired the core regulatory network of human embryonic stem cells. *Nature genetics* **42**, 631 (2010).

[196] Zhu, F. *et al.* The interaction landscape between transcription factors and the nucleosome. *Nature* **562**, 76–81 (2018). URL http://www.nature.com/articles/s41586-018-0549-5.

[197] Li, Q. & Wrange, O. Accessibility of a glucocorticoid response element in a nucleosome depends on its rotational positioning. *Molecular and Cellular Biology* **15**, 4375–4384 (1995). URL http://dx.doi.org/10.1128/mcb.15.8.4375.

[198] Bell, R. J. A. *et al.* The transcription factor gabp selectively binds and activates the mutant tert promoter in cancer. *Science* **348**, 1036–1039 (2015). URL http://dx.doi.org/10.1126/science.aab0015.

[199] Sharon, E. *et al.* Inferring gene regulatory logic from high-throughput measurements of thousands of systematically designed promoters. *Nature Biotechnology* **30**, 521–530 (2012). URL http://dx.doi.org/10.1038/nbt.2205.

[200] Cai, H. N., Arnosti, D. N. & Levine, M. Long-range repression in the drosophila embryo. *Proceedings of the National Academy of Sciences* **93**, 9309–9314 (1996). URL http://dx.doi.org/10.1073/pnas.93.18.9309.

[201] Thanos, D. & Maniatis, T. Virus induction of human ifnβ gene expression requires the assembly of an enhanceosome. *Cell* **83**, 1091–1100 (1995). URL http://dx.doi.org/10.1016/0092-8674(95)90136-1.

[202] Müller, J., Oehler, S. & Müller-Hill, B. Repression of lac promoter as a function of distance, phase and quality of an auxiliary lac operator. *Journal of Molecular Biology* **257**, 21–29 (1996). URL http://dx.doi.org/10.1006/jmbi.1996.0143.

[203] Lee, D., Karchin, R. & Beer, M. A. Discriminative prediction of mammalian enhancers from DNA sequence. *Genome Res.* **21**, 2167–2180 (2011).

[204] Morgunova, E. & Taipale, J. Structural perspective of cooperative transcription factor binding. *Current opinion in structural biology* **47**, 1–8 (2017).

[205] Xie, L. *et al.* A dynamic interplay of enhancer elements regulates klf4 expression in naïve pluripotency. *Genes & Development* **31**, 1795–1808 (2017). URL `http://dx.doi.org/10.1101/gad.303321.117`.

[206] Poplin, R. *et al.* A universal SNP and small-indel variant caller using deep neural networks. *Nat. Biotechnol.* **36**, 983–987 (2018).

[207] Luo, R., Sedlazeck, F. J., Lam, T.-W. & Schatz, M. C. A multi-task convolutional deep neural network for variant calling in single molecule sequencing. *Nature communications* **10**, 998 (2019).

[208] Chuai, G. *et al.* DeepCRISPR: optimized CRISPR guide RNA design by deep learning. *Genome Biol.* **19**, 80 (2018).

[209] Kim, H. K. *et al.* Deep learning improves prediction of CRISPR-Cpf1 guide RNA activity. *Nat. Biotechnol.* **36**, 239–241 (2018).

[210] Gentleman, R. C. *et al.* Bioconductor: open software development for computational biology and bioinformatics. *Genome Biol.* **5**, R80 (2004).

[211] Wilkinson, M. D. *et al.* The fair guiding principles for scientific data management and stewardship. *Scientific data* **3** (2016).

[212] Grüning, B. *et al.* Bioconda: sustainable and comprehensive software distribution for the life sciences. *Nat. Methods* **15**, 475–476 (2018).

[213] When it comes to reproducible science, git is code for success. `https://www.natureindex.com/news-blog/when-it-comes-to-reproducible-science-git-is-code-for-success`. Accessed: 2018-6-28.

[214] Kurtzer, G. M., Sochat, V. & Bauer, M. W. Singularity: Scientific containers for mobility of compute. *PLoS One* **12**, e0177459 (2017).

[215] Sochat, V. V., Prybol, C. J. & Kurtzer, G. M. Enhancing reproducibility in scientific computing: Metrics and registry for singularity containers. *PLoS One* **12**, e0188511 (2017).

[216] Köster, J. & Rahmann, S. Snakemake—a scalable bioinformatics workflow engine. *Bioinformatics* **28**, 2520–2522 (2012).

[217] Kornblith, S., Shlens, J. & Le, Q. V. Do better ImageNet models transfer better? (2018). `1805.08974`.

[218] Esteva, A. *et al.* Dermatologist-level classification of skin cancer with deep neural networks. *Nature* **542**, 115–118 (2017).

[219] Pawlowski, N., Caicedo, J. C., Singh, S., Carpenter, A. E. & Storkey, A. Automating morphological profiling with generic deep convolutional networks (2016).

[220] Zhang, W. *et al.* Deep model based transfer and Multi-Task learning for biological image analysis. *IEEE Transactions on Big Data* 1–1 (2018).

[221] Zeng, T., Li, R., Mukkamala, R., Ye, J. & Ji, S. Deep convolutional neural networks for annotating gene expression patterns in the mouse brain. *BMC Bioinformatics* **16**, 147 (2015).

[222] Simonyan, K., Vedaldi, A. & Zisserman, A. Deep inside convolutional networks: Visualising image classification models and saliency maps (2013). `1312.6034`.

[223] Landrum, M. J. *et al.* Clinvar: public archive of interpretations of clinically relevant variants. *Nucleic acids research* **44**, D862–D868 (2015).

[224] López-Bigas, N., Audit, B., Ouzounis, C., Parra, G. & Guigó, R. Are splicing mutations the most frequent cause of hereditary disease? *FEBS letters* **579**, 1900–1903 (2005).

[225] Liu, X., Wu, C., Li, C. & Boerwinkle, E. dbNSFP v3.0: A One-Stop database of functional predictions and annotations for human nonsynonymous and Splice-Site SNVs. *Hum. Mutat.* **37**, 235–241 (2016).

[226] Xiong, H. Y. *et al.* The human splicing code reveals new insights into the genetic determinants of disease. *Science* **347** (2015). URL `http://science.sciencemag.org/content/347/6218/1254806/tab-pdf`.

[227] Soufi, A., Donahue, G. & Zaret, K. S. Facilitators and impediments of the pluripotency reprogramming factors' initial engagement with the genome. *Cell* **151**, 994–1004 (2012). URL `http://dx.doi.org/10.1016/j.cell.2012.09.045`.

[228] Vaswani, A. *et al.* Attention is all you need. In *Advances in neural information processing systems*, 5998–6008 (2017).

[229] Zoph, B. & Le, Q. V. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578* (2016).

[230] Elsken, T., Metzen, J. H. & Hutter, F. Neural architecture search: A survey. *arXiv preprint arXiv:1808.05377* (2018).

[231] Wang, J., Levasseur, D. N. & Orkin, S. H. Requirement of nanog dimerization for stem cell self-renewal and pluripotency. *Proceedings of the National Academy of Sciences of the United States of America* **105**, 6326–6331 (2008). URL `http://dx.doi.org/10.1073/pnas.0802288105`.

[232] Hayashi, Y. *et al.* Structure-based discovery of NANOG variant with enhanced properties to promote self-renewal and reprogramming of pluripotent stem cells.

Proceedings of the National Academy of Sciences of the United States of America **112**, 4666–4671 (2015). URL `http://dx.doi.org/10.1073/pnas.1502855112`.

[233] James Faresse, N. *et al.* Genomic study of RNA polymerase II and III SNAPc-bound promoters reveals a gene transcribed by both enzymes and a broad use of common activators. *PLoS Genetics* **8**, e1003028 (2012). URL `http://dx.doi.org/10.1371/journal.pgen.1003028`.

[234] Wong, R. C.-B. *et al.* A novel role for an RNA polymerase III subunit POLR3G in regulating pluripotency in human embryonic stem cells. *Stem Cells* **29**, 1517–1527 (2011). URL `http://dx.doi.org/10.1002/stem.714`.

[235] McLaren, W. *et al.* The ensembl variant effect predictor. *Genome Biol.* **17**, 122 (2016).