UNIVERSITY OF CALIFORNIA,
IRVINE

Efficient large-scale machine learning algorithms for genomic sequences

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Science

by

Daniel Quang

Dissertation Committee:
Professor Xiaohui Xie, Chair
Professor Pierre Baldi
Professor Ali Mortazavi

2017

DEDICATION

To everyone who believed in me.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor, Xiaohui Xie. Your guidance, expertise, and overwhelming optimism have been instrumental to my graduate career. Next, thank you to my supportive parents. They instilled in me a strong work ethic and provided everything I needed to be a successful scientist. Finally, I want to thank all the UCI friends I made as a graduate student: Aaron Soffa, Alex Chao, Carrie Wang, Emma Mah, Karina Ramirez, Scott Lee, Shirley Zhang, Timothy Johnson, Zachary Butler, and Zachary Destefano. Your friendship means the world to me.

I came into UCI through the Mathematical and Computational Biology Gateway program. I owe everyone involved in the program a tremendous amount of gratitude. Without this program, I would not be where I am today. Thank you.

This work was supported by the National Institute of Biomedical Imaging and Bioengineering, National Research Service Award (EB009418) from the University of California, Irvine, Center for Complex Biological Systems and the National Science Foundation Graduate Research Fellowship under Grant No. (DGE-1321846). Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

CURRICULUM VITAE

Daniel Quang

EDUCATION

Doctor of Philosophy in Computer Science **2017**
University of California, Irvine *Irvine, California*

Bachelor of Engineering in Chemical Engineering **2012**
The Cooper Union for the Advancement of Science and Art *New York, New York*

RESEARCH EXPERIENCE

Graduate Research Assistant **2012–2017**
University of California, Irvine *Irvine, California*

TEACHING EXPERIENCE

Reader **2014**
University of California, Irvine *Irvine, California*

REFEREED JOURNAL PUBLICATIONS

EXTREME: an online EM algorithm for motif-discovery. **2014**
Bioinformatics

DANN: a deep learning approach for annotating the pathogenicity of genetic variants. **2015**
Bioinformatics

Motif signatures in stretch enhancers are enriched for disease-associated genetic variants. **2015**
Epigenetics and chromatin

DanQ: a hybrid convolutional and recurrent deep neural network for quantifying the function of DNA sequences. **2016**
Nucleic acids research

SOFTWARE

EXTREME `https://github.com/uci-cbcl/EXTREME`
Online EM algorithm for motif discovery.

DANN `https://cbcl.ics.uci.edu/public_data/DANN/`
Multilayer perceptron for functionally annotating both coding and non-coding genetic variants.

DanQ `https://github.com/uci-cbcl/DanQ`
Stacked convolutional and recurrent neural network for predicting chromatin markers from raw sequence data.

FactorNet `https://github.com/uci-cbcl/FactorNet`
Extension of the DanQ framework for predicting transcription factor binding across cell types.

ABSTRACT OF THE DISSERTATION

Efficient large-scale machine learning algorithms for genomic sequences

By

Daniel Quang

Doctor of Philosophy in Computer Science

University of California, Irvine, 2017

Professor Xiaohui Xie, Chair

High-throughput sequencing (HTS) has led to many breakthroughs in basic and translational biology research. With this technology, researchers can interrogate whole genomes at single-nucleotide resolution. The large volume of data generated by HTS experiments necessitates the development of novel algorithms that can efficiently process these data. At the advent of HTS, several rudimentary methods were proposed. Often, these methods applied compromising strategies such as discarding a majority of the data or reducing the complexity of the models. This thesis focuses on the development of machine learning methods for efficiently capturing complex patterns from high volumes of HTS data.

First, we focus on on *de novo* motif discovery, a popular sequence analysis method that predates HTS. Given multiple input sequences, the goal of motif discovery is to identify one or more candidate motifs, which are biopolymer sequence patterns that are conjectured to have biological significance. In the context of transcription factor (TF) binding, motifs may represent the sequence binding preference of proteins. Traditional motif discovery algorithms do not scale well with the number of input sequences, which can make motif discovery intractable for the volume of data generated by HTS experiments. One common solution is to only perform motif discovery on a small fraction of the sequences. Scalable algorithms that simplify the motif models are popular alternatives. Our approach is a stochastic method

that is scalable and retains the modeling power of past methods.

Second, we leverage deep learning methods to annotate the pathogenicity of genetic variants. Deep learning is a class of machine learning algorithms concerned with deep neural networks (DNNs). DNNs use a cascade of layers of nonlinear processing units for feature extraction and transformation. Each layer uses the output from the previous layer as its input. Similar to our novel motif discovery algorithm, artificial neural networks can be efficiently trained in a stochastic manner. Using a large labeled dataset comprised of tens of millions of pathogenic and benign genetic variants, we trained a deep neural network to discriminate between the two categories. Previous methods either focused only on variants lying in protein coding regions, which cover less than 2% of the human genome, or applied simpler models such as linear support vector machines, which can not usually capture non-linear patterns like deep neural networks can.

Finally, we discuss convolutional (CNN) and recurrent (RNN) neural networks, variations of DNNs that are especially well-suited for studying sequential data. Specifically, we stacked a bidirectional recurrent layer on top of a convolutional layer to form a hybrid model. The model accepts raw DNA sequences as inputs and predicts chromatin markers, including histone modifications, open chromatin, and transcription factor binding. In this specific application, the convolutional kernels are analogous to motifs, hence the model learning is essentially also performing motif discovery. Compared to a pure convolutional model, the hybrid model requires fewer free parameters to achieve superior performance. We conjecture that the recurrent layer allows our model spatial and orientation dependencies among motifs better than a pure convolutional model can. With some modifications to this framework, the model can accept cell type-specific features, such as gene expression and open chromatin DNase I cleavage, to accurately predict transcription factor binding across cell types. We submitted our model to the ENCODE-DREAM *in vivo* Transcription Factor Binding Site Prediction Challenge, where it was among the top performing models. We implemented

several novel heuristics, which significantly reduced the training time and the computational overhead. These heuristics were instrumental to meet the Challenge deadlines and to make the method more accessible for the research community.

HTS has already transformed the landscape of basic and translational research, proving itself as a mainstay of modern biological research. As more data are generated and new assays are developed, there will be an increasing need for computational methods to integrate the data to yield new biological insights. We have only begun to scratch the surface of discovering what is possible from both an experimental and a computational perspective. Thus, further development of versatile and efficient statistical models is crucial to maintaining the momentum for new biological discoveries.

Chapter 1

Introduction

Epigenetics refers to modifications to the genome that do not affect the underlying nucleotide sequence. These modifications are important for the control of gene regulation and establishing cell identity. Examples of epigenetics include the 3D chromosome conformation, DNA methylation, and histone modifications. Unlike the underlying genome, which is largely static within an individual, the epigenome, which includes all epigenetic changes at a genome-wide level, can be dynamically altered by environmental conditions.

HTS has provided an unprecedented opportunity to query the epigenome. Since the introduction of HTS, many experimental assays that utilize this technology have been developed. Although HTS assays are designed for different purposes, they typically adhere to the following multi-step scheme:

1. **Collect a homogeneous bulk of cells**. Although the exact number depends on the assay and application, HTS assays require a large number of cells, reaching as many as tens of millions of cells in some cases. For this reason, immortalized cell lines are widely used because they are convenient for growing a large number of cells. One common criticism of cell lines is that they are derived from cancerous or transformed

samples, and therefore not representative of endogenous biology. Primary tissues are one alternative, but they can be difficult to collect and they are not as uniform or reproducible as cell lines.

2. **Fragmentation and enrichment**. The order of these two steps depends on the specific assay. Fragmentation shears the nucleotide sequences into smaller strands. This can be achieved by various means, such as nucleases and sonication. Enrichment positively selects for fragments of interest, while filtering away irrelevant fragments.

3. **Amplification**. The enriched short sequences are duplicated in a procedure called polymerase chain reaction so that enough identical DNA strands are available to analyze.

4. **Sequencing**. In this step, letter nucleotide sequences are obtained from the cloned DNA fragments. Each nucleotide sequence is called a "read." Sequencing is highly parallelized.

5. **Read mapping**. Reads from the sequencing step are computationally mapped back to a reference genome. The overlap, or pileup, of reads form a histogram of read depth along the genome. This histogram is a genomic signal that is essentially a discrete-time signal.

By no means a complete list, examples of HTS assays include:

1. **Whole genome sequencing (WGS)**. Perhaps one of the most basic HTS assays, but also one of the most expensive, WGS is the process of determining the complete DNA sequence of an individual's genome. Putative genetic variants, such as single nucleotide polymorphisms (SNPs), insertions, and deletions, are identified through differences between aligned read sequences and the reference genome. Variants may carry information about an individual's susceptibility to certain diseases. In order to

get a statistically reliable assessment of individual putative variants, samples must be sequenced at sufficient depths, which significantly increases cost. In some sense, WGS may be better characterized as whole genome resequencing, since reconstructing the whole genome is a considerably more difficult task.

2. **ChIP-seq [48]**. ChIP-seq is an assay that combines chromatin immunoprecipitation and HTS to characterize binding interactions between proteins and DNA *in vivo*. Binding sites appear as "peaks" in the genomic signal. ChIP-seq can also identify genomic regions of histone modifications. These histone modifications are thought to demarcate regulatory elements.

3. **DNase-seq [27, 47]**. DNase-seq profiles accessible chromatin based on the genome-wide sequencing of regions sensitive to cleavage by DNase I. FAIRE-seq [37] and ATAC-seq [19] are also HTS assays that can profile accessible chromatin, although we do not focus on these assays in this thesis. There are also two different variations of the DNase-seq protocol: a single-cut protocol [27] and a double-cut protocol [47]. The latter is much more prominent in the literature and the one we focus on in this thesis. At deep enough sequencing depth, protein binding sites appear as "footprints" (FPs) in the genomic signal [43, 70, 17].

4. **RNA-seq [68]** RNA-seq uses HTS to reveal the presence and quantity of RNA *in vivo*. RNA fragments are converted to DNA fragments via reverse transcription before the sequencing step.

Advances in HTS research has been encouraged by the ever-decreasing cost of sequencing. To put this into perspective: the estimated cost for generating the initial "draft" human genome sequence completed in June 2000 is $300 million worldwide; by 2006, the cost to generate a high-quality 'draft' human genome sequence had dropped to $14 million; by late 2015, that figure had fallen below $1,500 [1]. As a result of the robustness and decreasing costs

of HTS, many large data consortia have emerged. Examples include the Encyclopedia of DNA Elements (ENCODE) [29], the Roadmap Epigenomics Project [84], Genotype-Tissue Expression Project (GTEx) [62], the 1000 Genomes Project [2], and the Cancer Genome Atlas (TCGA) [65]. Although these consortia have different but related goals, they follow a similar premise: collect -omic scale data on a variety of individuals, cell types, and conditions and disseminate the data to the public. With such a large volume of data entering the public domain, there is a growing need for novel computational methods to efficiently analyze and integrate these diverse datasets. One common application of integrating these datasets is for the purpose of identifying regulatory elements such as enhancers, insulators, and promoters. Some algorithms accomplish this by dividing the entire genome systematically into segments, and then assigning the resulting genome segments into "chromatin states" by applying machine learning methods such as Hidden Markov Models, Dynamic Bayesian Networks, or Self-Organizing Maps [30, 46, 67].

In this thesis, we introduce novel computational methods to solve a few of the myriad of long-standing problems in computational genomics. Our methods place a strong emphasis on scalability and non-linearity.

1.1 Dissertation Outline and Contributions

The thesis is outlined as follows:

In chapter 2, we present our contributions to *de novo* motif discovery and the resulting improvement in several large sequence datasets. The contributions include an update to the popular motif discovery algorithm, MEME [9], by replacing its batch Expectation-Maximization (EM) algorithm [28] with an online EM algorithm [21]. The main difference between these two EM algorithms is that the batch version performs a parameter update

each time after processing the whole training dataset, whereas the online version performs a parameter update after processing a single training sample, or a mini-batch of training samples. The online EM algorithm also converges considerably faster than the batch EM version. If required, the online EM algorithm can be combined with a streaming strategy so that data are efficiently loaded as needed from the hard drive into system memory, which in turn considerably reduces the memory resources required to run the algorithm. Compared to MEME, our method is several orders of magnitude faster, and this gap quickly widens as a function of the size of the dataset. These contributions are released as an open-source software package called EXTREME, which is available at https://github.com/uci-cbcl/EXTREME. Portions of this chapter were published as part of [76].

In chapter 3, we present our contributions to pathogenic variant annotation. Pathogenic variant annotation is the task of identifying genetic variants that cause disease. This is often done by assigning a score to a variant that describes its propensity to cause a disease. We developed a deep learning approach that trains a DNN on a large set of tens of millions of labeled genetic variants, treating the problem as a binary classification task. Similar to the online EM algorithm used in our EXTREME algorithm, the stochastic gradient descent method used to train DNNs efficiently performs parameter updates on small mini-batches of training data. Our method has the added benefit of being able to assign scores to variants, which many prior annotation methods tended to avoid. It is also an improvement on Combined AnnotationDependent Depletion (CADD) [55], which uses a linear support vector machine (SVM) [26] to complete the same task. Linear SVMs cannot capture non-linear interactions among features, which limits its performance relative to DNNs. These contributions are released as an open-source software package called DANN and a genome-wide set of precomputed SNP scores, which are available at https://cbcl.ics.uci.edu/public_data/DANN/. Portions of this chapter were published as part of [75].

In chapter 4, we present another deep learning method for variant annotation. The imple-

mentation is based around CNNs and RNNs using raw DNA sequences as inputs to predict cell type-specific epigenetic markers. To combine the strengths of both types of DNNs, we stacked a recurrent layer on top of a convolutional layer to form a hybrid architecture. The kernels of the convolutional layer are analogous to motifs. Hence, the model training is also implicitly performing motif discovery, similar to our EXTREME algorithm. Our method is an improvement on DeepSEA [105], which uses a pure convolutional model with more free weights and layers than our hybrid model does. The source code of the method is released as an open-source software package called DanQ, which is available at https://github.com/uci-cbcl/DanQ. Portions of this chapter were published as part of [77].

Finally, in chapter 5, we introduce a method for predicting TF binding across cell types. Our proposed solution extends the hybrid architecture of the DanQ framework. The model is trained on reference binding data and uses cell-type specific features, such as DNase I hypersensitivity and gene expression, to generalize from training cell types to testing cell types. We submitted our model ENCODE-DREAM *in vivo* Transcription Factor Binding Site Prediction Challenge, where it was among the top performing models. These contributions are released as an open-source software package called FactorNet, available at https://github.com/uci-cbcl/FactorNEt.

Chapter 2

EXTREME: motif discovery by online Expectation-Maximization

2.1 Introduction

TFs are proteins that play an important role in transcriptional regulation by promoting or blocking the recruitment of RNA polymerase II. They can bind specifically to recognition sequences on the genome or to other TFs in a complex. HTS assays generate a rich amount of information on the sequence preference of TFs. Peak sequences in ChIP-Seq [48] can provide the genome-wide binding sites of a single TF. DNase-Seq, which sequences open chromatin regions in the genome, can provide single nucleotide resolution for the binding sites of many TFs [43, 70]. When sequenced deep enough, binding sites appear as dips, or footprints (FPs), in the DNase-Seq signal. FPs only identify the locations of the TF binding sites; they do not identify the proteins that are bound there. These assays can provide functional information for thousands to millions of bp regions in the genome.

The task of identifying the sequence preference of a TF is called motif discovery. Motif

discovery algorithms can be classified as either search-based or probabilistic. Search-based algorithms infer a motif as a regular expression or consensus sequences. Probabilistic algorithms infer a motif as a position weight matrix (PWM). PWMs were first introduced as alternatives to consensus sequences [95]. In the first step of constructing a PWM, a basic position frequency matrix (PFM) is constructed by counting the occurrences of each nucleotide at each position. This step is usually performed after a sequence alignment. From the PFM, a position probability matrix (PPM) can now be constructed by dividing nucleotide counts at each position by the number of sequences, thereby normalizing the values. Finally, a PWM is constructed from a PPM by computing the log ratio of nucleotide probabilities with respect to a background distribution. Both PPMs and PWMs assume statistical independence between positions in the pattern, as the probabilities for each position are calculated independently of other positions. Each column can therefore be regarded as an independent categorical distribution. Other, more complex, motif models such as the Slim model [49] relax this assumption, but PWMs nevertheless remain ubiquitous in bioinformatics community due to their simplicity and ease of interpretability.

While PWMs provide more information about a TF's binding specificity than consensus sequences, inferring PWMs is not always practical. Probabilistic motif discovery programs usually employ algorithms such as expectation-maximization (EM) [28] for inference. These algorithms scale poorly with dataset size. Search-based algorithms are therefore preferred for large datasets. DREME [5] is an example of a search-based algorithm designed for large datasets.

MEME is a popular probabilistic motif discovery program [9]. It uses the EM algorithm to infer PWMs. Since its inception in 1994, it has gone through several versions. However, MEME scales poorly with large datasets. One strategy to improve MEME's performance is to discard many of the sequences. This is the strategy used by MEME-ChIP [63], which accepts 500 sequences. In comparison, a ChIP-seq dataset may contain tens of thousands of binding

peak sequences, and a DNase-seq dataset may contain approximately one million footprint sequences. However, discarding sequences can decrease the chance of discovering motifs corresponding to infrequent cofactors. Another strategy, as utilized in STEME, applies suffix trees to accelerate MEME [81]. However, STEME is only practical for finding motifs of up to width 8 on large datasets because its efficiency tails off quickly as the motif width increases. Other strategies for accelerating MEME involve specialized hardware such as parallel pattern matching chips on PCI cards [89] or graphical processing units (GPUs) [22]. However, these implementations require hardware not available to most researchers.

To overcome these issues, we propose an online implementation of the MEME algorithm that we have named EXTREME. The online EM algorithm adheres closely to the original EM algorithm (hereafter referred to as the batch EM algorithm) [21], and it has achieved remarkable success in other large data bioinformatics applications such as transcript quantitation [85] and deconvolving subclonal tumor populations [60]. Normally, the online EM algorithm is designed for cases where not all data can be stored at once. Although most computers have enough memory to store entire sequence datasets at once, the online EM algorithm is still advantageous for motif discovery because, for large sample sizes, the online EM algorithm converges faster than the batch EM algorithm. We show that many of the features of the original MEME algorithm can be adapted to the online methodology. Furthermore, we show that EXTREME can achieve similar results to MEME in a fraction of the execution time. We also show that using the entire dataset is necessary to discover infrequent motifs, which is not always practical to do with MEME. To the best of our knowledge, this is the first application of the online EM algorithm to motif discovery.

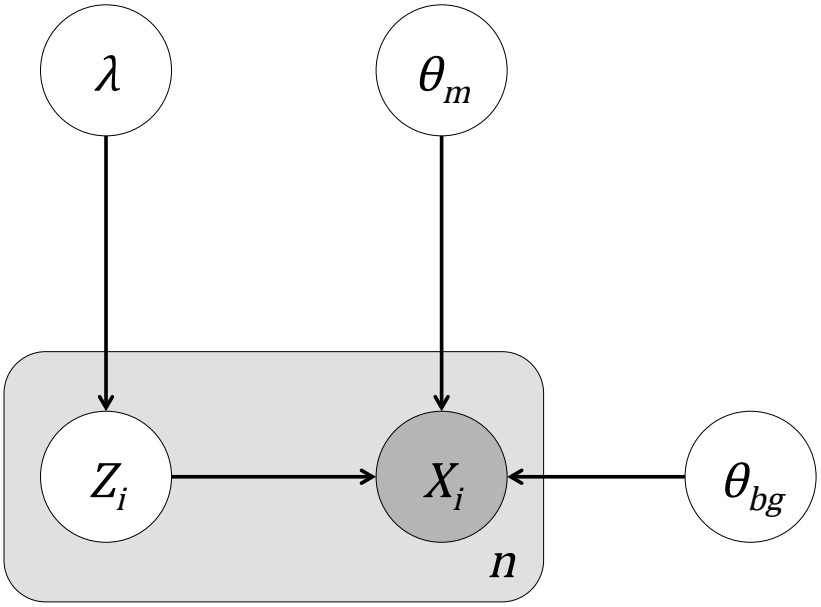


Figure 2.1: MEME's model: λ_m, the prior probability of a binding site; Z_i, the binary latent variable representing whether the i-th W-mer is a motif instance; X_i, the i-th W-mer; θ_m, the PPM of the motif; θ_{bg}, the nucleotide frequency vector of the background distribution.

2.2 Methods

2.2.1 MEME

The original MEME algorithm applies the batch EM algorithm to infer PPMs. Here, we provide a brief overview of MEME's model and how MEME applies the batch EM algorithm to infer parameters.

MEME's model

Let $Y = (Y_1, Y_2, \ldots, Y_N)$ be the dataset of sequences, where N is the number of sequences in the dataset. Each sequence is over the alphabet $\mathcal{A} = (A, C, G, T)$. MEME uses a mixture model that breaks up the dataset into all n (overlapping) subsequences of length W and treats each subsequence as independent and identically distributed (i.i.d.) observations. We will

refer to this new dataset as $X = (X_1, X_2, \ldots, X_n)$. The mixture model is a two-component model that assumes each subsequence is either an instance of the motif or background. Other variants of MEME place additional constraints. The one occurrence per sequences (OOPS) variant assumes that each sequence contains one instance of the motif. The zero or one occurrence per sequence (ZOOPS) variant assumes each sequence can have zero or only one occurrence of the motif. These two variants make slight modifications to MEME's probabilistic model. We will only consider the two-component model.

The background component is characterized as a zero-order Markov model parameterized by the vector $\theta_{bg} = (f_{0,A}, f_{0,C}, f_{0,G}, f_{0,T})$ where $f_{0,k}$ is the background frequency of letter k. The background model can be viewed as a PPM with W identical columns. The motif model is characterized by the PPM $\theta_m = (f_1, f_2, \ldots, f_W)$. Each $f_j = (f_{j,A}, f_{j,C}, f_{j,G}, f_{j,T})$ is a parameter of an independent random variable describing a multinomial trial representing the distribution of letters at position j in the motif. The corresponding PWM, $\theta_{pwm} = (g_1, g_2, \ldots, g_W)$, can be computed from θ_{bg} and θ_m as follows:

$$g_{j,k} = \log_2 \frac{f_{j,k}}{f_{0,k}} \tag{2.1}$$

λ_m parameterizes the probability that any W-mer is generated by the motif model while $\lambda_{bg} = 1 - \lambda_m$ is the probability that any W-mer is generated by the background model. $\theta = (\theta_m, \theta_{bg})$ and $\lambda = (\lambda_m, \lambda_{bg})$ are unknown parameters that are inferred from the known data X. Therefore, the MEME model is

$$p(Z_i = 1|\theta, \lambda) = \lambda_m, 1 \leq i \leq n \tag{2.2}$$

$$p(X_i|Z_i,\theta) = p(X_i|\theta_m)^{Z_i} p(X_i|\theta_{bg})^{1-Z_i} \tag{2.3}$$

where Z_i is a binary latent variable which has a value of 1 if X_i is drawn from the motif model or 0 if X_i is drawn from the background model. Z_i's true value is unknown, but its conditional expected value, defined here as $Z_i^{(0)}$, for a given set of parameters can be calculated as follows:

$$Z_i^{(0)} = E[Z_i|X,\theta,\lambda] = \frac{p(X_i|\theta_m)\lambda_m}{p(X_i|\theta_m)\lambda_m + p(X_i|\theta_{bg})\lambda_{bg}} \tag{2.4}$$

To calculate $Z_i^{(0)}$, we need to know the form of $p(X_i|\theta_m)$ and the form of $p(X_i|\theta_{bg})$. MEME assumes the distributions of the motif class and background class are

$$p(X_i|\theta_m) = \prod_{j=1}^{W} \prod_{k\in\mathcal{A}} f_{j,k}^{I(k,X_{i,j})} \tag{2.5}$$

$$p(X_i|\theta_{bg}) = \prod_{j=1}^{W} \prod_{k\in\mathcal{A}} f_{0,k}^{I(k,X_{i,j})} \tag{2.6}$$

where $X_{i,j}$ is the letter in the jth position of subsequence X_i, and $I(k,a)$ is an indicator function

$$I(k,a) = \begin{cases} 1 & if\ a = k \\ 0 & otherwise \end{cases} \tag{2.7}$$

The MEME model is depicted in plate notation in Fig. 2.1.

Batch EM

λ and θ are iteratively improved in the batch EM algorithm. In the E-step, the expected counts of all nucleotides at each position are calculated based on the current guess of the parameters. In the M-step, the parameters are updated based on the values calculated in the E-step. MEME repeats the E and M steps until the change in θ_m (Euclidean distance) falls below a threshold (default: 10^{-6}). The E and M steps are as follows:

E-step:

$$c_{j,k} := \sum_{i=1}^{n} E_i Z_i^{(0)} I(k, X_{i,j}) \quad (1)$$

$$c_{0,k} := \sum_{i=1}^{n} \sum_{j=1}^{W} \left(1 - Z_i^{(0)}\right) I(k, X_{i,j}) \quad (2)$$

M-step:

$$f_{j,k} := \frac{c_{j,k}+\beta_k}{\sum_{k \in \mathcal{A}} (c_{j,k}+\beta_k)} \quad (1)$$

$$\lambda_m := \sum_{i=1}^{n} \frac{Z_i^{(0)}}{n} \quad (2)$$

$$\text{for } k \in \mathcal{A} \text{ and } j = 0, 1, 2, \ldots, W$$

To discover multiple motifs, MEME associates an "erasing factor" E_i for each subsequence X_i. The erasing factors vary between 0 and 1 and are set to 1 initially to indicate no erasing has taken place. Each time a motif is discovered, the erasing factors are reduced by a factor representing the probability that the position overlaps an occurrence of that motif. More details concerning how MEME erases are in [7]. MEME also implements pseudo counts $\beta = (\beta_A, \beta_C, \beta_G, \beta_T)$ in the M-step to prevent any letter frequency $f_{j,k}$ from becoming 0. This is because if any letter frequency $f_{j,k}$ becomes 0, its value cannot change.

EM performs maximum likelihood estimation to maximize an objective function. The new estimates in the M-step are always guaranteed to increase the value of the objective function. As the E and M steps are repeated, EM algorithms converge to a maximum. For MEME,

the objective function is the expected value of the log likelihood of the model parameters θ and λ given the joint distribution of the data X and missing data Z:

$$\begin{aligned} E[\log L(\theta, \lambda | X, Z)] &= \sum_{i=1}^{n} Z_i^{(0)} \log(p(X_i|\theta_m)\lambda_m) \\ &\quad + \sum_{i=1}^{n} \left(1 - Z_i^{(0)}\right) \log(p(X_i|\theta_{bg})\lambda_{bg}) \end{aligned} \tag{2.8}$$

Seeding

The EM algorithm is sensitive to initial conditions and prone to converging to local maxima. To mitigate this problem, MEME tests many seeds and runs the EM algorithm to convergence from the "best" seed. The exact details of how MEME performs seeding can be found in [8].

Scoring the motifs

Motif instances are determined according to Bayesian decision theory. After a motif is discovered, a subsequence X_i is classified as being an occurrence of the motif only if

$$\log\left(\frac{p(X_i|\theta_m)}{p(X_i|\theta_{bg})}\right) > \log\left(\frac{\lambda_{bg}}{\lambda_m}\right) \tag{2.9}$$

For each motif discovered, MEME calculates its E-value. This E-value is the number of motifs, with the same width and number of occurrences, that can generate an equal or higher log likelihood ratio if the dataset had been generated according to background model. The log likelihood ratio $llr = \log(p(sites|motif)/\log(sites|background))$ is a measure of how different the sites are from the background model. Calculating the E-value exactly can be time consuming, so it is not computed directly. It is instead heuristically calculated as a function of the total information content and the number of occurrences [6].

Time complexity

For each iteration of the batch EM algorithm, the number of operations performed is approximately proportional to W. Each batch EM iteration has a time complexity of $O(nW)$. Although the number of iterations can vary, it is typically proportional to n. Therefore, in practice the algorithm scales quadratically with the size of the dataset and has a time complexity of $O(n^2W)$. The seed searching also scales quadratically with the size of the dataset [8].

2.2.2 EXTREME

EXTREME shares many similarities with MEME, especially in the implementation. At the center of the EXTREME algorithm is the online EM algorithm. We provide an overview of the online EM algorithm and how EXTREME implements the online EM algorithm to discover motifs.

Online EM

Like the batch EM algorithm, the online EM algorithm also repeatedly iterates between E and M steps, which update the parameters. In contrast to the batch EM algorithm, each iteration of the online EM algorithm operates on only one observation, X_i, instead of the whole dataset X.

Following the instructions in [21], the E and M steps, as derived from (2.8), are:

The step size is $\gamma_i = \gamma_0 i^{-\alpha}$. α and γ_0 are set to 0.6 and 0.05, respectively. These are by no means the most optimized set of parameters, but they are adequate for accurate motif discovery. As shown in [21], the online EM algorithm converges to a local maximum of the

E-step:

$$s_{m,i} \coloneqq s_{m,i-1} + \gamma_i \left(Z_i^{(0)} - s_{m,i-1} \right) \quad (1)$$

$$c_{j,k,i} \coloneqq c_{j,k,i-1} + \gamma_i \left(Z_i^{(0)} I(k, X_{i,j}) - c_{j,k,i-1} \right) \quad (2)$$

$$c_{0,k,i} \coloneqq c_{0,k,i-1} + \gamma_i \left(\sum_{j=1}^{W} \left(1 - Z_i^{(0)} \right) I(k, X_{i,j}) - c_{0,k,i-1} \right) \quad (3)$$

M-step:

$$f_{j,k} \coloneqq \frac{c_{j,k,i}}{\sum_{k \in \mathcal{A}} c_{j,k,i}} \quad (1)$$

$$\lambda_m \coloneqq s_{m,i} \quad (2)$$

for $k \in \mathcal{A}$, $j = 1, 2, \ldots, W$, and $i = 1, 2, \ldots, n$

likelihood function (2.8) for $\alpha \in (0.5, 1]$.

The E and M steps are repeated until a convergence threshold (default: 10^{-6}) in terms of the symmetrized Kullback-Leibler divergence (KLD) between the PPM estimates at a user-defined number of intervals (default: 100) of W-mers at the end of a complete pass through the dataset is satisfied. The KLD between two PPMs A and B is calculated as follows:

$$KLD(A, B) = \frac{1}{2} \sum_{j=1}^{W} \sum_{k \in \mathcal{A}} \left(A_{j,k} \log \left(\frac{A_{j,k}}{B_{j,k}} \right) + B_{j,k} \log \left(\frac{B_{j,k}}{A_{j,k}} \right) \right) \quad (2.10)$$

If convergence is not reached at the end of a pass, the exponent α is updated to the midpoint between α's current value and one and EXTREME performs another pass through the dataset. EXTREME repeats these steps until the convergence threshold is met.

To accommodate pseudo counts, we modify the indicator function from (2.7):

$$I(k, a) = \begin{cases} 1 + \beta_k & if\ a = k \\ \beta_k & otherwise \end{cases} \quad (2.11)$$

By default, EXTREME sets β_k to 0.0001 times the frequency of letter k in the entire dataset.

To accommodate reverse complements, we also modify the calculation of $Z_i^{(0)}$ from (2.4) so that for each X_i, the reverse complement is also evaluated and $Z_i^{(0)}$ takes the higher of the two values. MEME, in contrast, handles reverse complements by adding a reverse-complemented copy of the data, essentially doubling the size of the data.

Seeding

Before running the online EM algorithm, the order of the W-mers X_i is randomized. The online EM algorithm is therefore a stochastic algorithm. This means that different runs of the online EM algorithm can yield different results, even if ran multiple times from the same initial conditions. This can present a problem for seeding because even using the best seed from MEME's heuristic is not guaranteed to generate the optimal or even consistent solutions, causing EXTREME to converge to local maxima. On the other hand, this also means that seeds that would yield non-optimal solutions in MEME can yield optimal solutions in EXTREME. In fact, local maxima may actually correspond to biologically relevant motifs, especially in datasets that are rich in motifs such as DNase-Seq data. Furthermore, an efficient online EM implementation of MEME offers very little benefit if runtimes are dominated by the inefficient seed search.

EXTREME's seeding strategy applies a search-based motif discovery algorithm to find motifs to initialize the online EM algorithm. Similar to DREME [5], the seeding algorithm finds words that are enriched in a sequence dataset relative to a negative sequence dataset. We use the same dinucleotide shuffle algorithm employed in DREME to generate a dinucleotide-shuffled version of the input sequence set as the negative sequence set. The seeding algorithm counts the number of occurrences of words in the positive sequence set and the negative sequence set and associates a "z-score" with each word. The z-score is given by

$$z = \frac{s_+ - s_-}{\sqrt{s_-}} \tag{2.12}$$

where s_+ and s_- are the number of occurrences of the word in the positive sequence and negative sequence sets, respectively. If s_- is zero for a word, it is changed to one to prevent division by zero. Unlike DREME, our seeding algorithm searches for words that are not exact. Each word contains g universal wildcard letters surrounded by flanking sites of l unambiguous letters. For example, TCAGNNGGAC is a word with a gap length, g, of 2 and a half-length, l, of 4. The gap length, g, varies between the user-defined parameters g_{min} and g_{max}. Z-scores for each value of g are normalized by dividing by the standard deviation of all z-scores for each respective value of g. Words that have a normalized z-score that exceed a user-defined threshold, z_{thresh}, and have at least a user-defined number of occurrences, s_{min}, in the positive sequence set are aligned and grouped together using a hierarchical clustering algorithm we adapted from [104]. Word clusters are converted to frequency count matrices by counting the number of occurrences of each letter at each position along the alignment. The counts are weighted by the normalized z-score of each word in a cluster so that more significant words will contribute more to the count matrix than less significant words. A count matrix, which is also a PFM, is converted to a PPM, θ_m, by dividing each matrix element by its respective row sum. The initial expected counts, c, is initially set to the initial θ_m as well. θ_{bg} and the expected background counts c_0 are set to the nucleotide frequency in the dataset. λ_m and $s_{m,0}$ are initialized to the predicted number of motif occurrences divided by n, the total number of W-mers. We predict the number of motif occurrences for a given PPM seed as the number of W-mers that have a goodness-of-fit score greater than 0.7 (see [71] for details).

We also alter the form of $p(X_i|\theta_m)$ from (2.5):

$$p(X_i|\theta_m) = \psi \prod_{j=1}^{W} \prod_{k \in \mathcal{A}} f_{j,k}^{I(k,X_{i,j})} \tag{2.13}$$

The bias factor ψ has a value between 0 and 1. A bias factor closer to 0 biases the motif discovery towards subsequences that more closely match the current motif guess, decreasing

the number of discovered motif occurrences. A bias factor closer to 1 makes the motif discovery less selective, increasing the number of motif occurrences. After convergence, motif occurrences are identified using (2.9). ψ is initially set to 1, and its value is varied in a binary search fashion until the number of discovered motif occurrences is between $sites_{min}$ (default: 10) and $sites_{max}$ (default: 5 times the number of predicted motif sites). Up to 15 different values of ψ are tried before EXTREME stops. Because each initial PPM guess can be tested independently, this seeding strategy can be parallelized to allow multiple motifs to be discovered simultaneously. Hierarchical clustering of the discovered motifs can then identify individual motif classes.

Time complexity

Each pass through the dataset with the online EM algorithm has a time complexity of $O(nW)$. Typically, the online EM algorithm reaches convergence after one to five passes through the data, so the overall time complexity is proportional to the width of the motif and the size of the dataset. The seeding algorithm's word search also scales linearly with the dataset size, while the hierarchical clustering is inefficient and can scale cubically with the number of words to cluster. In practice, EXTREME as a whole scales linearly in time complexity with the dataset size.

Implementation

EXTREME is written in Python. To calculate E-values, EXTREME uses Cython bindings to the original MEME C source code to call the appropriate functions. EXTREME requires about 8 Gb of memory for a 10 Mbp dataset. Most of the memory is devoted to MEME's E-value calculation, which involves a preprocessing step that does not scale well to large numbers of motif sites. Although we use MEME's E-value in some of the analyses in this

chapter, in practice we found the calculation heuristic to be unreliable and recommend omitting it for most applications.

2.3 Results

MEME is a popular motif discovery algorithm. It has been a valuable tool in the ongoing challenge of identifying regulatory elements. However, its performance scales poorly with large datasets. Experiments such as ChIP-Seq and DNase-Seq generate data that are too large for MEME to process in a practical amount of time without discarding most of the data. To overcome this challenge, we have developed EXTREME, a motif discovery algorithm that can process ChIP-Seq and DNase-Seq data efficiently without discarding any data. We first show, using simulated datasets, that MEME's running time scales much faster than EXTREME's running time with respect to dataset size. Using a ChIP-Seq dataset and a DNase-Seq dataset, we demonstrate that using the entire dataset of sequences is necessary to discover infrequent motifs. We also show that the motifs discovered by EXTREME are similar in quality to the motifs discovered by MEME.

2.3.1 Comparison of MEME and EXTREME performance

We compare MEME and EXTREME using several simulated datasets. Simulated datasets are generated with the RSAT suite of tools [98]. We generate 4 sequence datasets, each containing 1000 random masked hg19 genomic sequences of a single length (100, 200, 300, or 400 bps), using the RSAT *random-genome-fragments* tool. This masked reference genome was preprocessed with RepeatMasker [93] and Tandem Repeats Finder [13] so that repeats (with period of twelve or less) are masked by capital Ns. For each of the 4 sequence datasets, we implant 50, 100, 500, or 1000 instances of the JASPAR [88] VDR/RXRA heterodimer

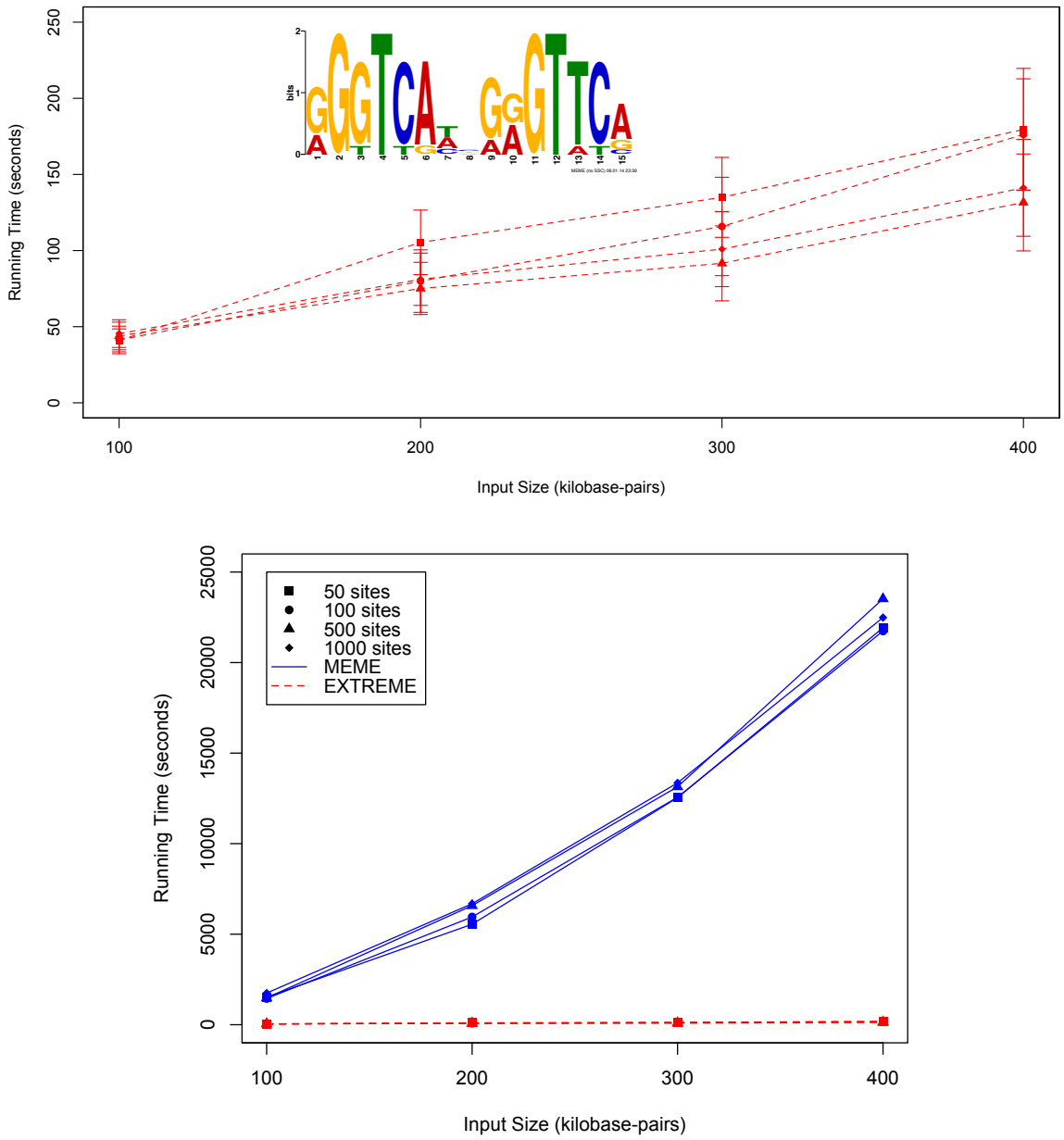


Figure 2.2: **Comparison of MEME and EXTREME running times on simulated datasets of varying sequence length and motif sites.** The x-axis is the total number of bps in the simulated dataset. The y-axis is the total running time it takes for MEME or EXTREME to complete seeding and reach convergence. The sequence logo of the VDR/RXRA heterodimer motif used to generate the simulated datasets is displayed in the top plot.

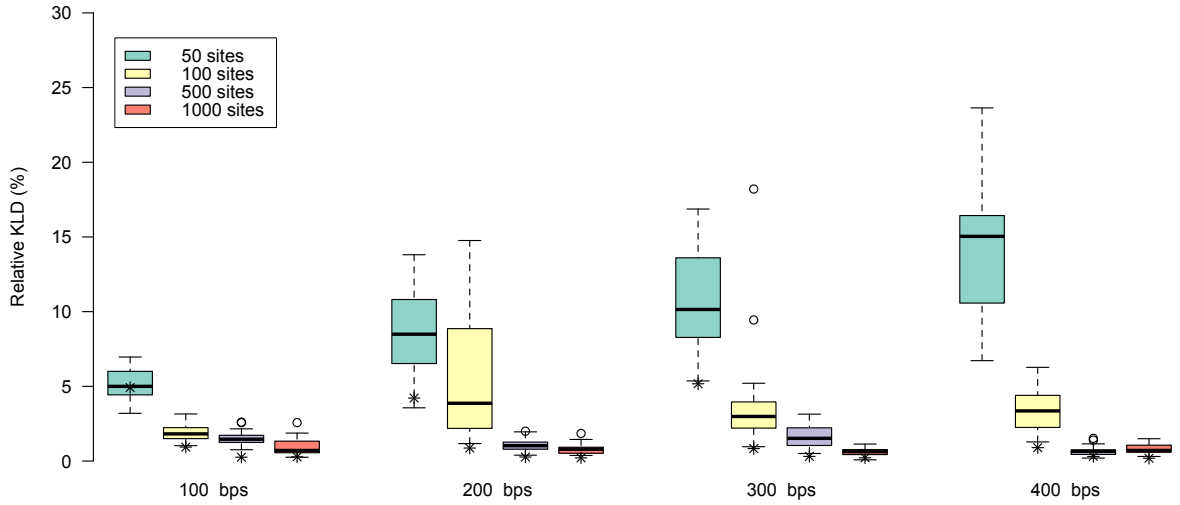


Figure 2.3: **EXTREME captures similar motifs to MEME as the dataset becomes larger and contains more motif occurrences**. Boxplots of the KLD (Eqn. 2.10) between aligned columns of the discovered motifs in the EXTREME runs and the true PPM relative to the KLD between the true PPM and a uniform background model. The 16 datasets are grouped by length of the sequences and colored by the number of implanted motif sites. If available, the KLD between the PPM discovered by MEME for the respective dataset and the true PPM is marked by a *.

motif using the RSAT *random-motifs implant-sites* tools, yielding a total of 16 simulated datasets, each containing 1000 sequences of varying lengths and number of motif sites.

For the seeding step of each EXTREME run, we search for words with a half-length $l = 6$, a gap length g between $g_{min} = 0$ and $g_{max} = 2$, a normalized z-score greater than the threshold $z_{thresh} = 5$, and at least $s_{min} = 5$ occurrences in the positive sequence set. The words are clustered and we select the cluster containing the most words to convert to a PPM seed from which to initialize the online EM algorithm. Because the online EM algorithm is a stochastic algorithm, we repeat the online EM portion of the run 30 times for each dataset with different random seeds to initialize the pseudorandom number generator in order to get a good estimate of performance. We also run MEME on each of the 16 simulated datasets to find a single motif of a width between 12 and 17 under the two-component model to

approximate the same parameters for the EXTREME run. Fig. 2.1 shows that MEME's running time scales much faster than EXTREME's running time with respect to the input size for all noise levels. Extrapolating from these data, MEME can take weeks to discover a motif in a 10 Mbp dataset. EXTREME can complete this same task in hours. With the exception of one of the datasets, MEME is marginally more accurate than EXTREME in each case (Fig. 2.3). In the one exception, MEME fails to converge to the correct motif because there are not enough true motif occurrences relative to the dataset size for MEME's seeding algorithm to pick a good seed. As the number of motif occurrences increases, both MEME and EXTREME better approximate the true PPM and the relative difference between their results diminish. Although EXTREME's running time and accuracy vary more as the noise level increases, EXTREME still consistently generates results comparable to those of MEME in a fraction of MEME's running time.

2.3.2 Discovering motifs in ChIP-Seq data

We compare the performance of MEME and EXTREME for discovering motifs in ChIP-Seq data using a dataset generated by the Myers Lab at the Hudson Alpha Institute for Biotechnology [16]. The ChIP-Seq data correspond to an REST ChIP performed on the GM12878 cell line. Peaks were already called and organized into BED files by the authors. We further process the data by intersecting replicates and extracting genomic sequences from the middle 100 bps of the intersected regions from the same hg19 masked reference genome we use for the simulated data. The resulting sequence dataset consists of 2849 sequences and 282980 bps.

We run EXTREME on the ChIP-Seq dataset to discover multiple motifs. For the seeding step, we search for words with a half-length of 8, a gap length between 0 and 10, inclusive, a normalized z-score greater than 5, and at least 5 occurrences in the positive sequence set.

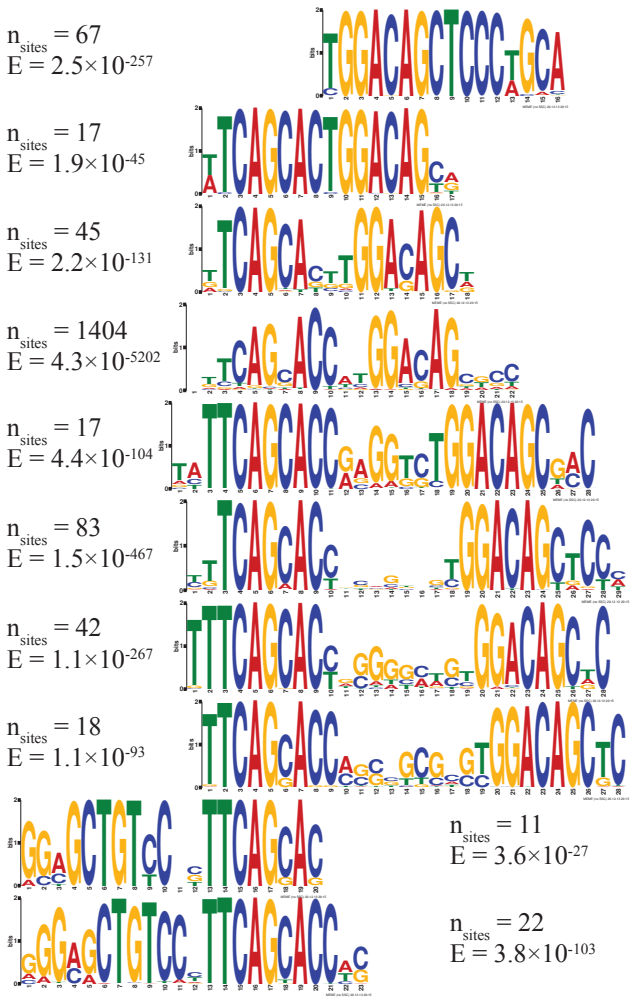


Figure 2.4: Motifs discovered by EXTREME in the GM12878 REST ChIP-Seq dataset. Each motif comes from one of the 10 motif clusters. Motifs are aligned to highlight the varying distances and orientations between the half-sites. Number of non-overlapping motif sites in non-repetitive regions and E-values shown next to each motif. E-values are calculated according to MEME's heuristic.

The word search takes 32 seconds to find 1248 words. Hierarchical clustering groups these words into 23 clusters, taking 91 seconds to complete. These 23 clusters are converted to PPMs of widths between 16 and 29 bps, providing seeds for the online EM algorithm. Each seed is processed by the online EM algorithm on a separate core in parallel. 20 of the 23 seeds successfully yield motifs within 15 different values of the bias factor ψ (2.13). The Supplementary material of the article this chapter is based on contains these 20 motifs in MEME Minimal Motif Format [76]. Hierarchical clustering of the 20 motifs groups them

into 10 clusters. Online EM running times range from 67 seconds to 859 seconds, taking an average of 361 seconds. Running times vary because different seeds can converge to different motifs and may require additional passes through the data to reach convergence. For comparison, we also run MEME on the ChIP-Seq dataset to find a single motif of a width between 16 and 29 bps under the two-component model using a single core. MEME takes 8191 seconds to find a single motif. While comparison between the multi-core EXTREME run to the single-core MEME run is not straight-forward, it should be noted that the total computing time for EXTREME, which sums the running times for the seeding and each of the online EM runs, is 8305 s. In the computing time it takes for MEME to discover a single motif, EXTREME finds 10 motif clusters in roughly the same amount of time. The disparity between the two programs' performances is compounded by the fact that MEME discovers multiple motifs in serial, and would require roughly the same running time to find each additional motif.

Many of the discovered motifs are novel, demonstrating varying half-site distances and orientations (Fig. 2.4). Interestingly, two of the discovered motifs show that the half-sites are reversed. To determine whether the reversed motif is functional, we scan for sequences in the ChIP-Seq dataset matching one of the reversed motifs' consensus sequence, align these sequences, and extract GERP scores [25]. Sequences containing this reversed motif are enriched in high GERP scores, showing that these sequences are conserved and possibly functional (Fig. 2.5).

Some of the motifs discovered in this dataset have a low number of occurrences. One of the motifs, for example, only has 11 sites in the data. It would be very unlikely to discover these infrequent motifs if the majority of sequences are discarded. This highlights the importance of using the entire dataset for thorough motif discovery.

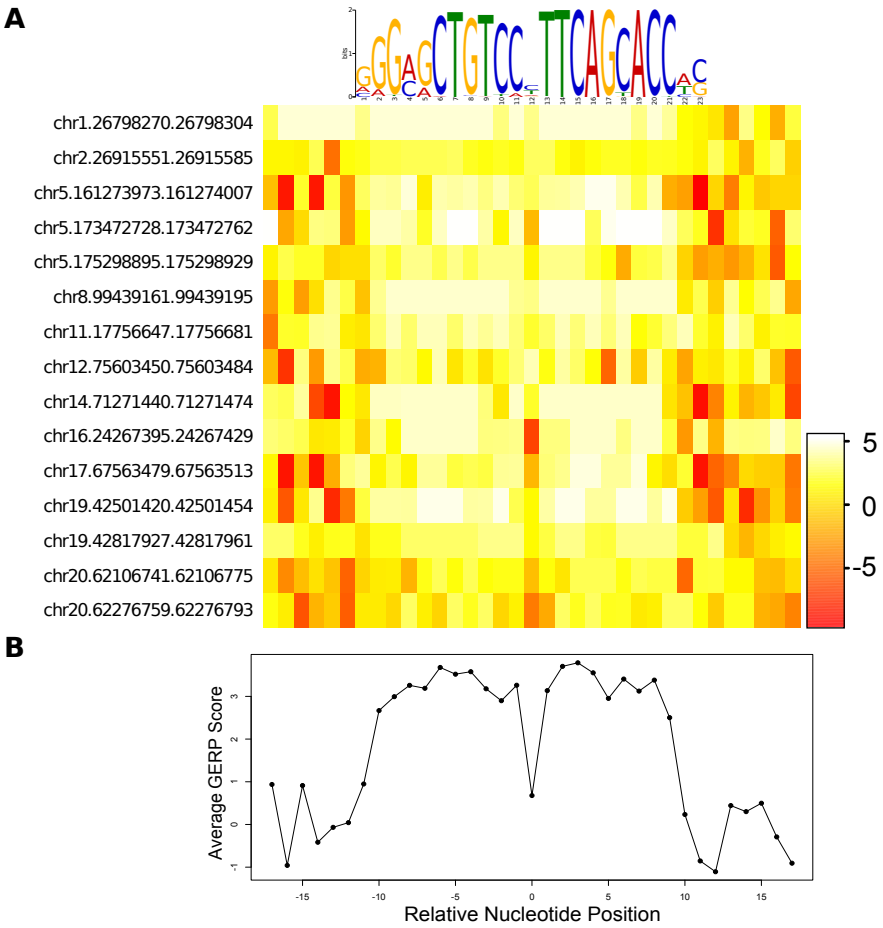


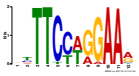
Figure 2.5: **Conservation analysis of the reversed REST motif.** Sequences in the GM12878 REST ChIP-Seq dataset containing the consensus sequence GCTGTCC-NTTCAGCA or its reverse complement are aligned with 10 bp flanking sequences. *(A)* GERP scores are plotted for each nucleotide in a heatmap. The motif's sequence logo is aligned at the top for reference. *(B)* The average GERP score plotted against the genomic positions, relative to the center of the alignment.

2.3.3 Discovering motifs in DNase-Seq data

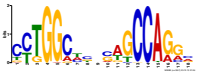
To assess the performance of MEME and EXTREME for DNase-Seq data, we use a DNase-Seq FP dataset generated by the Stamatoyannopoulos Lab at the University of Washington [70]. The DNase-Seq data correspond to a footprinting experiment performed on the K562 cell line. FPs are already organized into a BED file by the authors. We further process the FP data by extending each FP by 5 bps on each side and then merging any intersecting regions. Genomic sequences are extracted from the masked hg19 reference genome. The

(A) $n_{sites} = 10186$
$E = 1.8 \times 10^{-8878}$

(B) $n_{sites} = 11072$
$E = 5.0 \times 10^{-7021}$



(C) $n_{sites} = 931$
$E = 4.5 \times 10^{-353}$



(D) $n_{sites} = 923$
$E = 4.3 \times 10^{-1130}$

(E) $n_{sites} = 464$
$E = 2.4 \times 10^{-897}$

(F) $n_{sites} = 1771$
$E = 6.0 \times 10^{-184}$

Figure 2.6: **Six examples of motifs discovered by EXTREME in the K562 DNase footprint dataset.** Number of non-overlapping motif sites in non-repetitive regions and E-values shown below each motif. The E-values show how significant the motifs are, calculated according to MEME's heuristic.

resulting dataset consists of 198527 sequences and 10487345 bps.

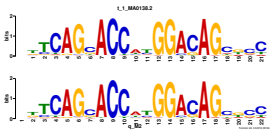
We first discover motifs in the DNase-Seq dataset using MEME. We do not run MEME on the whole dataset because we know MEME can take months to complete for a dataset of this size. We therefore run MEME-ChIP on the dataset, which runs MEME on 600 randomly selected sequences. For the data subset, MEME discovers two motifs that strongly resemble previously discovered motifs (CTCF and SP1). The other discovered motifs are repetitive or fail to meet our E-value threshold of 0.01.

In the seeding step of EXTREME, we first search for words with a half-length of 4, a gap length between 0 and 10 bps, inclusive, a normalized z-score greater than 5, and at least 10 occurrences in the positive sequence dataset. The word search takes 836 seconds to yield 761 words. Hierarchical clustering of the words takes 23 seconds to group the words into 129 clusters. We then convert the clusters to 129 PPM seeds of widths between 8 and 19 bps. Each seed is processed independently by the online EM algorithm on a separate core in parallel. Running times vary for each of the online EM runs, ranging from 4475 seconds to 18300 seconds, completing in an average of 7390 seconds. Hierarchical clustering groups the discovered motifs into 22 distinct clusters.

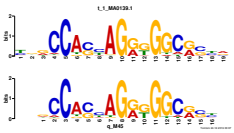
To discover additional motifs in the DNase-Seq data, we mask the 7 most abundant motifs

from different motif clusters by replacing instances of those motifs with capital Ns and restart the motif discovery. We remove these motif instances because the first round of motif discovery shows that many different seeds can converge to the same motif, and we want to bias the motif discovery towards different motifs. Based on TOMTOM [41] analysis, the 7 motifs strongly match known motifs (TOMTOM $E < 0.01$): CTCF, SP1, SRF, NRF1, JUNDM2, ZNF143, and TAL1/GATA1. In this second round of motif discovery, we search for words with a half-length of 5, a gap length between 0 and 10 bps, inclusive, a normalized z-score greater than 8, and at least 10 occurrences in the positive sequence dataset. The word search takes 888 seconds to yield 1187 words. Hierarchical clustering of the words completes in 102 seconds and yields 357 clusters, which are then converted to PPM seeds of widths between 10 and 21bps. Each seed is independently processed by the online EM algorithm on a separate core in parallel. Online EM run times range from 3330 seconds to 22702 seconds, completing in an average of 7605 seconds. Hierarchical clustering condense the motifs into 131 clusters.

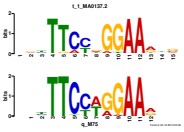
Examples of motifs discovered in the K562 dataset are shown Fig. 2.6. All motifs discovered by EXTREME in the K562 dataset are available in MEME Minimal Motif Format in the Supplementary material of [76]. Many of the motifs discovered by EXTREME have a low number of occurrences relative to the total size of the dataset. One motif only has 464 occurrences in the 10.5 Mbp dataset (Fig. 2.6E). These kinds of motifs are too infrequent to be discovered in subsets of the data. Discovering motifs in a subset of the data is only possible for motifs that are present in high abundance, such as the ones shown in Fig. 2.6A and 2.6B, which are also the motifs discovered by the MEME run on the data subset. Again, this highlights the importance of using the whole dataset for motif discovery. Using MEME to discover these infrequent motifs is not practical because MEME can take months to discover a motif in a dataset as large as the K562 dataset. Furthermore, the number of occurrences for motifs are less than expected. For example, EXTREME only finds 1771 occurrences of the CCAAT box motif (Fig. 2.6F), even though the ENCODE NFYA ChIP-Seq data



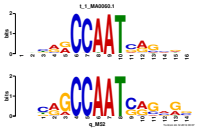
(A) REST, $E = 3.1 \times 10^{-25}$



(B) CTCF, $E = 1.2 \times 10^{-21}$



(C) STAT1, $E = 5.6 \times 10^{-6}$



(D) NFYA, $E = 3.1 \times 10^{-10}$

Figure 2.7: **TOMTOM comparisons of motifs discovered by EXTREME with motifs in databases.** Each panel shows the logo of the motif discovered by EXTREME (lower logo) aligned with the best matching motif in the databases (upper logo), along with the name of the best matching motif and significance value of the match.

indicate it should be present in at least a third of all human promoters. The reason for the discrepancy is likely due to the way [70] called FPs. [70] reported high-confidence FPs at an FDR of 1%. This is a very stringent threshold and we therefore expect their footprinting algorithm to call many false negatives as a result.

2.3.4 Comparison to known motifs

We assess the similarity of the motifs discovered by EXTREME in the DNase-Seq and ChIP-Seq datasets to known motifs using TOMTOM. Some of the motifs discovered by EXTREME have highly significant matches to known motifs (Fig. 2.7). Many of the motifs discovered, however, are novel and fail to meet our TOMTOM E-value threshold of 0.01. Validating these novel motifs requires further computational or experimental scrutiny.

2.4 Discussion

A search-based seeding strategy combined with the online EM algorithm is effective for efficient *de novo* motif discovery in large datasets. EXTREME uses the online EM algorithm

to discover motifs that very closely match motifs discovered by MEME. MEME can take months to discover even a single motif in a large dataset like the DNase-Seq dataset. While strategies such as discarding sequences is effective for quickly discovering abundant motifs, it is insufficient for finding infrequent motifs, which are numerous in DNase-Seq data. EXTREME can quickly process entire large datasets without discarding sequences or using specialized hardware. If available, EXTREME can take advantage of parallelized hardware configurations, which is useful for rapidly discovering multiple motifs in large datasets. Although such configurations are not available to all researchers, EXTREME can still be used with more traditional configurations to serially discover multiple motifs at a substantially faster rate than MEME can.

We expect EXTREME to be a valuable tool for thorough motif discovery in large datasets. Its ability to discover multiple motifs in DNase-Seq data will be especially useful for understanding transcriptional regulation. Because motifs discovered by EXTREME closely match motifs discovered by MEME, the results are highly compatible with existing algorithms and databases. For example, a motif discovered by EXTREME in a DNase I footprint dataset can easily be queried against a database of known motifs, which in turn can reliably associate FPs with well-studied TFs. Novel motifs discovered in ChIP-seq datasets can also be confidently associated with the immunoprecipitated TF. This is especially useful for the study of TFs that lack a suitable antibody for ChIP experiments.

While EXTREME is effective in motif discovery, there is still much room for improvement. For example, the code can be rewritten in a faster programming language, such as C. We chose the Python programming for its compatibility and readability, but the language is known to have many overhead issues that make it much slower compared to other programming languages. Although we emphasized that one of EXTREME's strengths is that it does not require specialized hardware, another way we can improve EXTREME's performance is to implement it with a backend that can leverage GPUs. As GPUs become more ubiquitous

in research due to the prevalence of deep learning methods, our initial concerns will become less of an issue. Regardless, many of these backends can seamlessly utilize either CPUs or GPUs. Future updates can also incorporate more MEME features such as the OOPS and ZOOPS models. We are also interested in systematically applying EXTREME to existing ChIP-seq and DNase-seq datasets from ENCODE. By doing so, we can considerably add to existing motif databases, helping fill missing gaps in our understanding of regulatory genomics.

2.5 Software availability

To encourage further investigation, we have made EXTREME publicly available at the Github repository http://github.com/uci-cbcl/EXTREME under the GNU General Public License.

Chapter 3

DANN: annotating the pathogenicity of genetic variants using a deep neural network

3.1 Introduction

Identifying the genetic variants responsible for diseases can be very challenging. The majority of candidate variants lie in noncoding sections of the genome, whose role in maintaining normal genome function is not well understood. Most annotation methods can only annotate protein coding variants, excluding >98% of the human genome. Over 1,200 genome-wide association studies (GWASs) have identified nearly 6,500 disease- or trait-predisposing SNPs, 93% of which are located in noncoding regions [44], highlighting the importance of a predictive model for noncoding variants. One annotation method, Combined Annotation–Dependent Depletion (CADD) [55], can annotate both coding and noncoding variants. CADD trains a linear kernel SVM to separate observed genetic variants from

simulated genetic variants. Observed genetic variants are derived from differences between human genomes and the inferred human-chimpanzee ancestral genome. Because of natural selection effects, observed variants are depleted of deleterious variants. Simulated genetic variants are enriched for deleterious variants.

CADD's SVM can only learn linear representations of the data, which limits its performance. To overcome this, we implemented a DNN algorithm that we have named DANN (**D**eleterious **A**nnotation of genetic variants using **N**eural **N**etworks). A DNN is an artificial neural network with several hidden layers of units between the input and output layers. The extra layers give a DNN added levels of abstraction, but can greatly increase the computational time needed for training. Deep learning techniques and GPU hardware can significantly reduce the computational time needed to train DNNs. DNNs outperform simpler linear approaches such as logistic regression (LR) and SVMs for classification problems involving many features and samples.

The specific type of DNN we propose to implement is called a feedforward neural network, also known as a multilayer perceptron (MLP). One of the defining aspects of an MLP is the dense connections between adjacent layers. MLPs are most appropriately for applied to data where the order of the features are irrelevant. Other applications of MLPs in bioinformatics include gene expression inference [23] and splicing pattern prediction [59].

3.2 Methods

3.2.1 DNN training

The DANN MLP model consists of an input layer, three 1000-node hidden layers, and a sigmoid function output layer (Fig. 3.2). Each node in a non-input layer takes as input the

activation vector from the previous layer and applies a non-linear transformation. For example, the first node in the first hidden layer transforms the input vector, $x = (x_1, x_2, \ldots, x_9 49)$, and outputs the following value:

$$h_1^{(1)} = f\left(\sum_{i=1}^{949} W_{1,i}^{(0,1)} x_i + b_1^{(0,1)}\right) \tag{3.1}$$

where $W^{(j,j+1)}$ denotes the weight matrix between the j-th and the next layer, $b^{(j,j+1)}$ is the corresponding bias vector, and f is a non-linear activation function. If we extend f to apply to vectors in an element-wise fashion, then we can compactly write the model as a series of matrix multiplications and non-linear transformations mapping from the input vector, x, to the scalar output, $\hat{y}$:

$$\begin{aligned} h^{(1)} &= f\left(W^{(0,1)} x + b^{(0,1)}\right) && (1) \\ h^{(2)} &= f\left(W^{(1,2)} h^{(1)} + b^{(1,2)}\right) && (2) \\ h^{(3)} &= f\left(W^{(2,3)} h^{(2)} + b^{(2,3)}\right) && (3) \\ \hat{y} &= \sigma\left(W^{(3,4)} h^{(3)} + b^{(3,4)}\right) && (4) \end{aligned}$$

Our choice of the hidden activation function, f, is the hyperbolic tangent function:

$$f(z) = tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{3.2}$$

Another common choice for the hidden unit is the rectified linear unit (ReLU):

$$f(z) = \max(0, z) \tag{3.3}$$

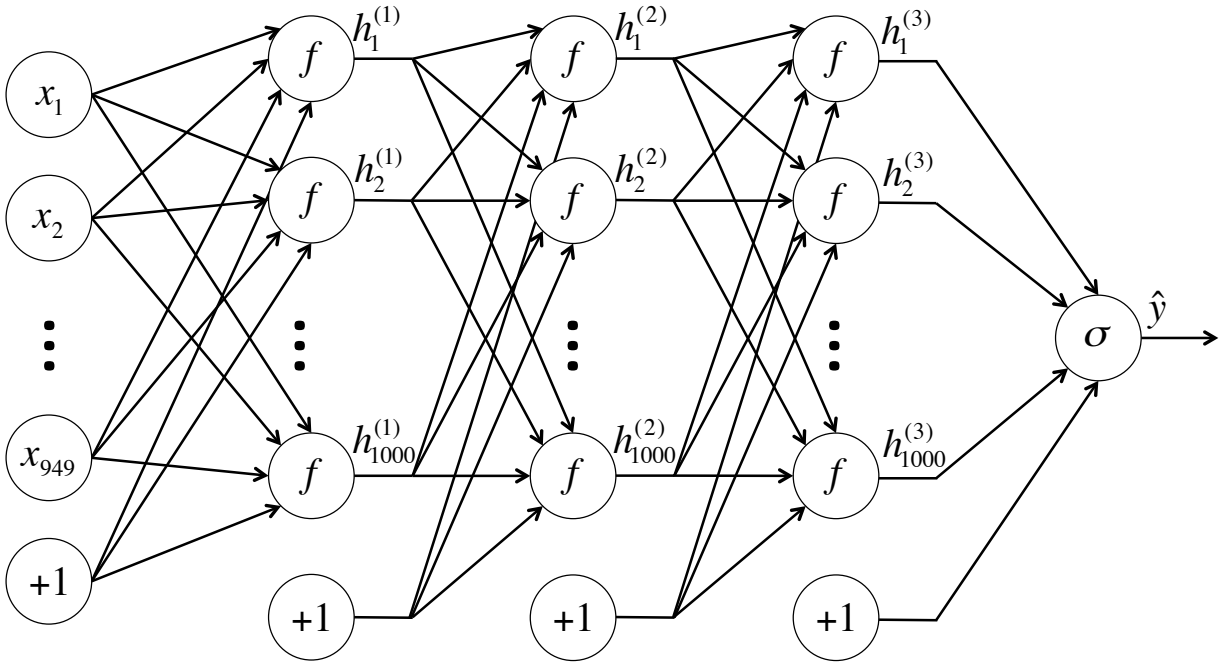


Figure 3.1: **Graphical illustration of the feedforward neural network.**

We do not employ ReLUs in this study, but we do heavily use them in chapters 4 and 5. Yet another common choice for the activation function, f, is the sigmoid function, which we use as the final transformation in the output layer:

$$f(z) = \sigma(z) = \frac{1}{1 + e^{-z}} \tag{3.4}$$

For binary classification tasks, the sigmoid function is a natural choice for the final output. It is a real-valued and differentiable function, making it amenable for gradient-based methods, and it is bounded between zero and one, making it comparable to the binary labels of classification tasks. The sigmoid output of the model, $\hat{y}$, is interpreted as the probability that the model assigns to the true label, y, to be one. Observed and simulated variant labels correspond to binary labels zero and one, respectively.

The goal of model training is to find the "best" set of weights, θ. This can be achieved by defining a loss function, L, and applying the standard (or "batch") gradient descent to iteratively update the weights toward a local minimum according to the following rule:

$$\theta := \theta - \gamma \nabla_\theta L \tag{3.5}$$

where γ is the learning rate and is typically set to a small constant. The gradient, $\nabla_\theta L$, can be efficiently computed with the backpropagation algorithm [86].

In this application, we minimize the mean cross entropy loss function, a common choice for binary classification tasks, across N training samples:

$$L = -\frac{1}{N} \sum_{n=1}^{N} \left[y_n \log \hat{y}_n + (1 - y_n) \log \left(1 - \hat{y}_n\right) \right] = \frac{1}{N} \sum_{n=1}^{N} L_n \tag{3.6}$$

The gradient descent algorithm shares many similarities with the EM algorithm that we described in the previous chapter. Both algorithms optimize an objective function and can be trained in either a batch or online fashion. Many of motivations for using an online version over the batch version are also shared between the EM and gradient descent algorithms, including memory efficiency and faster convergence. The online version of gradient descent, which is commonly called the stochastic gradient descent (SGD), updates the weights after processing each individual training sample according to the following rule:

$$\theta := \theta - \gamma \nabla_\theta L_n \tag{3.7}$$

A compromise between online and batch versions of gradient descent is to compute the gradient against a "mini-batch" of training examples at each iteration. This modification can perform significantly better than true SGD because the code can make use of fast parallelized vectorization libraries rather than computing each step separately. It may also result in smoother convergence, as the gradient computed at each step uses more training examples. This is the more commonly used version of gradient descent, and it is the one we use to train the DANN model. We use a mini-batch size of 100.

We also apply dropout, a method that reduces overfitting by randomly setting a set a fraction of the activations to zero [94]. We set the hidden node dropout rate to 0.1.

To reduce training time, we apply momentum training, which adjusts the parameter increment as a function of the gradient and learning rate [97]. It slightly modifies the SGD update (eqn. 3.7) as follows:

$$\nu \coloneqq \mu\nu + \gamma\nabla_\theta L_n \quad (1)$$
$$\theta \coloneqq \theta - \nu \quad (2)$$

where μ and γ are the momentum rate and learning rate, respectively. We fix the learning rate to a value of 0.01. The momentum rate is initially 0.01, and linearly increases to 0.99 at the end of each epoch for the first 10 epochs and then remains at 0.99.

We trained the DNN using deepnet (https://github.com/nitishsrivastava/deepnet) on an NVIDIA Tesla M2090 card. We chose the deepnet package because of its support for sparse matrices.

3.2.2 Models for comparison

As a baseline comparison, we trained a LR model. For LR training, we applied SGD using the scikit-learn library [72] with parameter $\alpha = 0.01$, which we found to maximize the

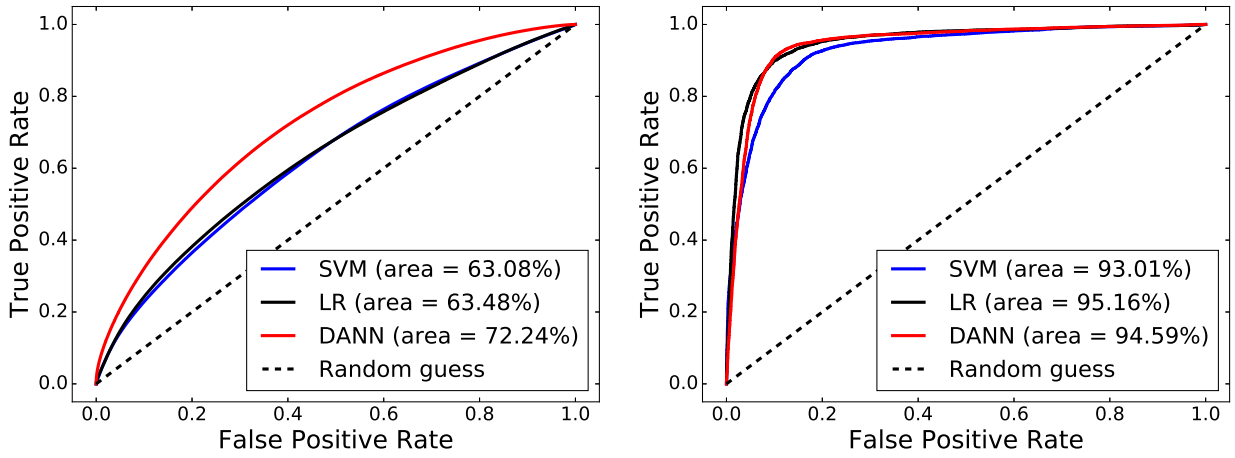
accuracy of the LR model. We also train an SVM using the LIBOCAS v0.97 library [32] with parameter $C = 0.0025$, closely replicating CADD's training as much as possible.

3.2.3 Features

There are a total of 949 features defined for each variant. The feature set is sparse, and includes a mix of real valued numbers, integers, and binary values. For example, amino acid identities are only defined for coding variants. To account for this, we include Boolean features that indicate whether a given feature is undefined, and missing values are imputed. Moreover, all n-level categorical values, such as reference allele identity, are converted to n individual Boolean flags. Other features include evolutionary scores and gene model annotations. See the Supplementary of [55] for more details about the features and imputation. LR and DNN are sensitive to feature scaling, so we preprocess the features to have unit variance before training either model; however, we chose not to zero-mean normalize the features in order to preserve the sparsity of the features.

3.2.4 Training data

CADD's training data consist of 16,627,775 "observed" variants and 49,407,057 "simulated" variants. We trained all three models on this dataset to differentiate the simulated variants from the observed variants. To account for the imbalance between the two datasets, we randomly sampled 16,627,775 simulated variants for training. These 33,255,550 variants are split into a "training set", a "validation set", and a "testing set" in an approximately 8:1:1 ratio. The three models are trained on training set. For SGD, each gradient step is not guaranteed to minimize the loss function; at 1/10 epoch intervals throughout the 20 epochs of training the validation set is evaluated in order to select the "best" model that maximizes classification accuracy on the validation set. The validation set is also used to fine tune



(A) Testing set (B) ClinVar and ESP

Figure 3.2: **ROC curves comparing performances of the neural network (DANN), support vector machine (SVM), and logistic regression (LR) models**. The models are evaluated on discriminating (A) "simulated" variants from "observed" variants in the testing set and (B) pathogenic ClinVar variants from likely benign ESP alleles (DAF $\geq$ 5%). A random guess would give a point along the dashed line (the so-called line of *no-discrimination*) from the bottom left to the top right corners.

hyperparameters such as dropout rate, minibatch size, etc. Finally, the models are regularly evaluated on the testing set to monitor for overfitting. In contrast, [55] trained CADD using an "ensemble" strategy that involves training SVMs on ten different subsets of the training data. We found little performance improvement when we applied this strategy.

3.3 Results and Discussion

To compare the performance of the three models, we generated receiver operating characteristic (ROC) curves discriminating the 3,326,573 simulated and observed variants in the testing set and calculated the area under the curve (AUC) (Fig. 3.2). We used the discriminant values of the SVM and the sigmoidal function output of the DNN and LR models as classifiers for the ROC curves. We do not directly compare to CADD because it can only evaluate 100,000 variants at a time and CADD was already trained on testing set variants; however, the SVM we trained performs very similarly to CADD despite being trained on a

smaller dataset (data not shown). The classification accuracies of the SVM, LR, and DNN models are 58.2%, 59.8%, and 66.1%, respectively. A few observations emerge from our analysis. First, LR performs better than SVM, suggesting that the max margin regularization employed by SVM plays little role in this particular dataset. Second, DNN performs significantly better than both LR and SVM, leading to a 19% reduction in the error rate and 14% improvement in the AUC relative to SVM. This suggests the importance of accounting for nonlinear relationships among features, likely due to the heterogeneity of features generated in genome annotations. Third, although DNN improves on the linear methods, its accuracy is still unsatisfactory. We suspect a few factors might contribute to this: 1) The training data are inflated with mislabeled samples. Observed variants can be under positive or weak purifying selection, and therefore be functional. Conversely, many simulated variants can be nonfunctional since they are randomly sampled from the genome. 2) The features currently used in genome annotation are insufficient for functional prediction. 3) The model training needs further improvement.

We also generated ROC curves showing the models discriminating pathogenic mutations defined by the ClinVar database [10] from likely benign Exome Sequencing Project (ESP) [33] alleles with a derived allele frequency (DAF) $\geq 5\%$ (Fig. 3.2B, n = 10,000 pathogenic/10,000 likely benign). Coding variants constitute 85.6% and 43.0% of the ClinVar and ESP databases, respectively, reducing the difficulty of annotation since many more informative features are available in coding regions. For variants with multiple gene annotations, we only selected the gene annotation that yielded the highest score from each model. All three models greatly improve on the AUC metric, with the LR and DANN models outperforming SVM; however, the performance gap between the models is much smaller than the gap in the testing set.

The differences between the testing set and the ClinVar/ESP set brings up several questions. First, what is the proper training and evaluation sets for measuring the pathogenicity of genetic variants. Ideally, for the purposes of machine learning, the training, validation, and

evaluation sets should all come from the same probability distribution. Clearly, this is not the case as the ClinVar/ESP set carries a drastically different proportions of coding variants than the training or testing sets. While this may be seen as a design flaw on part of the CADD authors for curating their training and evaluation sets as they did, this scenario is often unavoidable in genomics, owing to a variety of factors such as difficulty in reproducing results and biases in data curation. We surmise, based on our results, that DANN can generally predict variant pathogenicity better than the other methods, but this conclusion relies on the assumption that a better performance on the testing set leads to better performance for predicting pathogenicity. This also raises the question of what actually constitutes a pathogenic or deleterious variant. CADD drew an equivalence between disease and a reduction in organismal fitness; however, some diseases may not necessarily impact fitness. Other methods, such as fitCons [40] and delta-SVM [56], evaluate their methods on cell type-specific datasets and claimed better performance than CADD, without using the same evaluations sets as CADD. As of the writing of this thesis, there appears to be no consensus for evaluating variant annotation methods. Finally, we also question how the evaluation sets are generated. The majority of genetic variation is benign, but the evaluation sets tend to be artificially balanced. Most genomic datasets are very sparse. For example, a TF may be bound to a few thousand sites in the genome, which covers much less than 1% of the entire genome. Evaluation metrics should reflect this imbalance. We will revisit these questions in the subsequent chapters.

In conclusion, we have improved considerably upon CADD's SVM methodology. We can even achieve better performance with LR, suggesting that LR is the preferred linear classifier in this case. DANN achieves the best overall performance, substantially improving upon the linear methods in terms of accuracy and separation on the testing set, which contains mostly noncoding variants. This makes DANN the most useful annotation algorithm since the vast majority of human variation is noncoding. When limited to a coding biased dataset, all three models perform well, but the performance gap is small. Given DANN's superior

performance for annotating noncoding variants, which comprise the overwhelming majority of genetic variation, we expect DANN to play an important role in prioritizing putative causal variants, such as those derived from GWAS, for further downstream analysis.

3.4 Software availability

Source code, training data, and testing data described in this chapter are available at https://cbcl.ics.uci.edu/public_data/DANN/ under the MIT license. For convenience, we provide pre-computed DANN scores for all possible SNPs in the hg19 genome. Since SNPs subsume the vast majority of genetic variation, these scores should meet the demands of most research applications. DANN SNP scores are also available in several widely used general annotation programs, including ANNOVAR [101], Varsome (https://varsome.com/), and dbNSFP [61].

Chapter 4

DanQ: stacked convolutional and recurrent neural network for predicting chromatin markers from raw nucleotide sequences

4.1 Introduction

In the previous chapter, we describe several annotation methods that integrate various genomic datasets, such as evolutionary scores and gene model annotations. However, there has been a growing interest to predict function directly from raw genomic sequence, without using any curated datasets such as gene models or multiple species alignments. Previous studies have shown that the sequence features alone can predict epigenomic elements, sometimes even in a cell type-specific manner [103, 12, 36]. A model that can predict function directly from sequence may reveal novel insights about noncoding elements without any

bias introduced by the curated datasets, echoing the motivations discussed in the previous chapter.

Convolutional neural networks (CNNs) are variants of DNNs that are appropriate for the aforementioned task. CNNs use a weight-sharing strategy to capture local patterns in structured data such as sequences. This weight-sharing strategy is especially useful for studying DNA because the convolution filters, or kernels, can capture sequence motifs. Consequently, the CNN training process, which is also typically done in an online fashion, is essentially performing motif discovery. In this regard, a DNA sequence-based CNN shares many similarities with our EXTREME algorithm. One key difference between the two algorithms is that the EM algorithm used in EXTREME is often viewed as an "unsupervised" algorithm whereas CNNs are typically trained in a "supervised" fashion. DeepBind is one such method that uses a CNN for the purpose of identifying the sequence specificities of DNA- and RNA-binding proteins [3]. DeepSEA is a recently developed algorithm that utilizes a CNN for predicting DNA function [105]. The CNN is trained in a supervised joint multi-task fashion to simultaneously learn to predict large-scale chromatin-profiling data, including TF binding, DNase I sensitivity and histone-mark profiles across multiple cell types, allowing the CNN to learn tissue-specific functions. It claims to significantly outperform gkm-SVM [36], a related algorithm that can also predict the regulatory function from raw DNA sequences, but uses an SVM instead of a CNN for predictions. To predict the effect of regulatory variation, both gkm-SVM [57] and DeepSEA use a similar strategy of predicting the function of both the reference and allele sequences and processing the score differences.

Another variation of DNNs is the recurrent neural network (RNN). Unlike a CNN, connections between units of a RNN form a directed cycle. This creates an internal state of the network that allows it to exhibit dynamic spatial behavior. A bi-directional long short-term memory network (BLSTM) is a variant of the RNN that combines the outputs of two RNNs, one processing the sequence from left to right, the other one from right to left. Instead

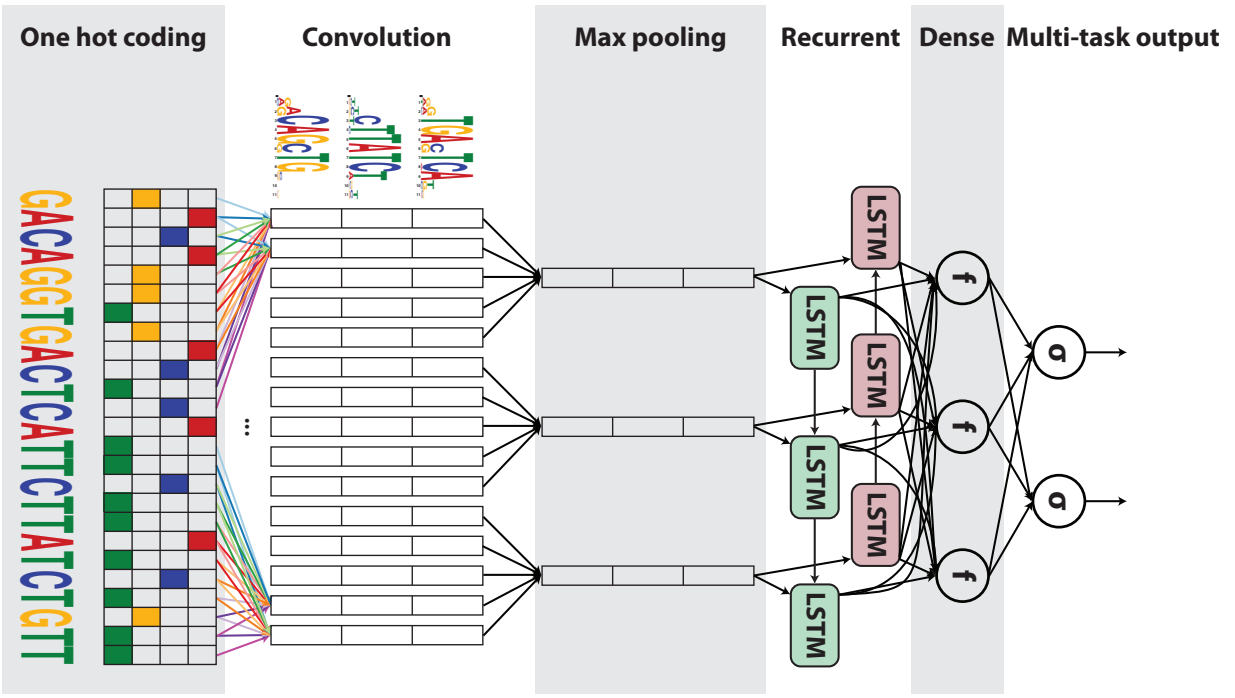


Figure 4.1: **Graphical illustration of the DanQ model.** An input sequence is first one hot encoded into a 4-row bit matrix. A convolution layer with rectifier activation acts as a motif scanner across the input matrix to produce an output matrix with a row for each convolution kernel and a column for each position in the input (minus the width of the kernel). Max pooling reduces the size of the output matrix along the spatial axis, preserving the number of channels. The subsequent BLSTM layer considers the orientations and spatial distances between the motifs. BLSTM outputs are flattened into a layer as inputs to a fully connected layer of rectified linear units. The final layer performs a sigmoid nonlinear transformation to a vector that serves as probability predictions of the epigenetic marks to be compared via a loss function to the true target vector.

of regular hidden units, the two RNNs contain LSTM blocks, which are "smart" network units that can remember a value for an arbitrary length of time. BLSTMs can capture long-term dependencies and have been effective for other machine learning applications such as phoneme classification [39], speech recognition [38], machine translation [96], and human action recognition [106]. Although BLSTMs are effective for studying sequential data, they have not been applied for DNA sequences.

Hence, we propose DanQ (carrying on the theme of naming annotation algorithms after the author), a hybrid framework that combines CNNs and BLSTMs (Figure 4.1). The first

layers of the DanQ model are designed to scan sequences for motif sites through convolution filtering. Whereas the convolution step of the DeepSEA model contains three convolution layers and two max pooling layers in alternating order to learn motifs, the convolution step of the DanQ model is much simpler and contains one convolution layer and one max pooling layer to learn motifs. The max pooling layer is followed by a BLSTM layer. Our rationale for including a recurrent layer after the max pooling layer is that motifs can follow a regulatory grammar governed by physical constraints that dictate the in vivo spatial arrangements and frequencies of combinations of motifs, a feature associated with tissue-specific functional elements such as enhancers [76, 78]. Following the BLSTM layer, the last two layers of the DanQ model are a dense layer of rectified linear units and a multi-task sigmoid output, similar to the DeepSEA model.

DanQ surpasses other methods for predicting the properties and function of DNA sequences across several metrics. In addition, we show that the convolution kernels learned by the model can be converted to motifs, many of which significantly match known motifs. We expect DanQ to provide novel insights into noncoding genomic regions and contribute to understanding the potential functions of complex disease- or trait-associated genetic variants.

4.2 Methods

4.2.1 Features and data

The DanQ framework uses the same features and data as the DeepSEA framework. Briefly, the human GRCh37 reference genome was segmented into non-overlapping 200-bp bins. Targets were computed by intersecting 919 ChIP-seq and DNase-seq peak sets from uniformly processed ENCODE [29] and Roadmap Epigenomics [84] data releases, yielding a length 919 binary target vector for each sample. Each sample input consists of a 1,000-bp sequence

centered on a 200-bp bin that overlaps at least one TF binding ChIP-seq peak, and is paired with the respective target vector. Based on this information, we expected that each target vector would contain at least one positive value; however, we found that about 10% of all target vectors were all negatives. Each 1,000-bp DNA sequence is one-hot encoded into a 1,000 x 4 binary matrix, with columns corresponding to A, G, C and T. Training, validation, and testing sets were downloaded from the DeepSEA website. Samples were stratified by chromosomes into strictly non-overlapping training, validation and testing sets. The validation set was not used for training or testing. Reverse complements are also included, effectively doubling the size of each dataset.

For evaluating performance on the test set, the predicted probability for each sequence was computed as the average of the probability predictions for the forward and reverse complement sequence pairs, similar to DeepSEAs evaluation experiments.

4.2.2 Long Short-Term Memory (LSTM) architecture

Given an input vector sequence $x = (x_1, x_2, \ldots, x_T)$, a standard RNN computes the hidden vector sequence $h = (h_1, h_2, \ldots, h_T)$. Unlike a CNN or MLP, which computes hidden activations as only a function of the inputs, an RNN computes hidden activations as a function of both the inputs and adjacent activation units from $t = 1$ to $t = T$ as follows:

$$h_t = \mathcal{H}(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \tag{4.1}$$

where W denotes weight matrices (e.g. W_{xh} is the input-hidden weight matrix), b denotes bias vectors (e.g. b_h is hidden bias vector), and $\mathcal{H}$ is the hidden layer function. The zeroth hidden vector, h_0, is often set to the zero vector.

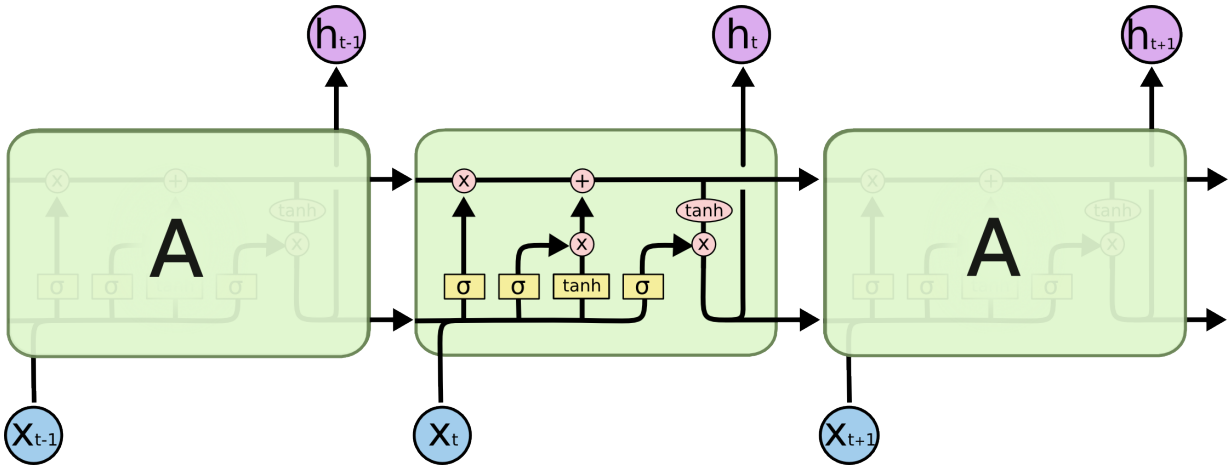


Figure 4.2: **Graphical illustration of the repeating modules of an LSTM network.** Courtesy of Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/).

$\mathcal{H}$ is usually an element-wise non-linear function, such as the sigmoid function (Eqn. 3.4). An LSTM network is a type of RNN that treats $\mathcal{H}$ as four interacting layers (Fig. 4.2). It is designed to capture long-term dependencies in sequential data, addressing a deficit commonly plaguing RNNs [45]. For an LSTM network, $\mathcal{H}$ is implemented by the following composite function:

Input gate	$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i)$	(1)
Forget gate	$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f)$	(2)
Candidate state	$\widetilde{C}_t = \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c)$	(4)
Cell state	$C_t = f_t C_{t-1} + i_t \widetilde{C}_t$	(5)
Output gate	$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$	(4)
Output	$h_t = o_t \tanh(c_t)$	(6)

Intuitively, LSTM networks are able to retain information over long ranges due to the cell state (5). The cell state runs straight down the entire chain of LSTM modules, with only some minor linear interactions. Consequently, the cell states can achieve arbitrarily large positive or negative values. In contrast, traditional RNNs use "squashing" functions such as the sigmoid or hyperbolic tangent function that cause the gradient (error signal) to decrease exponentially with the number of recurrent layers, causing the front layers to train very slowly. The cell state helps LSTM networks avoid this "vanishing gradient problem", but

LSTM networks are still prone to the "exploding gradient problem". Typically, the latter is addressed by clipping gradients to a maximum value.

4.2.3 DanQ model and training

Detailed specifications of the DanQ neural network architectures are presented below. Numbers to the right of the forward slash indicate values unique to the DanQ-JASPAR model, a larger model in which about half of the convolution kernels are initialized with motifs from the JASPAR database [64].

1. **Convolution layer.** (320/1024 kernels. Window size: 26/30. Step size: 1.)
2. **Pooling layer.** (Window size: 13/15. Step size: 13/15.)
3. **BLSTM layer.** (320/512 forward and 320/512 backward LSTM neurons)
4. **Fully connected layer.** (925 ReLU neurons)
5. **Sigmoid output layer.** (919 sigmoid neurons)

The first model we trained contains, referred to as DanQ, contains 46,926,479 free weight and was trained for 60 epochs, each training epoch completing in approximately 6 hours. The second model we trained, which we designated as DanQ-JASPAR because half of the kernels are initialized with motifs from the JASPAR database [64], contains 67,892,175 free weights, was trained for 30 epochs, and each epoch of training completes in approximately 12 hours. In comparison, the DeepSEA model contains three convolutional and two max pooling layers, and 52,843,119 free weights.

Dropout [94] is included to randomly set a proportion of neuron activations from the max pooling and BLSTM layers to a value of 0 in each training step to regularize the DanQ models. The dropout parameters are as follows:

- **Layer 2:** 20%
- **Layer 3:** 50%
- **All other layers:** 0%

All weights are initialized by randomly drawing from $unif$(-0.05,0.05) and all biases are initially set to 0. In addition to random initialization, an alternative strategy is to initialize kernels from known motifs: a random subsection of a kernel is set equal to the values of the position frequency matrix minus 0.25, and its corresponding bias is randomly drawn from $unif$(-1.0,0.0). We tried both in our implementation.

We update neural network model weights, θ, using the RMSprop algorithm [100] with a minibatch size of 100 to minimize the training loss. Similar to momentum training [97], the RMSprop algorithm modifies the gradient update (Eqn. 3.7). The main idea behind RMSprop is to divide the learning rate, γ, for a weight by a running average of the magnitudes of recent gradients for that weight.

$$v \coloneqq \rho v + (1-\rho)(\nabla_\theta L_n)^2 \quad (1)$$
$$\theta \coloneqq \theta - \frac{\gamma}{\sqrt{v}+\epsilon}\nabla_\theta L_n \quad (2)$$

where γ is the learning rate, ρ is the "forgetting factor", and ϵ is a "fuzz factor" to prevent division by zero. We set these values to 0.001, 0.9, and 10^{-8}, respectively.

Validation loss is evaluated at the end of each training epoch to monitor convergence and for model selection. Our choice of loss function is the average multi-task binary cross entropy loss function, a slight modification of the single-task binary cross entropy loss function (Eqn. 3.6) to account for the 919 chromatin mark labels per sample:

$$L = -\frac{1}{N}\sum_{n=1}^{N}\frac{1}{919}\sum_{m=1}^{M}\left[y_{n,m}\log\hat{y}_{n,m} + (1-y_{n,m})\log\left(1-\hat{y}_{n,m}\right)\right] = \frac{1}{N}\sum_{n=1}^{N}L_n \quad (4.2)$$

where M is the number of labels per sample, y is the label matrix of N rows and M columns and $\hat{y}$ is the predicted label matrix. M is the total number of labels per sample, which for the purposes of this chapter is 919, one for each chromatin mark. In contrast, y and $\hat{y}$ in the single-task version (Eqn. 3.6) are both vectors of length N.

Our implementation utilizes the Keras 0.2.0 library (https://github.com/fchollet/keras) with the Theano 0.7.1 [11, 14] backend. An NVIDIA Titan Z GPU was used for training the model.

4.2.4 Logistic regression

For benchmark purposes, we also trained a logistic regression (LR) baseline model, similar to our evaluations of the DANN model in the previous chapter. Unlike the DanQ and DeepSEA models, the LR model does not process raw sequences as inputs. Instead, the LR model uses zero-mean and unit variance normalized counts of k-mers of lengths 1-5bp as features. The LR model was regularized with a small L2 weight regularization of 10^{-6}. Similar to the training of the DanQ models, the LR model was trained using the RMSprop algorithm with a minibatch size of 100 to minimize the average multi-task binary cross entropy loss function on the training set. Validation loss is evaluated at the end of each training epoch to monitor convergence. We note that this method of training is equivalent to training 919 individual single-task LR models.

4.2.5 Functional SNP prioritization

The DanQ functional SNP prioritization framework shares the same datasets, features, and training algorithm as the DeepSEA functional SNP prioritization framework, essentially exchanging DeepSEA chromatin effect predictions with DanQ chromatin effect predictions. Essentially, it converts the 919 chromatin mark predictions into a single score that can be

directly compared with other annotation methods like DANN [75] and CADD [55]. Briefly, we downloaded positive and negative SNP sets for training and testing. We also downloaded DeepSEA functional SNP scores for these variants for benchmarking purposes. Positive SNPs include expression quantitative trait loci (eQTLs) from the Genome-Wide Repository of Associations between SNPs and Phenotypes (GRASP) database [58] and noncoding trait-associated SNPs identified in GWAS studies from the US National Human Genome Research Institute (NHGRI) GWAS Catalog [102]. Negative SNPs consist of 1000 Genomes Project SNPs [2] with controlled minor allele frequency distribution in 1000 Genomes population. The negative SNPs are further split into training and testing sets, the former consisting of 1,000,000 randomly selected noncoding 1000 Genomes SNPs with minor allele frequency distribution matched with the eQTL or GWAS positive standards and the latter consisting of negative SNPs of varying distances to positive standard SNPs. We trained two boosted ensemble classifier models, one for the GRASP set and one for the GWAS set, using the XGBoost implementation (https://github.com/tqchen/xgboost). The hyperparameters we used for XGBoost training are as follows:

- **booster:** gbtree
- **subsample:** 0.5
- **alpha:** 0.001
- **lambda:** 10
- **eta:** 0.05
- **max_depth:** 0.05
- **max_delta_step:** 1
- **gamma:** 1

- **num_round:** 100 for the GWAS model, 300 for the GRASP model

Features were computed as in Zhou and Troyanskaya [105], replacing DeepSEA chromatin effect predictions with DanQ chromatin effect predictions. All features were standardized to mean 0 and variance 1 before training. Unequal positive and negative training sample sizes were balanced with sample weights. The performance of each model was estimated by tenfold cross-validation and across several negative groups.

4.3 Results

We first train a DanQ model containing 320 convolution kernels for 60 epochs, evaluating the average multi-task cross entropy loss on the validation set at the end of each epoch to monitor the progress of training. To regularize the model, we also include dropout to randomly set a proportion of neuron activations from the max pooling and BLSTM layers to a value of 0 in each training step.

For benchmarking purposes, we compare a fully trained DanQ model to a LR baseline model and the published DeepSEA model. To compare performance among models, we first calculated the area under the receiver operating characteristics curve (ROC AUC) for each of the 919 binary targets on the test set (Figure 4.3). In terms of the ROC AUC score, DanQ outperforms the DeepSEA model for two of the targets as shown in the examples at the top of Figure 4.3, although this performance difference is relatively small. This pattern extends to the remaining targets as DanQ outperforms DeepSEA for 94.1% of the targets, although the difference is again comparatively small with an absolute improvement of around 1-4% for most targets. Despite the simplicity of the LR models, the ROC AUC statistics suggests that LR is an effective predictor, with ROC AUC scores typically over 70%. Given the sparsity of positive binary targets ($\sim$2%), the ROC AUC statistic is highly inflated by

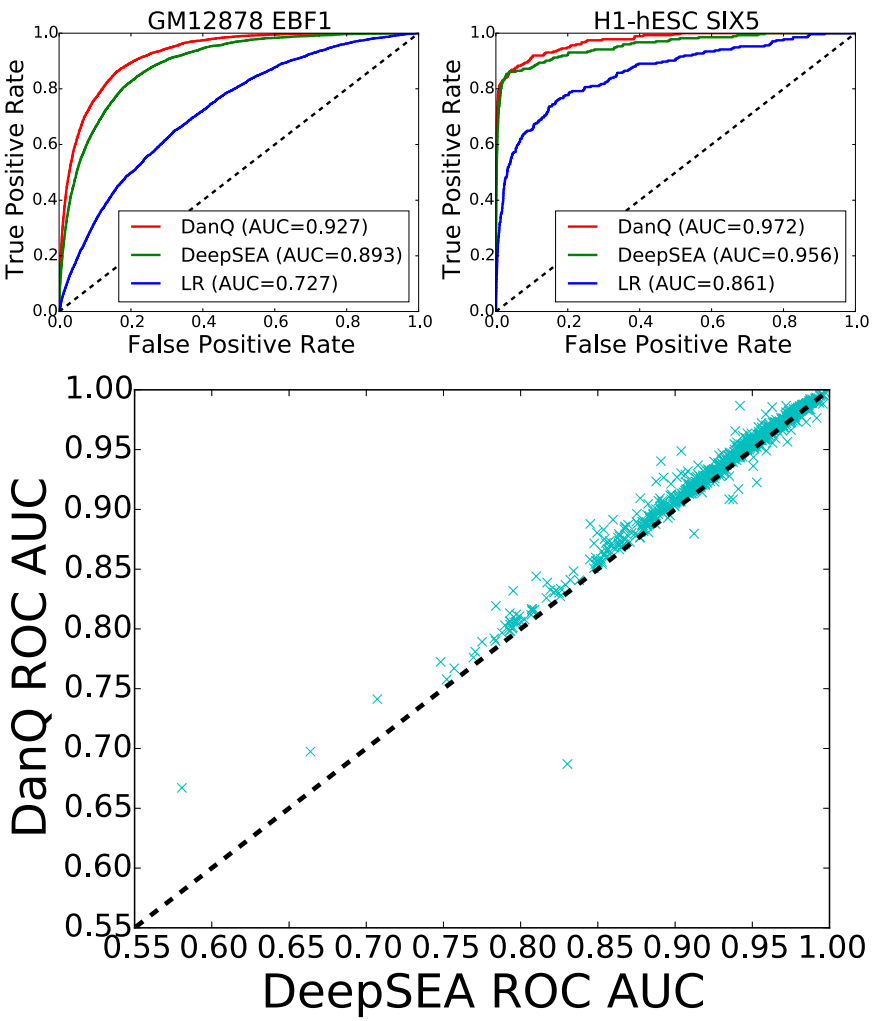


Figure 4.3: **ROC comparison between DanQ and DeepSEA for predicting chromatin marks.** (top) ROC curves for the GM12878 EBF1 and H1-hESC SIX5 targets comparing the performance of the three models. (bottom) Scatterplot comparing DanQ and DeepSEA ROC AUC scores. DanQ outperforms DeepSEA for 94.1% of the targets in terms of ROC AUC.

the class imbalance, a fact overlooked in the original DeepSEA paper.

A better metric to measure the performance is the area under precision-recall curve (PR AUC) (Figure 4.4). Neither the precision nor recall take into account the number of true negatives, thus the PR AUC metric is less prone to inflation by the class imbalance than the ROC AUC metric is. As expected, we found the PR AUC metric to be more balanced, as demonstrated by how the LR models now achieve a PR AUC below 5% for the two examples

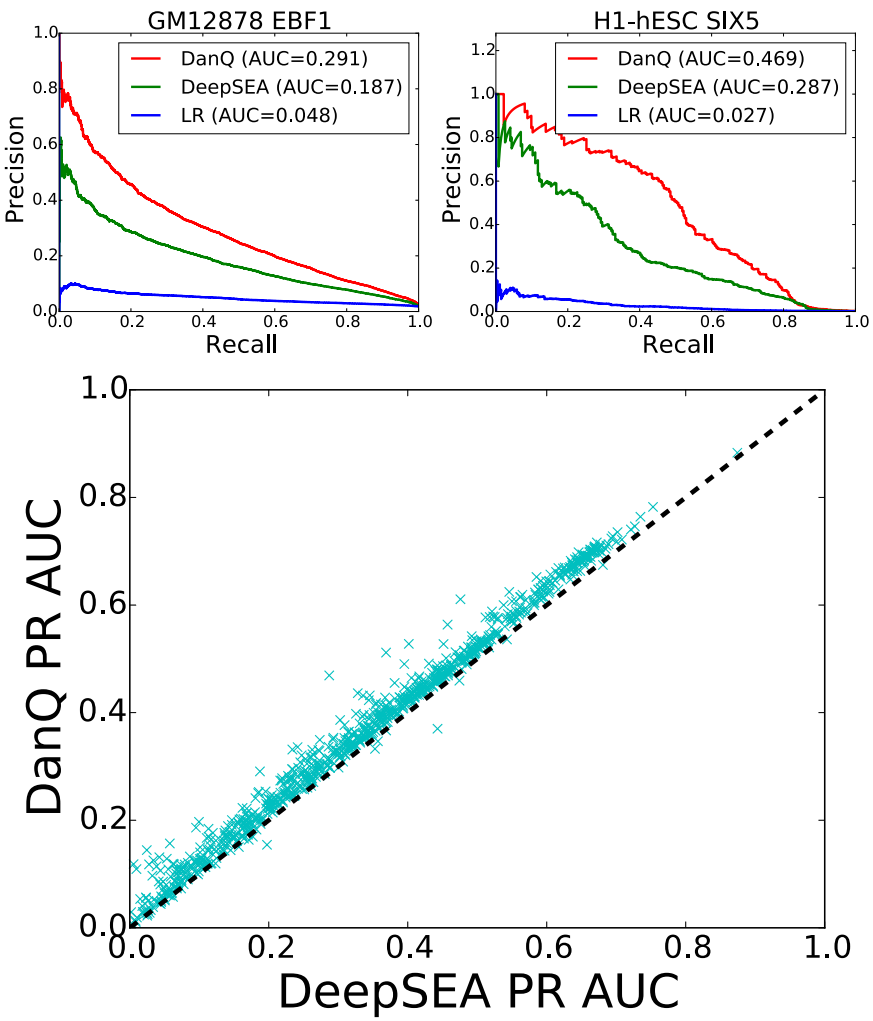


Figure 4.4: **PR comparison between DanQ and DeepSEA for predicting chromatin marks.** (top) PR curves for the GM12878 EBF1 and H1-hESC SIX5 targets comparing the performance of the three models. (bottom) Scatterplot comparing DanQ and DeepSEA PR AUC scores. DanQ outperforms DeepSEA for 97.6% of the targets in terms of PR AUC.

at the top of Figure 4.4, far below the performance of the other two models. Moreover, the performance gap between DanQ between DeepSEA is much more pronounced under the PR AUC statistic than under the ROC AUC statistic. For the two examples shown, the absolute improvement is over 10% and the relative improvement is over 50% under the PR AUC metric, and 97.6% of all DanQ PR AUC scores surpass DeepSEA PR AUC scores. These results show that adding recurrent connections significantly increases the modeling power of DanQ.

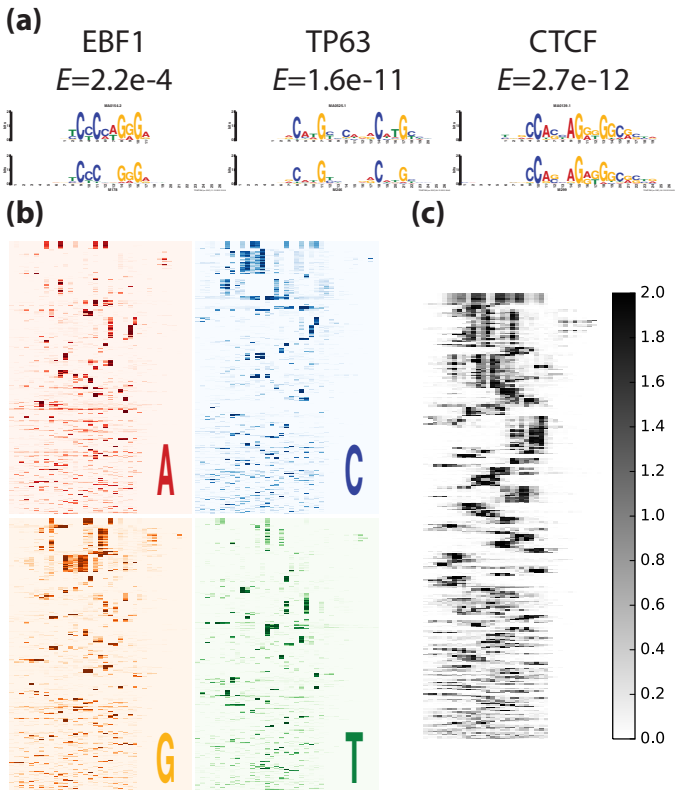


Figure 4.5: (a) Three convolution kernels (bottom) visualized and aligned with EBF1, TP63, and CTCF motif logos (top) from JASPAR using TOMTOM. Significance values of the match are displayed below motif names. (b) All 320 convolution kernels are converted to sequence logos and aligned with RSAT. The heatmaps are colored according to the information content of the respective nucleotide at each position. (c) Same as (b), except the heatmap is colored by the sum of the information content of each letter.

Using a similar approach described in the DeepBind method [3], we converted the kernels from the convolution layer of the DanQ models to position frequency matrices, or motifs. Then, we aligned these motifs to known motifs using the TOMTOM algorithm [41]. Of the 320 motifs learned by the DanQ model, 166 significantly match known motifs ($E < 0.01$) (Fig. 4.5A). Next, we aligned and clustered the 320 motifs together into 118 clusters using the RSAT matrix clustering tool [66], and confirmed that the model learned a large variety of informative motifs (Fig. 4.5B and 4.5C).

Given the large scope of the data, we conjectured that our current model did not exhaust the entire space of useful motif features despite the large variety of motifs learned. Moreover,

weight initialization is known to play crucial role for the performance neural networks [97] and we hypothesized that a better initialization strategy can further improve the performance of our neural network. Thus, we trained a larger model containing 1024 convolutional kernels of which about half are initialized with known motifs from JASPAR [64]. This alternative way of initialization can further improve the performance of DanQ (Tab. 4.1).

	Training		Validation		Testing	
Method	Loss	Accuracy	Loss	Accuracy	Loss	Accuracy
Predict all zeros	N/A	97.84%	N/A	98.05%	N/A	97.94%
LR	0.0798	97.90%	0.0743	98.09%	0.0771	97.99%
DeepSEA	0.0551	98.20%	0.0509	98.36%	0.0554	98.21%
DanQ	0.0539	98.23%	0.0491	98.39%	0.0538	98.24%
DanQ-JASPAR	**0.0521**	**98.26%**	**0.0482**	**98.40%**	**0.0533**	**98.24%**

Table 4.1: **Accuracy and loss on training, validation, and testing sets for each of the models.** The DanQ model initialized with JASPAR motifs performed the best across all metrics, as indicated in bold. Note that due to the huge class imbalance, all models achieved high accuracies.

Finally, we extended DanQ to prioritize functional SNPs based on differences of predicted chromatin effect signals between reference and allele sequences. This analysis replicates DeepSEA's framework, which converts 919 pairs of chromatin marker predictions into a single scores, allowing a more direct comparison with annotation methods such as CADD [55], GWAVA [83], and FunSeq2 [34]. The authors of DeepSEA claim their results demonstrate that DeepSEA can outperform CADD in this task. Specifically, we downloaded training and testing SNP sets [105] that we used to train and evaluate boosted ensemble classifiers. The positive SNPs are annotated "functional" noncoding positive SNPs are eQTL SNPs from the GRASP database [58] and noncoding trait-associated SNPs identified in GWAS studies from the US NHGRI GWAS Catalog [102]. Negative "nonfunctional" variant standards consist of 1000 Genomes Project SNPs [2] with controlled minor allele frequency distribution in 1000 Genomes population. These variant sets are the same sets used by the DeepSEA functional SNP prioritization framework for training and testing. The DanQ framework outperforms the DeepSEA framework across most of the testing sets, with the performance difference

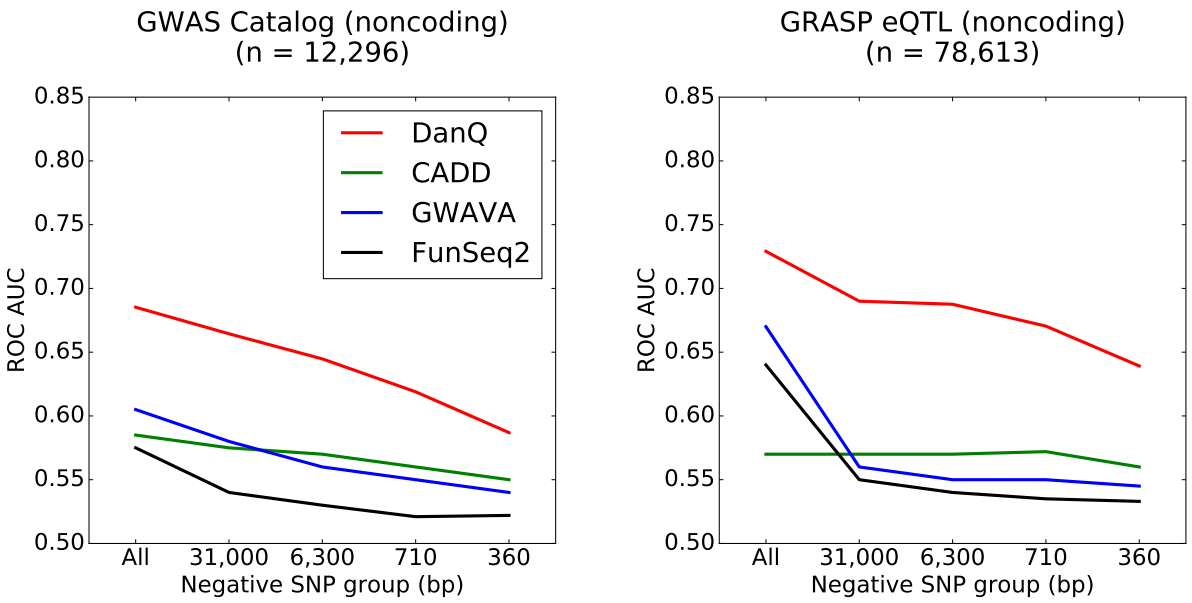


Figure 4.6: **Comparison of the DanQ and DeepSEA methods for functional SNP prioritization**. The positive set includes annotated GRASP eQTLs or GWAS Catalog noncoding SNPs. The negative set includes 1000 Genomes Project noncoding SNPs (across several negative-SNP groups of varying distances to the positive SNPs). The x axes show the average distances of negative-SNP groups to a nearest positive SNP. The All negative-variant groups are randomly selected negative 1000 Genomes SNPs. Model performance is assessed with area under the receiver operating characteristic curves (ROC AUC).

being 0.5-2% in terms of the ROC AUC metric (Fig. 4.6).

4.4 Discussion

In conclusion, DanQ is a powerful method for predicting the function of DNA directly from sequence alone, making it a valuable asset for studying the function of noncoding DNA. Its hybrid architecture allows it to simultaneously learn motifs and a complex regulatory grammar between the motifs. The additional modeling capacity afforded by the recurrent connections allows DanQ to significantly outperform DeepSEA, a pure CNN model that lacks recurrent modeling. This performance gap is demonstrated across several metrics,

including a direct comparison of AUC statistics between the two models. We argue that the PR AUC statistic is a much more balanced metric than the ROC AUC statistic to assess performance in this case due to the massive class imbalance. In fact, the performance gap can be quite drastic under the PR AUC statistic, reaching well over a 50% relative improvement for some epigenetic marks. Despite the significant improvement in performance, there is still much room for improvement because most of the PR AUC scores are below 70% for either model. Furthermore, the significant improvement in chromatin effect prediction does not immediately translate to an equally large improvement in functional variant prediction. One factor that may limit performance in this regard is that while the GRASP eQTL and GWAS catalog SNPs we label as positive variants are associated with phenotypes, these SNPs may not be the causal variants. Instead, the causal variants are likely in linkage disequilibrium with these SNPs. Thus, we hypothesize that extending our framework to study the link between phenotypes and haplotypes instead of phenotypes and individual SNPs may improve prediction performance. Nevertheless, the improved capability of DanQ to predict chromatin effects means it can better predict the epigenetic changes caused by genetic variants, which is useful information for prioritizing the causal variant among a group of tightly-linked variants and predicting the phenotypic outcomes of genome editing, the latter of which is beneficial for several fields including synthetic biology and transgenic animal studies. However, many of the issues raised in the previous chapter are applicable here as well. Although DeepSEA claims it can outperform CADD, the DeepSEA authors only showed this using testing sets of their own choosing, instead of using the same ClinVar/ESP testing set the CADD authors used (Fig. 3.2B). Moreover, the tenfold cross-validation method used to evaluate the DeepSEA's functional SNP prioritization may also have inflated the results in DeepSEA's favor. The other annotation methods DeepSEA was compared to were not provided with the same treatment. We hypothesize that the tenfold cross-validation strategy may actually be picking up implicit biases present between the positive and negative sets, instead of any relevant or useful biological features. For this reason, we chose not to repeat

the tenfold cross-validation procedure for the ClinVar/ESP testing set since the positive and negative sets are known to carry different proportions of coding variants. As a result of this issue, we argue that there is a pressing need in the bioinformatics community to systematize the evaluation of annotation method. This will require agreed-upon evaluation metrics and datasets. Such systems are readily available for other machine learning applications such as MNIST and CIFAR with image recognition, and TIMIT for speech recognition, but not for biological applications.

There are several avenues of future interest to explore. First, the model can be made fully recurrent so it can process sequences of arbitrary length, such as whole chromosome sequences, to generate sequential outputs. In contrast, our current setup can only processes sequences of constant length with static output. A fully recurrent architecture may also benefit our effort to study variants since it would allow us to explore the long-range consequences of genetic variants, as well as the cumulative effects of SNPs that are in linkage disequilibrium with each other. Second, we are interested in incorporating new ChIP-seq and DNase-seq datasets from more cell types as they become available. Incorporating other types of data, such as methylation, nucleosome positioning, and transcription may also yield novel results and improve functional variant prioritization. Finally, we are committed to updating and improving the DanQ model. As our results have shown, the model architecture and weight initialization can influence performance. Previously, we manually selected model parameters. For example, the DanQ model contains 320 kernels because the DeepSEA model also contains 320 kernels in its first convolutional layer, making the two models somewhat more comparable at the architectural level. Interestingly, although our first model contains fewer free weights than DeepSEA, our first model still significantly outperforms DeepSEA. In addition, the choice of 1,024 kernels in the JASPAR-based model was made to accommodate 519 motifs in the JASPAR database in addition to an approximately equal number of randomly initialized kernels. One interesting prospect is to utilize distributed computing-based hyperparameter tuning algorithms to automatically find the optimal combination of model

architecture, initial weights and hyperparameters. We will commit to providing regular updates as the model improves. Also, our motif analysis has shown that neural network training is an effective motif discoverer. Hence our updates will include motifs from the model in MEME minimal format, a flexible format compatible with most motif-related programs, as a resource to the community. To the best of our knowledge, this is the first application of a hybrid convolution and recurrent network architecture for the purpose of predicting function *de novo* from DNA sequences. We expect this hybrid architecture will be continually explored for the purpose of studying biological sequences.

4.5 Software availability

Source code described in this chapter is available at https://github.com/uci-cbcl/DanQ. The github repository also contains motifs derived from the model kernels in minimal MEME format and TOMTOM results comparing the motifs to known motifs in the JASPAR database.

Chapter 5

FactorNet: predicting cell type specific protein-DNA binding from nucleotide-resolution sequential data

5.1 Introduction

The Encyclopedia of DNA Elements (ENCODE) [29] and NIH Roadmap Epigenomics [84] projects have generated a large number of HTS epigenetic datasets for dozens of different cell and tissue types. Owing to several constraints, including cost, time or sample material availability, these projects are far from completely mapping every mark and sample combination. This disparity is especially large for TF binding profiles because ENCODE has profiled over 600 human biosamples and over 200 TFs, translating to over 120,000 possible pairs of biosamples and TFs, but as of the writing of this article only about 8,000 TF binding profiles are available. Due to the strong correlations between epigenetic markers, computational methods have been proposed to impute the missing datasets. One such imputation

method is ChromImpute [31], which applies ensembles of regression trees to impute missing chromatin marks. With the exception of CTCF, ChromImpute does not impute TF binding. Moreover, ChromImpute does not take sequence context into account, which can be useful for predicting the binding sites of TFs like CTCF that are known to have a strong binding motif.

Computational methods designed to predict TF binding include PIQ [90], Centipede [73], and msCentipede [79]. These methods require a collection of motifs and DNase-seq data to predict TF binding sites in a single tissue or cell type. While such an approach can be convenient because the DNase-seq signal for the cell type considered is the only mandatory experimental data, it has several drawbacks. These models are trained in an unsupervised fashion using algorithms such as expectation maximization (EM). From our experience, EM-based algorithms can be very computationally inefficient. To compensate for this issue, PIQ, Centipede, and msCentipede limit training and evaluation to motif matches, which represent a small and unrepresentative fraction of the whole genome. Furthermore, the manual assignment of a motif for each TF is a strong assumption that completely ignores any additional sequence contexts such as co-binding, indirect binding, and non-canonical motifs. This can be especially problematic for TFs like REST, which we showed can have have eight non-canonical binding motifs (Fig. 2.4).

DNNs offer an attractive alternative for solving the imputation problem. They can efficiently identify complex non-linear patterns from large amounts of feature-rich data. As mentioned in the previous chapter, CNNs are RNNs are variants of the DNN that are especially well-suited for studying sequential data, which are ubiquitous in biology. Examples of CNNs in genomics include DanQ[77], DeepSEA [105], Basset [50], and DeepBind [3]. DanQ combines the CNN with an RNN to form a CNN-RNN hybrid architecture that can outperform pure convolutional models (Chapter 4). These four CNN methods accept raw DNA sequence inputs and are trained in a supervised fashion to discriminate between the presence and

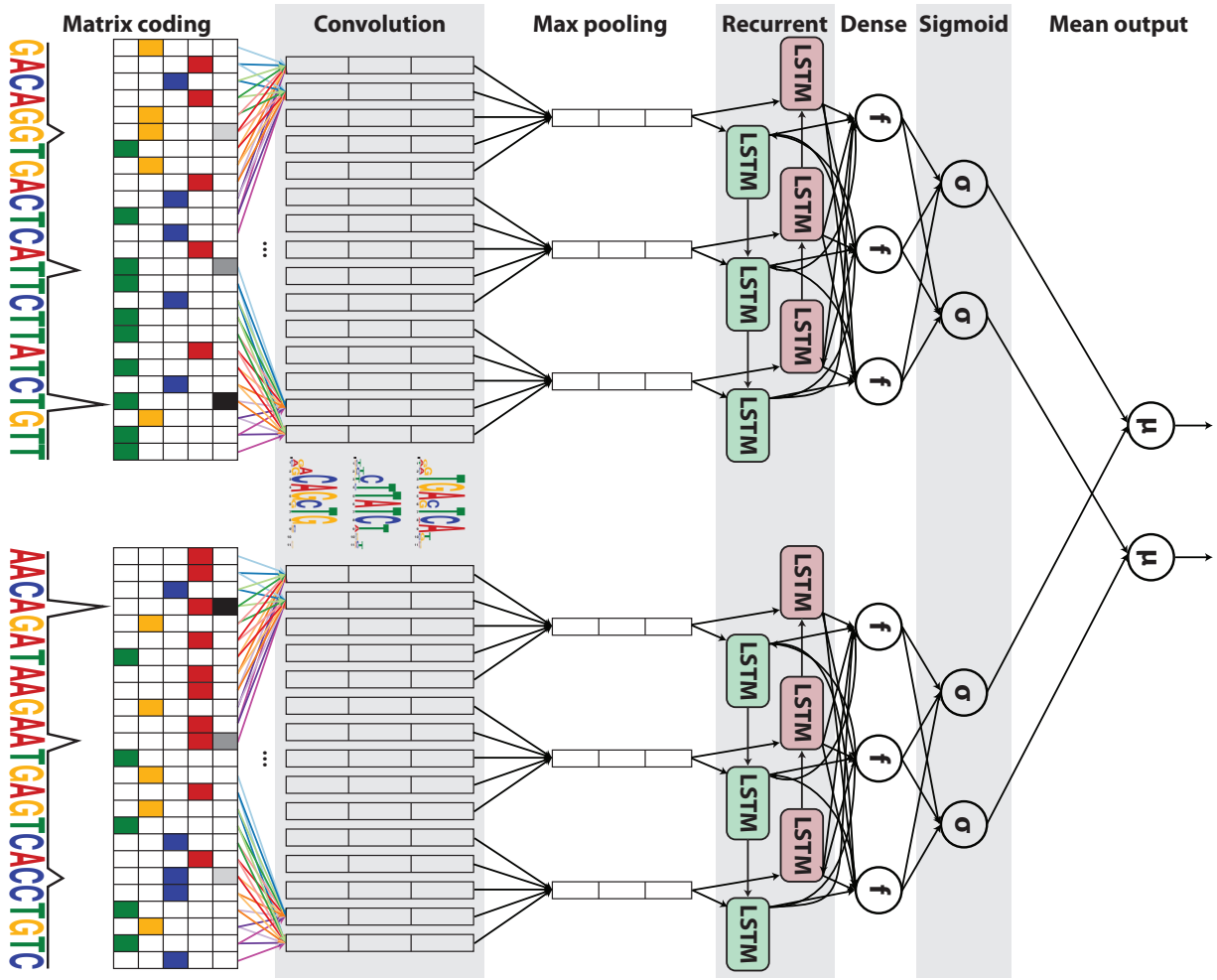


Figure 5.1: **Simplified diagram of the FactorNet model.** An input DNA sequence (top) is first one hot encoded into a 4-row bit matrix. Real-valued single-nucleotide signal values are concatenated as extra rows to this matrix. A rectifier activation convolution layer transforms the input matrix into an output matrix with a row for each convolution kernel and a column for each position in the input (minus the width of the kernel). Each kernel is effectively a sequence motif. Max pooling downsamples the output matrix along the spatial axis, preserving the number of channels. The subsequent recurrent layer contains long short term memory (LSTM) units connected end-to-end in both directions to capture spatial dependencies between motifs. Recurrent outputs are densely connected to a layer of rectified linear units. The activations are likewise densely connected to a sigmoid layer that nonlinear transformation to yield a vector of probability predictions of the TF binding calls. An identical network, sharing the same weights, is also applied to the reverse complement of the sequence (bottom). Finally, respective predictions from the forward and reverse complement sequences are averaged together, and these averaged predictions are compared via a loss function to the true target vector. Although not pictured, we also include a sequence distributed dense layer between the convolution and max pooling layer to capture higher order motifs.

absence of epigenetic markers, including TF binding, open chromatin, and histone modifications. Consequently, these algorithms are not suited to the task of predicting epigenetic markers across cell types. Instead, they are typically designed for other tasks such as motif discovery or functional SNP annotation.

To predict cell type-specific TF binding, we developed FactorNet, which combines elements of the aforementioned algorithms. FactorNet trains a DNN on data from one or more reference cell types for which the TF or TFs of interest have been profiled, and this model can then predict binding in other cell types. The FactorNet model builds upon the DanQ CNN-RNN hybrid architecture by including additional real-valued coordinated-based signals such as DNase-seq signals as features. FactorNet is similar to a recently developed method called DeepCpG, which integrates sequence context and neighboring methylation rates to predict single-cell DNA methylation states using a CNN and a bidirectional RNN [4]. We also extended the DanQ network into a "Siamese" architecture that accounts for reverse complements (Figure 5.1). This Siamese architecture applies identical networks to both strands to ensure that both the forward and reverse complement sequences return the same outputs, essentially halving the total amount of training data, ultimately improving training efficiency and predictive accuracy. Both networks share the same weights. Siamese networks are popular among tasks that involve finding similarity or a relationship between two comparable objects. Two examples are signature verification [18] and assessing sentence similarity [69]. Another recent method, TFImpute [74], shares many similarities with FactorNet. Like FactorNet, TFImpute is intended to impute missing TF binding datasets, and it uses a CNN-RNN architecture, and a weight-sharing strategy to handle reverse complements. TFImpute is a sequence-only method and therefore more comparable to DeepSEA, DeepBind, and Basset. Unlike FactorNet, TFImpute does not directly accept cell type-specific data like DNase-seq as model inputs, hence its ability to generalize across cell types is questionable.

We submitted the FactorNet model to the ENCODE-DREAM *in vivo* Transcription Factor Binding Site Prediction Challenge, where it is among the top four ranked teams. All results discussed in this chapter are derived from data in the Challenge. The challenge delivers a crowdsourcing approach to figure out the optimal strategies for solving the problem of TF binding imputation. Another one of the goals of the challenge is to provide a platform for systematically and uniformly assessing the predictive models. This second goal was motivated by the multitude of examples in the literature where authors artificially generate questionable evaluation datasets. Often, these evaluation datasets are not representative of actual use cases, and may inflate results in favor of the methods developed by the respective authors. Similar issues were also highlighted in Chapters 3 and 4 of this thesis with respect to variant annotation.

5.2 Methods

5.2.1 ENCODE-DREAM Challenge dataset

The ENCODE-DREAM Challenge dataset is comprised of DNase-seq, ChIP-seq, and RNA-seq data from the ENCODE project or The Roadmap Epigenomics Project covering 14 cell types and 32 TFs. All annotations and pre-processing are based on hg19/GRCh37 release version of the human genome and GENCODE release 19 [42]. Data are restricted to chromosomes X and 1-22. Chromosomes 1, 8 and 21 are set aside exclusively for evaluation purposes and binding data were completely absent for these three chromosomes during the challenge. TF binding labels are provided at a 200 bp resolution. Specifically, the genome is segmented into 200 bp bins sliding every 50 bp. Each bin is labeled as bound (B), unbound (U) or ambiguously bound (A) depending on the majority label of all nucleotides in the bin. Ambiguous bins overlap peaks that fail to pass the irreproducible discovery rate (IDR)

threshold of 5% and are excluded from evaluation. A more complete description of the dataset, including pre-processing details such as peak calling, can be found in the ENCODE-DREAM Challenge overview paper (in preparation as of the writing of this thesis) and website (https://www.synapse.org/ENCODE).

5.2.2 Evaluation

The TF binding prediction problem is evaluated as a two-class binary classification task. For each test TF/cell type pair, the following performance measures are computed:

1. **auROC**. The area under the receiver operating characteristic curve is a common metric for evaluating classification models. It is equal to the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one.

2. **auPR**. The area under the precision-recall curve is more appropriate in the scenario of few relevant items. As mentioned in the previous chapter, this measure is preferred over the auROC for TF binding prediction [77]. Unlike the auROC, the auPR does not take into account the number of true negatives called.

3. **Recall at fixed FDR**. The recall at a fixed false discovery rate represents a point on the precision-recall curve. Like the auPR metric, this metric is appropriate in the scenario of few relevant items. This metric is often used in applications such as fraud detection in which the goal may be to maximize the recall of true fraudsters while tolerating a given fraction of customers to falsely identify as fraudsters.

As illustrated in Figure 5.1, the FactorNet Siamese architecture operates on both the forward and reverse complement sequences to ensure that both strands return the same outputs during both training and prediction. Although a TF might only physically bind to one

strand, this information cannot usually be inferred directly from the peak data. Thus, the same set of labels are assigned to both strands in the evaluation step.

5.2.3 Features and data pre-processing

FactorNet works directly with standard genomic file formats and requires relatively little pre-processing. BED files provide the locations of reference TF binding sites and bigWig files provide dense, continuous data at single-nucleotide resolution. bigWig values are included as extra rows that are appended to the 4-row one hot input DNA binary matrix. FactorNet can accept a variable number of bigWig files as input features, and we found the following signals to be the most informative for prediction:

1. **DNase I cleavage**. For each cell type, reads from all DNase-seq replicates were trimmed down to first nucleotide on the 5' end, pooled and normalized to 1x coverage using deepTools [80].

2. **35 bp mapability uniqueness**. This track quantifies the uniqueness of a 35 bp subsequence on the positive strand starting at a particular base, which is important for distinguishing where in the genome DNase I cuts can be detected. Scores are between 0 and 1, with 1 representing a completely unique sequence and 0 representing a sequence that occurs more than 4 times in the genome. Otherwise, scores between 0 and 1 indicate the inverse of the number of occurrences of that subsequence in the genome. It is available from the UCSC genome browser under the table wgEncodeDukeMapabilityUniqueness35bp.

In addition to sequential features, FactorNet also accepts non-sequential metadata features. At the cell type level, we applied principal component analysis to the inverse hyperbolic sine transformed gene expression levels and extracted the top 8 principal components. Gene

expression levels are measured as the average of the fragments per kilobase per million for each gene transcript. At the bin level, we included Boolean features that indicate whether gene annotations (coding sequence, intron, 5' untranslated region, 3' untranslated region, and promoter) and CpG islands [35] overlap a given bin. We define a promoter to be the region up to 300 bps upstream and 100 bps downstream from any transcription start site. To incorporate these metadata features as inputs to the model, we append the values to the dense layer of the neural network and insert another dense layer containing the same number of ReLU neurons between the new merged layer and the sigmoid layer (Fig. 5.1).

5.2.4 Training

Our implementation is written in Python, utilizing the Keras 1.2.2 library [24] with the Theano 0.9.0 [11, 14] backend. We used an NVIDIA Titan X Pascal GPU for training.

FactorNet supports single-task and multi-task training. Both types of neural network models are trained using the Adam algorithm [54] with a minibatch size of 100 to minimize the mean binary cross entropy loss function (Eqn. 4.2) on the training set. Adam (short for Adaptive Moment Estimation) is an update to the RMSProp optimizer, which we described in the previous chapter. It uses running averages of both the gradients and the second moments of the gradients. Given parameters θ_n and a loss function L_n, where n indexes the current training iteration (indexed at 1), Adam's parameter update is given by:

$$
\begin{aligned}
m_{n+1} &\coloneqq \alpha m_n + (1-\alpha)\nabla_\theta L_n && (1)\\
v_{n+1} &\coloneqq \beta v_n + (1-\beta)(\nabla_\theta L_n)^2 && (2)\\
\hat{m} &\coloneqq \tfrac{m_{n+1}}{1-\alpha^n} && (3)\\
\hat{v} &\coloneqq \tfrac{v_{n+1}}{1-\beta^n} && (4)\\
\theta_{n+1} &\coloneqq \theta_n - \gamma \tfrac{\hat{m}}{\sqrt{\hat{v}}+\epsilon}\nabla_\theta L_n && (5)
\end{aligned}
$$

where γ is the learning rate, ϵ is a "fuzz factor" to prevent division by zero, and α and β are the "forgetting factors" for gradients and second moments of gradients, respectively. We set

these values to 0.001, 10^{-8}, 0.9, and 0.999 respectively.

One or more chromosomes are set aside as a validation set. Validation loss is evaluated at the end of each training epoch and the best model weights according to the validation loss are saved. Training sequences of constant length centered on each bin are efficiently streamed from the hard drive in parallel to the model training. Random spatial translations are applied in the streaming step as a form of data augmentation. Each epoch, an equal number of positive and negative bins are randomly sampled and streamed for training, but this ratio is an adjustable hyperparameter (see Tab. for a detailed explanation of all hyperparameters). In the case of multi-task training, a bin is considered positive if it is confidently bound to at least one TF. Bins that overlap a blacklisted region [29] are automatically labeled negative and excluded from training. We also include dropout [94] to reduce overfitting.

Single-task training

Data from multiple cell types can be leveraged for single-task training by treating bins from all cell types as individually and identically distributed (i.i.d.) records. To make single-task training run efficiently, one bin is allotted per positive peak and these positive bins are included at most once per epoch for training. Each epoch, negative bins are also drawn randomly without replacement from the training chromosomes. For example, if we were to train on a single cell type that has 10,000 peaks for a particular TF, then we may train on 10,000 positive bins and 10,000 negative bins each epoch. Ambiguously bound bins are excluded from training.

Multi-task training

FactorNet can only perform multi-task training when training on data from a single cell type due to the variation of available binding data for the cell types. For example, the ENCODE-

DREAM Challenge provides reference binding data for 15 TFs for GM12878 and 16 TFs for HeLa-S3, but only 8 TFs are shared between the two cell types. Unlike the single-task models, which ignore ambiguous bins during training, the multi-task models assign negative labels to the ambiguous bins because of the frequent overlap of confidently and ambiguously bound regions. Compared to single-task training, multi-task training takes considerably longer to complete due to the larger number of positive bins. At the start of training, positive bins are identified by first segmenting the genome into 200 bins sliding every 50 bp and discarding all bins that fail to overlap at least one confidently bound TF site. Each epoch, negative bins are drawn randomly with replacement from the training chromosomes.

5.2.5 Bagging

Ensembling is a common strategy for improving classification performance. At the time of the Challenge, we implemented a simple ensembling strategy commonly called "bagging submissions", which involves averaging predictions from two or more models. Instead of averaging prediction probabilities directly, we first convert the scores to ranks, and then average these ranks. Rank averaging is more appropriate than direct averaging if predictors are not evenly calibrated between 0 and 1, which is often the case with the FactorNet models.

5.3 Results

5.3.1 Predictive performance varies across transcription factors

Table 5.1 shows a partial summary of FactorNet cross-cell type predictive performances on a variety of cell type and TF combinations as of the conclusion of the ENCODE-DREAM Challenge. Final rankings in the Challenge are based on performances over 13 TF/cell

Table 5.1: **Partial summary of FactorNet cross-cell type predictive performances on the ENCODE-DREAM Challenge data.** Each final ranking TF/cell type pair is demarcated with a *. For each final ranking TF/cell type pair, we provide, in parentheses, performance scores based on the evaluation pair's original ChIP-seq fold change signal.

Factor	**Cell type**	**auROC**	**auPR**	**Recall at 50% FDR**
CTCF*	iPSC	0.9966 (0.9998)	0.8608 (0.9794)	0.9142 (0.9941)
CTCF	GM12878	0.9968	0.8451	0.8777
CTCF*	PC-3	0.9862 (0.9942)	0.7827 (0.8893)	0.7948 (0.9272)
ZNF143	K562	0.9884	0.6957	0.7303
MAX	MCF-7	0.9956	0.6624	0.8290
MAX*	liver	0.9882 (0.9732)	0.4222 (0.6045)	0.3706 (0.6253)
EGR1	K562	0.9937	0.6522	0.7312
EGR1*	liver	0.9856 (0.9741)	0.3172 (0.5306)	0.2164 (0.5257)
HNF4A*	liver	0.9785 (0.9956)	0.6188 (0.8781)	0.6467 (0.9291)
MAFK	K562	0.9946	0.6176	0.6710
MAFK	MCF-7	0.9906	0.5241	0.5391
GABPA	K562	0.9957	0.6125	0.6299
GABPA*	liver	0.9860 (0.9581)	0.4416 (0.5197)	0.3550 (0.5202)
YY1	K562	0.9945	0.6078	0.7393
TAF1	HepG2	0.9930	0.5956	0.6961
TAF1*	liver	0.9892 (0.9657)	0.4283 (0.4795)	0.4039 (0.4766)
E2F6	K562	0.9885	0.5619	0.6455
REST	K562	0.9958	0.5239	0.5748
REST*	liver	0.9800 (0.9692)	0.4122 (0.5596)	0.4065 (0.5945)
FOXA1*	liver	0.9862 (0.9813)	0.4922 (0.6546)	0.4889 (0.6728)
FOXA1	MCF-7	0.9638	0.4487	0.4613
JUND	H1-hESC	0.9948	0.4098	0.3141
JUND*	liver	0.9765 (0.9825)	0.2649 (0.6921)	0.1719 (0.7223)
TCF12	K562	0.9801	0.3901	0.3487
STAT3	GM12878	0.9975	0.3774	0.3074
NANOG*	iPSC	0.9885 (0.9876)	0.3539 (0.6421)	0.3118 (0.6680)
CREB1	MCF-7	0.9281	0.3105	0.2990
E2F1*	K562	0.9574 (0.9888)	0.2406 (0.6428)	0.0000 (0.6573)
FOXA2*	liver	0.9773 (0.9932)	0.2172 (0.7920)	0.0231 (0.8278)

type pairs. A score combining several primary performance measures is computed for each pair. In addition to the 13 TF/cell type pairs for final rankings, there are 28 TF/cell type "leaderboard" pairs. Competitors can compare performances and receive live updating of their scores for the leaderboard TF/cell type pairs. Scores for the 13 final ranking TF/cell type pairs were not available until the conclusion of the challenge. Our model achieved first

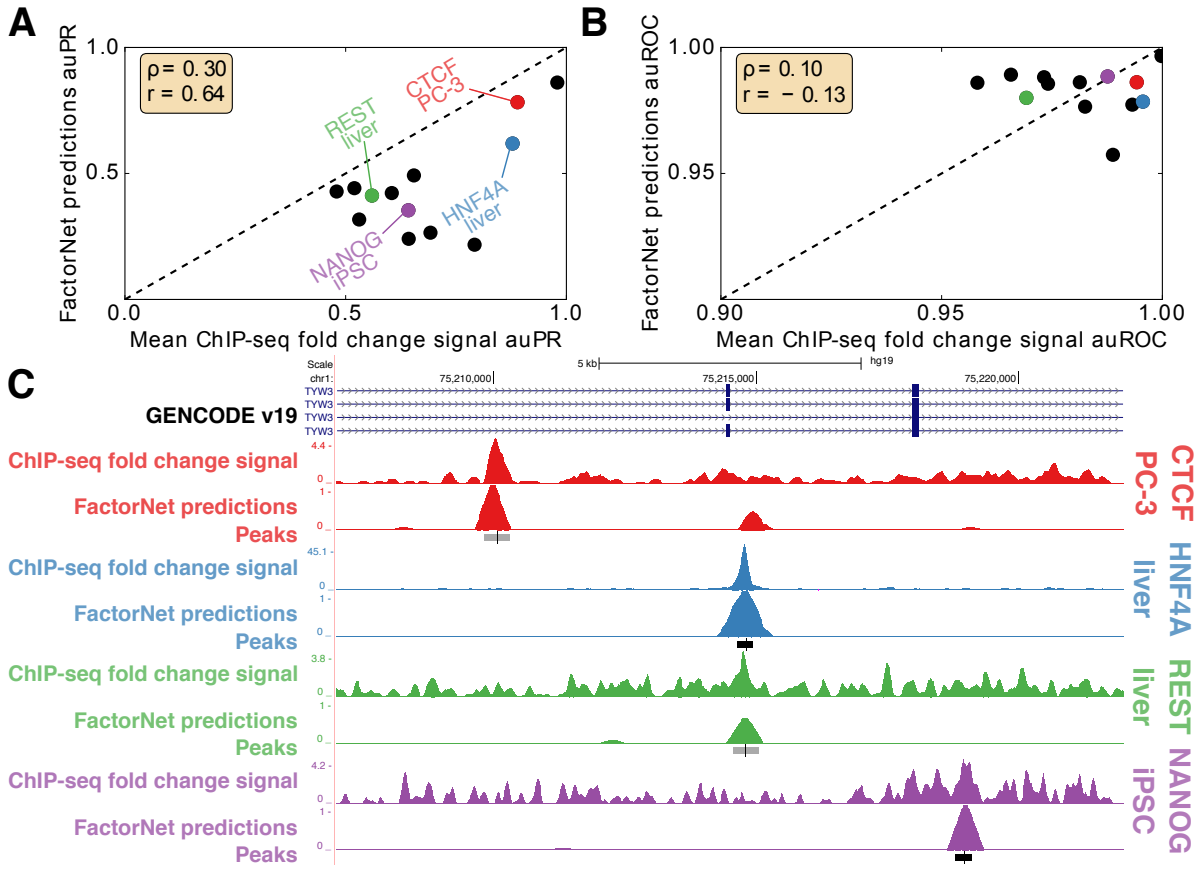


Figure 5.2: **Predictive performance and ChIP-seq signal varies across TF/cell-type pairs.** Scatterplots compare *(A)* auPR and *(B)* auROC scores between FactorNet predictions and mean ChIP-seq fold change signal. Each marker corresponds to one of the 13 final ranking TF/cell type pairs. Spearman (ρ) and Pearson (r) correlations are displayed in each plot. *(C)* Genome browser [51] screenshot displays the ChIP-seq fold change signal, FactorNet predictions, and peak calls for four TF/cell type pairs in the TYW3 locus. Confidently bound regions are more heavily shaded than ambiguously bound regions.

place on six of the 13 TF/cell type final ranking pairs, the most of any team.

FactorNet typically achieves auROC scores above 97% for most of the TF/cell type pairs, reaching as low as 92.8% for CREB1/MCF-7. auPR scores, in contrast, display a wider range of values, reaching as low as 21.7% for FOXA1/liver and 87.8% for CTCF/iPSC. For some TFs, such as CTCF and ZNF143, the predictions are already accurate enough to be considered useful. Much of the variation in auPR scores can be attributed to noise in the ChIP-seq signal used to generate the evaluation labels, which we demonstrate by building classifiers based on taking the mean in a 200 bp window of the ChIP-seq fold change

signal with respect to input control. Peak calls are derived from the SPP algorithm [53], which uses the fold-change signal and peak shape to score and rank peaks. An additional processing step scores peaks according to an IDR value, which is a measure of consistency between replicate experiments. Bins are labeled positive if they overlap a peak that meets the IDR threshold of 5%. The IDR scores are not always monotonically associated with the fold-changes. Nevertheless, we expect that performance scores from the fold-change signal classifiers should serve as overly optimistic upper bounds for benchmarking. Commensurate with these expectations, the auPR scores of the FactorNet models are less than, but positively correlative with, the respective auPR scores of the ChIP-seq fold-change signal classifiers (Figure 5.2A). Interestingly, this pattern does not extend to the auROC scores, and in more than half of the cases the FactorNet auROC scores are greater (Figure 5.2B). These results are consistent with our findings in Chapter 4 which showed the auROC can be unreliable and overly optimistic in an imbalanced class setting [87], which is a common occurrence in genomic applications [77], motivating the use of alternative measures like the auPR that ignore the overly abundant true negatives.

We can also visualize the FactorNet predictions as genomic signals that can be viewed alongside the ChIP-seq signals and peak calls (Figure 5.2C). Higher FactorNet prediction values tend to coalesce around called peaks, forming peak-like shapes in the prediction signal that resemble the signal peaks in the original ChIP-seq signal. The visualized signals also demonstrate the differences in signal noise across the ChIP-seq datasets. The NANOG/iPSC ChIP-seq dataset, for example, displays a large amount of signal outside of peak regions, unlike the HNF4A/liver ChIP-seq dataset which has most of its signal focused in peak regions.

The ENCODE-DREAM challenge data, documentation, and results can be found on the Challenge homepage: https://www.synapse.org/ENCODE

Interpreting neural network models

Using the same heuristic from DeepBind [3] and DanQ [77] that we used in Chapter 4, we visualized several kernels from a HepG2 multi-task model as sequence logos by aggregating subsequences that activate the kernels (Figure 5.3A). The kernels significantly match motifs associated with the target TFs. Furthermore, the aggregated DNase I signals also inform us of the unique "footprint" signatures the models use to identify true binding sites at single-nucleotide resolution. After visualizing and aligning all the kernels, we confirmed that the model learned a variety of motifs (Figure 5.3B). A minority of kernels display very little sequence specificity while recognizing regions of high chromatin accessibility (Figure 5.3C).

Saliency maps are another common technique of visualizing neural network models [92]. To generate a saliency map, we compute the gradient of the output category with respect to the input sequence. By visualizing the saliency maps of a genomic sequence, we can identify the parts of the sequence the neural network finds most relevant for predicting binding, which we interpret as sites of TF binding at single-nucleotide resolution. Using a liver HNF4A peak sequence and HNF4A predictor model as an example, the saliency map highlights a subsequence overlapping the summit that strongly matches the known canonical HNF4A motif, as well as two putative binding sites upstream of the summit on the reverse complement (Figure 5.3D).

5.3.2 Data variation influences predictive performance

In the cases for which two or more testing cell types are available for the same TF, we also observe some rather large disparities in performance. With the exception of FOXA1, FactorNet consistently performs poorer for liver than for other cell types, the difference in auPR reaching as much as 33.5% in the case of EGR1 (Table 5.1). Variation in data quality across cell type-specific datasets may partially explain these performance differences. The

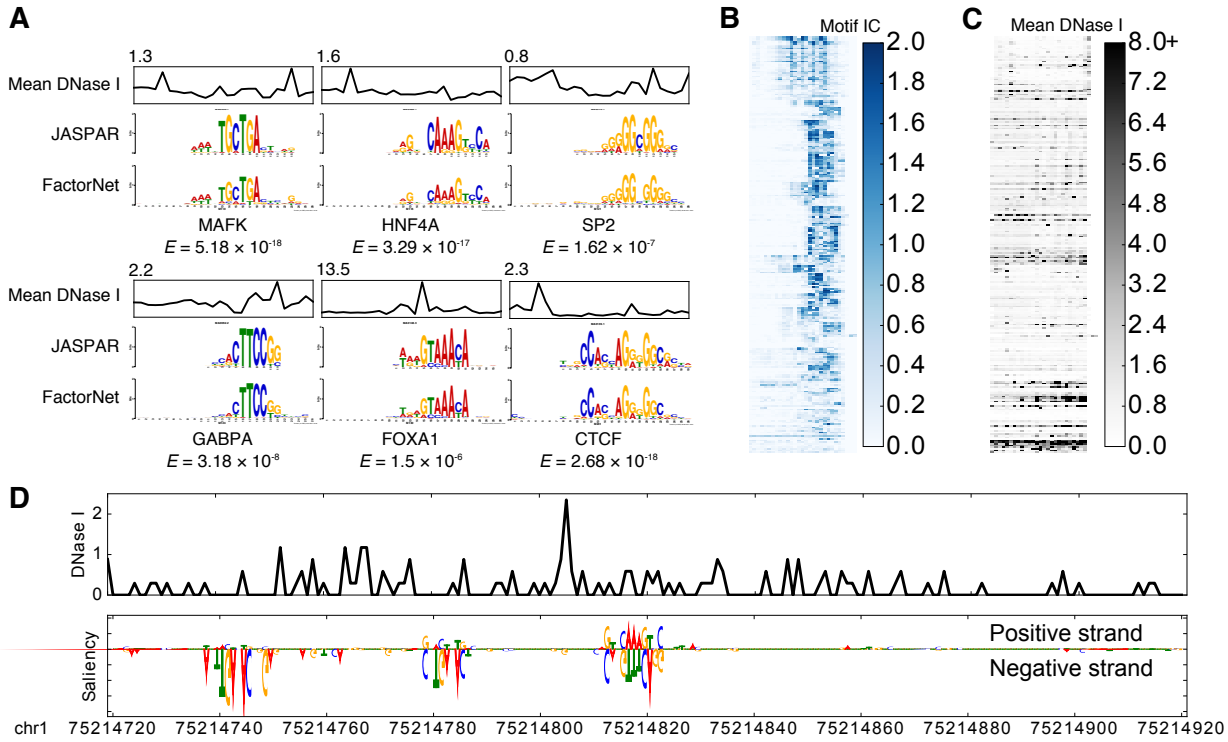


Figure 5.3: **Visually interpreting FactorNet models.** *(A)* Network kernels from a HepG2 multi-task FactorNet model are converted to sequence logos and aligned with motifs from JASPAR [64] using TOMTOM [41]. Mean normalized DNase I cleavage signals and their maximum values are displayed above the aligned logos. E-values measure similarity between query and target motifs, corrected for multiple hypothesis testing. All kernels are converted to sequence logos and aligned with RSAT [66]. The heatmaps are ordered by this alignment and colored according to the motif information content (IC) *(B)* or mean DNase I cleavage signal *(C)* at each nucleotide position. *(D)* Normalized liver DNase I cleavage signal and saliency maps of aligned stranded sequences centered on the summit of a liver HNF4A peak in the TYW3 locus (Figure 5.2C). Negative gradients are converted to zeros. We visualized saliency maps with the DeepLIFT visualizer [91]

DNase-seq data, which is arguably the most informative cell type-specific feature for binding prediction, widely varies in terms of sequencing depth and signal-to-noise ratio (SNR) across the cell types, which we measure as the fraction of reads that fall into conservative peaks (FRiP) (Figure 5.4A). Notably, liver displays the lowest SNR with a FRiP score of 0.05, which is consistent with its status as a primary tissue; all other cell types are cultured cell lines.

To further scrutinize the effect data variation has on performance, we trained several Fac-

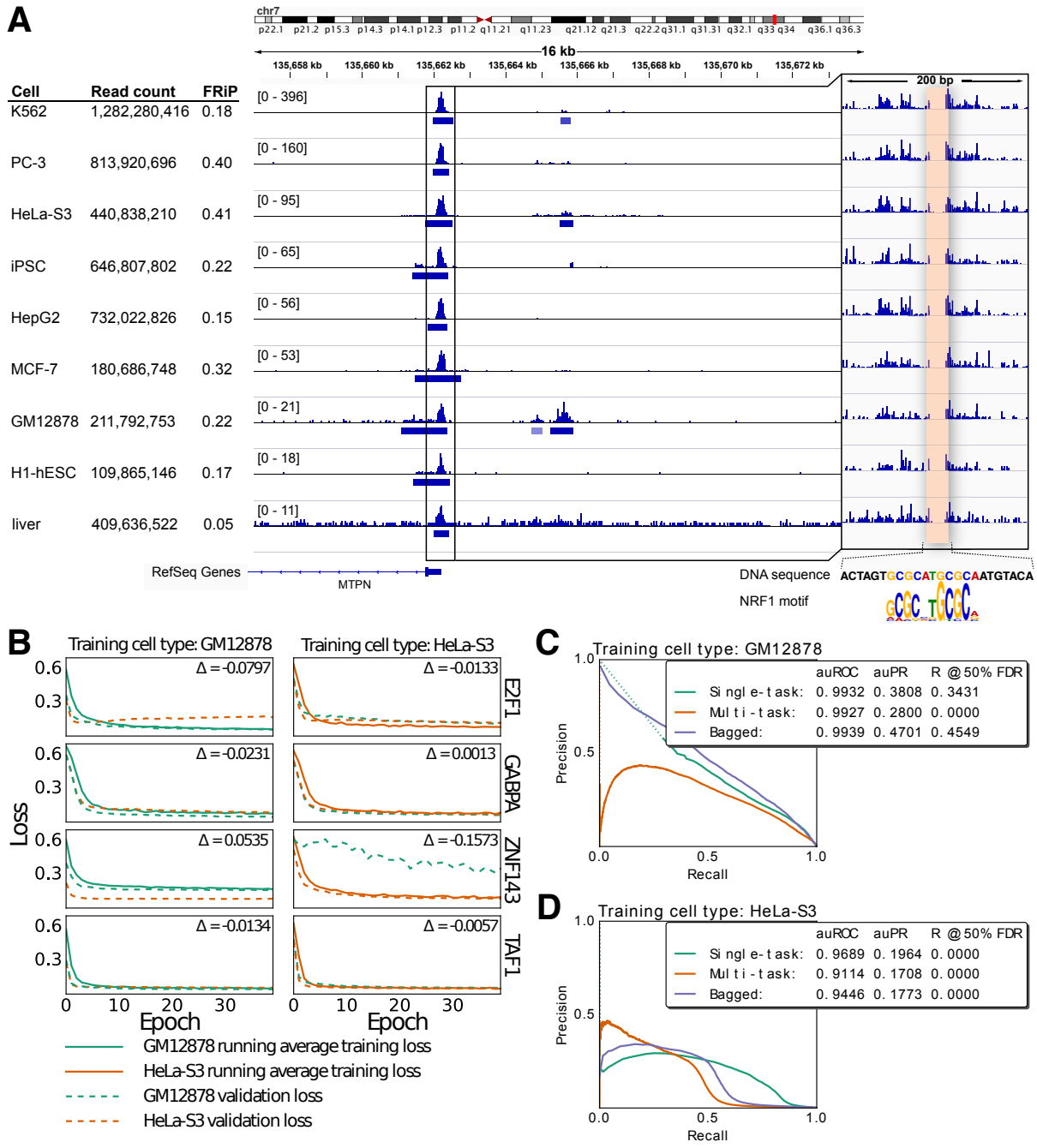


Figure 5.4: **Cell type-specific dataset variation influence cross-cell type performance.** *(A)* IGV [99] screenshot of pooled DNase I cleavage signal and conservative DNase-seq peaks. The inset is a magnified view of an NRF1 binding site. *(B)* Learning curves of E2F1 single-task models trained on a cell type. Difference between the smallest within- and cross-cell type validation losses are displayed in each plot. *(C and D)* PR curves of models trained exclusively on a cell type evaluated on E2F1/K562. Dotted lines are points of discontinuity. We generated single-task scores by bagging scores from two single-task models initialized differently. Final bagged models ensemble single- and multi-task models.

torNet single-task models and plotted the learning curves to monitor for overfitting (Figure 5.4B). Learning curves trace the predictive performance of neural networks on training and validation sets. They are useful for identifying signs of overfitting, a common problem in machine learning. These learning curves focus on the GM12878 and HeLa-S3 cell types, using one cell type for training and the other as a validation set. We selected these two cell types because they are the only two reference cell types for E2F1, which FactorNet performed particularly poor on. In addition, the HeLa-S3 DNase-seq data read count and FRiP score are both almost twice that of the read count and FRiP score for the GM12878 DNase-seq data.

From the learning curves of the E2F1 model trained on GM12878, we observe evidence of overfitting. The HeLa-S3 cross-cell type validation loss reaches a minimum value within four training epochs, after which it increases until it reaches a steady state value. In contrast, the GM12878 within-cell type validation loss steadily decreases past the first four epochs and remains much smaller than the HeLa-S3 validation loss throughout training. At first, we speculated the gap to be caused by the differences in the cell type DNase-seq data; however, based on the learning curves for other TFs, this may not necessarily be the sole reason. In the cases of GABPA and TAF1, the differences in validation losses is much smaller. One possible explanation for these results is the differences in the ChIP-seq protocols between the GM12878 and HeLa-S2 datasets. Unlike the other three TFs, the GM12878 and HeLa-S3 E2F1 ChIP-seq datasets were generated using two different antibodies: ENCAB037OHX and ENCAB000AFU, respectively. Both ZNF143 ChIP-seq datasets were generated using the same antibody (ENCAB000AMR), but the model trained on HeLa-S3 displays an unusually high validation loss difference. We speculate this is because the GM12878 ZNF143 ChIP-seq dataset was generated using both single-end 36 bp and paired-end 100 bp reads while the HeLa-S3 ZNF143 ChIP-seq dataset was generated using only single-end 36 bp reads. Given that paired-end 100 bp reads can map to genomic regions that are unmapable for the shorter 36 bp reads, we suspect that differences in read types can introduce significant

dataset-specific artifacts.

Given the differences in the GM12878 and HeLa-S3 E2F1 ChiP-seq datasets resulting from the use of different antibodies, we investigated whether a model exclusively trained on one cell type could improve our predictive performance for the K562/E2F1 testing set. To do so, we retrained single- and multi-task models exclusively on either GM12878 or HeLa-S3 and evaluated cross-cell type binding performance on the E2F1/K562 testing set. In contrast, the E2F1 model used at the conclusion of the Challenge was trained on data from both reference cell types. The K562 E2F1 ChIP-seq dataset was generated using the antibodies ENCAB037OHX and ENCAB851KCY, the former of which was also used for GM12878. Hence, we expect that the GM12878 model would be a better predictor for K562 E2F1 binding sites than the other two models, which we find to indeed be the case (Figure 5.4C and 5.4D). Although we managed to improve upon our previous E2F1 model, the cross-cell type performance for E2F1 is still inadequate, especially compared to TFs like CTCF. Predicting binding for TFs in the E2F family is notoriously difficult because members of this protein family share almost identical binding motifs, which in turn makes distinguishing between multiple members of the same family difficult. For TFs that are part of a large family sharing similar sequence binding preference, we conjecture that performance will be limited regardless of the choice of cell type or antibody.

5.3.3 Comparing single- and multi-task training

Although a thorough comparison between single- and multi-task training is beyond the scope of this paper, our results on E2F1 show that single- and multi-task models can differ in performance. Specifically, the cross-cell type auPR of the single-task GM12878 model is more than 10% greater than that of its respective multi-task model (Figure 5.4C and Figure 5.4D). To the best of our knowledge, the cross-cell type performance of each training method

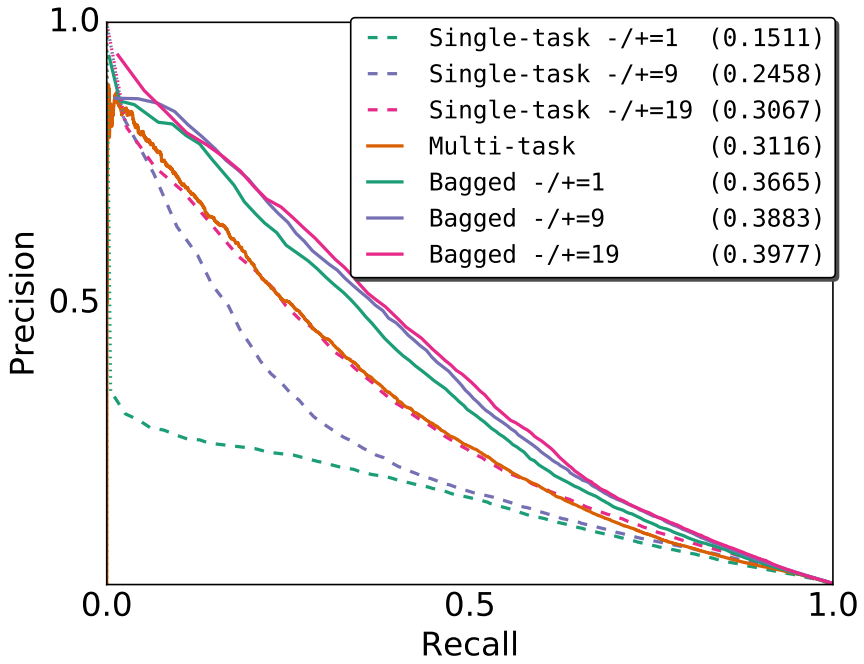


Figure 5.5: **Comparison of single- and multi-task training.** Cross-cell type precision-recall curves of single-task and multi-task NANOG binding prediction models trained on H1-hESC and evaluated on iPSC. Model weights were selected based on the within-cell type validation loss on chr11. We generated single-task scores by bagging scores from two single-task models initialized differently. The three single-task models differ in the ratio of negative-to-positive bins per training epoch. The bagged models are the rank average scores from the multi-task model and one of the three single-task models. auPR scores are in parentheses. Both training and testing ChiP-seq datasets use the ENCAB000AIX antibody.

depends on the TF/cell type pair. For example, when we retrained single-task and multi-task models for NANOG using H1-hESC as a reference cell type and evaluated the models on iPSC, the multi-task model's auPR score is over 16% greater than that of the single-task model (Figure 5.5).

While we initially assumed that the multi-task training confers an advantage by introducing additional information through the multiple labels, at least in the case of NANOG, there are too many conflating variables to immediately conclude this. One of these conflating variables is the differences in training data between single-task and multi-task models. In our current framework, the multi-task training contains significantly more negative bins per training epoch than the single-task training does to balance the positive bins from multiple

ChIP-seq datasets. By increasing the ratio of negative to positive samples per epoch for single-task training, we can close the gap between the two training methods in terms of the auPR score, demonstrating that the selection of negative bins affects predictive accuracy. Moreover, the single-task models each use 654,657 weights and require 30 seconds-5 minutes per training epoch whereas the multi-task models each use 5.4 million weights and require 2-3 hours per training epoch, making the former significantly more efficient than the latter. Regardless of whether single- or multi-task training is advantageous, ensembling predictions from both model types can yield significant improvements in performance. It should be noted this pattern did not hold true for the case of training on HeLa-S3 data and evaluating on E2F1/K562 (Figure 5.4D), and we speculate that the difference in antibodies may explain this discrepancy.

5.4 Discussion

In this work, we introduced FactorNet - an open source package to apply stacked convolutional and recurrent neural networks for predicting TF binding across cell types. While RNNs are computationally expensive to train, especially compared to CNNs, FactorNet incorporates several heuristics to significantly speed up model training and improve predictive performance. Using data from the ENCODE-DREAM Challenge, we demonstrated how our model can effectively integrate cell type-specific data such as DNase-seq to generalize TF binding from reference cell types to testing cell types. As of the conclusion of the Challenge, FactorNet is one of the top performing binding prediction models.

Through our post-Challenge analyses, we gained insights into the variables that affect predictive performance, allowing us to propose strategies for improving the model. First, we observed that the predictive performance widely varied over all TF/cell type pairs, especially in terms of the auPR metric. By leveraging the original ChIP-seq fold change signal,

we established upper bounds for the auPR metric for each final ranking TF/cell type pair. These bounds also correlate positively with auPR scores from FactorNet predictions, showing that a large amount of the variation in predictive performance can be attributed to the noise in the original ChIP-seq signal (Figure 5.2A). We expect that predictive performance for many TF/cell type pairs can be improved by redoing experiments with higher quality antibodies. Alternatively, ChIP-exo, a modification of ChIP-seq that uses exonucleases to degrade contaminating non-protein-bound DNA fragments [82], may improve the quality of ChIP signals. Next, we investigated the variation in the DNase-seq datasets. We found that the DNase-seq datasets greatly differ in terms of sequencing depth and SNR (Figure 5.4A). While we do correct for the variation in sequencing depth by normalizing the cleavage signals to 1x coverage, we do not correct for the variation in the SNR. The performance lost is most staggering for the liver cell type, which has the DNase-seq dataset with the lowest SNR. However, differences in DNase-seq SNR do not fully account for differences in predictive performance. By studying several within- and cross-cell type validation curves, we also concluded that differences in antibodies and read lengths can introduce significant dataset-specific biases (Figure 5.4B). Accordingly, we can improve performance by omitting less compatible cell type datasets (Figure 5.4C-D).

We also compared single- and multi-task training frameworks. Several previous studies have proposed that multi-task training can aid performance, but our results do not entirely favor one method over the other. For the K562/E2F1 cross-cell type testing set, the GM12878 single-task model outperformed GM12878 multi-task model (Figure 5.4C); however, for the NANOG/iPSC cross-cell type testing set, the H1-hESC multi-task model outperformed the H1-hESC single-task model (Figure 5.5). In the latter case, the performance gap can be narrowed by changing the proportion of negative to positive training samples in the single-task framework, suggesting that any additional gain granted by the multiple labels is eclipsed by the choice of negative sets. Nevertheless, ensembling single- and multi-task models together appears to be an effective method of improving predictive performance, at least if antibodies

and read lengths are kept consistent.

Another avenue we can explore for improving the model is hyperparameter tuning. We selected the hyperparameters for the models in this work arbitrarily for demonstration and uniformity purposes (Table S5.1-S5.3). Although we have not yet implemented them, distributed computing hyperparameter tuning algorithms [15] can systematize hyperparameter selection and improve performance.

One of the chief criticisms of neural networks is that they are "black box" models. While neural networks can achieve great performances in predictive tasks, the exact reasons for why this is the case is not always entirely clear. In contrast to these criticisms, we can visualize and interpret aspects of the FactorNet model. By converting network kernels to motifs, we show that FactorNet can recover motifs that are known to contribute to binding (Figure 5.3A). DNase I footprint patterns help discriminate true binding sites from putative sites that simply match a motif. Previous TF binding prediction methods, such as Centipede, require users to supply motifs. FactorNet relaxes this strong assumption and essentially performs *de novo* motif discovery during the learning process to identify the sequence patterns that are most useful for binding prediction. Saliency maps can also help elucidate the complex regulatory grammar that govern TF binding by visualizing the spatial positions and orientations of multiple binding sites that work together to recruit TFs (Figure 5.3D).

Our adherence to standardized file formats also makes FactorNet robust. For example, FactorNet can readily accept other genomic signals that were not included as part of the Challenge but are likely relevant to TF binding prediction, such as conservation and methylation. Along these same lines, if we were to refine our pre-processing strategies for the DNase-seq data, we can easily incorporate these improved features into our model as long as the data are available as bigWig files [52]. Other sources of open chromatin information, such as ATAC-seq [20] and FAIRE-seq [37], can also be used to replace or complement the existing DNase-seq data. In addition, FactorNet is not necessarily limited to only TF binding

predictions. If desired, users can provide the BED files of positive intervals to train predictive models for other markers, such as histone modifications. As more epigenetic datasets are constantly added to data repositories, FactorNet is already in a prime position to integrate both new and existing datasets.

In conclusion, FactorNet is a very flexible framework that lends itself to a variety of future research avenues. The techniques that we introduced in this paper will also be useful for the field of machine learning, especially since neural network models are becoming increasingly popular in genomics. Some of the design elements of FactorNet were motivated by the specific properties inherent in the structure of the data. Many of these properties are shared in data found in other applications of machine learning. For example, the directional nature and modularity of DNA sequences prompted us to search for a model that can discover local patterns and long-range interactions in sequences, which led us to ultimately select a hybrid neural network architecture that includes convolution and bidirectional recurrence. Natural language processing problems, such as topic modeling and sentiment analysis, can also benefit from such an architecture since language grammar is directional and modular. Another unique aspect of the data that guided our design is the double strandedness of DNA, which prompted us to adopt a Siamese architecture to handle pairs of input sequences. Protein-protein interaction prediction also involves sequence pairs and would likely benefit from a similar framework. Our heuristics for reducing training time and computational overhead will serve as useful guidelines for other applications involving large imbalanced data, especially if recurrent models are utilized. We therefore expect that FactorNet will be of value to a wide variety of fields.

5.5 Software availability

Source code is available at the github repository http://github.com/uci-cbcl/FactorNet under the MIT license. In addition to the source code, the github repository contains all models and data used for the ENCODE-DREAM challenge.

5.6 Supporting Information

Table S5.1: **Summary and description of the hyperparameters used for the single-task models in Figure 5.4B.**

Hyperparameter	Value	Description
-v validchroms	chr3 chr5 chr7 chr10 chr12 chr14 chr16 chr18 chr20 chrX	Sequences on these chromosomes are set aside for validation.
-e epochs	200 (ZNF143, TAF1), 300 (E2F1, GABPA)	Max number of epochs to train before training ends.
-ep patience	200 (ZNF143, TAF1), 300 (E2F1, GABPA)	Number of epochs with no improvement in the validation loss.
-lr learningrate	0.00001	Learning rate for the Adam optimizer. We decreased it from the default value of 0.001 to smooth the learning curves.
-n negatives	1	Number of negative bins to sample per positive bin per epoch.
-L seqlen	1000	Length, in bps, of input sequences to the model.
-w motifwidth	26	Width, in bps, of the convolutional kernels.
-k kernels	32	Number of kernels/motifs in the model.
-r recurrent	32	Number of recurrent units (in one direction) in the model.
-d dense	128	Number of units in the dense layer in the model.
-p dropout	0.5	Dropout rate between the recurrent and dense layers. Also the dropout rate between the dense and sigmoid layers.
-m metaflag	False	Flag for including cell type-specific metadata features (usually gene expression).
-g gencodeflag	True	Flag for including CpG island and gene annotations.
-mo motifflag	True	Flag for initializing two of the kernels to the PWM of the canonical motif (forward and reverse complement).
-s randomseed	Varies	Random seed for reproducibility.

Table S5.2: **Hyperparameters used for the multi-task models in Figures 5.3-5.5.** Unspecified values should be assumed to be the same as those found in Table S5.1.

Hyperparameter	Value	Notes
-v validchroms	chr11	
-e epochs	20	Fewer epochs needed for multi-task training due to the large number of training bins.
-ep patience	20	
-lr learningrate	0.001	Default value of 0.001 is sufficient for most applications.
-n negatives	1	
-g gencodeflag	False	Multi-task training does not currently incorporate any metadata features.
-mo motifflag	False	

Table S5.3: **Hyperparameters used for the single-task models in Figures 5.4C-D and 5.5.** Unspecified values should be assumed to be the same as those found in Table S5.1.

Hyperparameter	Value	Notes
-v validchroms	chr11	
-e epochs	100	Need more epochs than multi-task training due to fewer positive bins.
-ep patience	20	
-lr learningrate	0.001	Default value of 0.001 is sufficient for most applications.
-n negatives	Varies (but usually 1)	In some cases, increasing this value from 1 improves cross-cell type auPR scores for single-task models.

Chapter 6

Conclusion

HTS technology has undoubtedly revolutionized both basic and translational biological research. The massive throughput of data delivered by HTS has encouraged a more "data-driven" approach to research, moving from experimental procedures of limited scope concerning single genes to comprehensive, simultaneous sweeps elucidating insights at an "-omics" level. This data-driven paradigm is reinforced by the growing availability of HTS datasets, such as those generated by ENCODE and Roadmap Epigenomics. If desired, researchers can design projects purely based around exploring the genomic data that are already publicly available. Furthermore, as new HTS-based experiments are developed, new computational models will have to be developed in tandem. These computational models must meet the challenging demand of being scalable while maintaining sufficient complexity.

In this thesis, we have presented four machine learning methods integrating HTS-derived data to address several open challenges in functional genomics. First, we described improvements in *de novo* motif discovery. Our solution improves upon MEME, a popular *de novo* motif discovery algorithm, by replacing the batch EM algorithm with an online EM algorithm. The resulting algorithm, EXTREME, scales linearly with the size of the input sequence dataset,

whereas MEME scales quadratically. EXTREME's scalability proves especially useful for large sequence datasets that are derived from ChIP-seq peaks and DNase-seq footprints, which can contain hundreds of thousands to millions of base pairs. We demonstrated in a ChIP-seq dataset and a DNase-seq dataset that EXTREME can efficiently discover both novel and known motifs, the latter of which can significantly resemble motifs derived from MEME. We also showed that using the entire dataset is necessary to discover infrequent motifs. These infrequent motifs are commonly overlooked when applying motif discovery to a small fraction of the sequences, a commonly used strategy to compensate for MEME's poor time complexity. Second, we showed how deep learning methods can be adapted to integrate a diverse set of features to address the problem of pathogenicity annotation. Specifically, we trained a three layer multilayer perceptron binary classifier on a dataset of $\tilde{3}0$ million genetic variants to discriminate between "deleterious" and "benign" variants. Our method, DANN, improves upon CADD, which uses a linear support vector machine to solve the same problem; however, the results raised several questions. Mainly, what is the proper definition of a deleterious variant and what is the proper way, if any, of evaluating annotation methods? Third, we introduced a stacked convolutional and recurrent hybrid neural network to predict chromatin markers directly from raw nucleotide sequences. Our method, DanQ, outperformed DeepSEA, a pure convolutional model, in this task based on two performance measures: the area under the receiver operating curve and the area under the precision-recall curve, the latter of which was more appropriate for evaluation in this case due to the class imbalance. Once again, we also addressed the problem of variant annotation, and although our method outperformed DeepSEA in this task as well, the results also raised questions about how to properly evaluate a variant annotation method. Finally, we extended the DanQ architecture to integrate raw nucleotide sequences and cell type-specific genomic signals to predict transcription factor binding across cell types. The resulting model, FactorNet, addresses a need to impute gaps in the ChIP-seq experiment matrix of ENCODE and Roadmap Epigenomics. We submitted our model to the ENCODE-DREAM *in vivo* Transcription Factor Binding

Site Prediction Challenge, where it was among one of the top performing models. One of the primary goals of the challenge was to provide a uniform and systematic platform to evaluate the different competing methods. We developed many novel heuristics to reduce training time, which is of high-priority for competitions with strict deadlines. One of these heuristics is making the model a Siamese network, which applies identical subnetworks containing shared weights to both the forward and reverse strand sequences. This strategy allows the model to efficiently process both strands, in parallel, ensuring that both strands return the same output. Previous methods, including DeepSEA and DanQ, have treated both strands as independent and identically distributed records, an incorrect assumption that doubles the training times.

In terms of existing and future computational needs in genomics research, our solutions have only scratched the surface. Nevertheless, our contributions serve as important stepping stones in the development of current and future bioinformatics tools by highlighting the need for scalability and complexity. Our implementations of these improvements are all available as open-source software.

Bibliography

[1] The cost of sequencing a human genome. `https://www.genome.gov/sequencingcosts/`. Accessed: 2017-04-09.

[2] 1000 Genomes Project Consortium, G. R. Abecasis, A. Auton, L. D. Brooks, M. A. DePristo, R. M. Durbin, R. E. Handsaker, H. M. Kang, G. T. Marth, and G. A. McVean. An integrated map of genetic variation from 1,092 human genomes. *Nature*, 491(7422):56–65, Nov 2012.

[3] B. Alipanahi, A. Delong, M. T. Weirauch, and B. J. Frey. Predicting the sequence specificities of dna- and rna-binding proteins by deep learning. *Nat Biotechnol*, 33(8):831–8, Aug 2015.

[4] C. Angermueller, H. J. Lee, W. Reik, and O. Stegle. Deepcpg: accurate prediction of single-cell dna methylation states using deep learning. *Genome biology*, 18(1):67, 2017.

[5] T. L. Bailey. DREME: motif discovery in transcription factor ChIP-seq data. *Bioinformatics*, 27(12):1653–9, Jun 2011.

[6] T. L. Bailey, M. Bodén, T. Whitington, and P. Machanick. The value of position-specific priors in motif discovery using MEME. *BMC Bioinformatics*, 11(1):179, 2010.

[7] T. L. Bailey and C. Elkan. Unsupervised learning of multiple motifs in biopolymers using expectation maximization. *Machine learning*, 21(1-2):51–80, 1995.

[8] T. L. Bailey and C. Elkan. The value of prior knowledge in discovering motifs with MEME. In *Ismb*, volume 3, pages 21–29, 1995.

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

[10] M. Baker. One-stop shop for disease genes. *Nature*, 491(7423):171, Nov 2012.

[11] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[12] M. A. Beer and S. Tavazoie. Predicting gene expression from sequence. *Cell*, 117(2):185–198, 2004.

[13] G. Benson. Tandem repeats finder: a program to analyze DNA sequences. *Nucleic Acids Res*, 27(2):573–80, Jan 1999.

[14] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: a cpu and gpu math expression compiler. In *Proceedings of the Python for scientific computing conference*, volume 4, page 3. Austin, TX, 2010.

[15] J. Bergstra, D. Yamins, and D. D. Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. *ICML (1)*, 28:115–123, 2013.

[16] E. Birney, J. A. Stamatoyannopoulos, A. Dutta, R. Guigó, T. R. Gingeras, E. H. Margulies, Z. Weng, M. Snyder, E. T. Dermitzakis, and R. E. *et. al*. Thurman. Identification and analysis of functional elements in 1% of the human genome by the ENCODE pilot project. *Nature*, 447(7146):799–816, Jun 2007.

[17] A. P. Boyle, L. Song, B.-K. Lee, D. London, D. Keefe, E. Birney, V. R. Iyer, G. E. Crawford, and T. S. Furey. High-resolution genome-wide in vivo footprinting of diverse transcription factors in human cells. *Genome research*, 21(3):456–464, 2011.

[18] J. Bromley, J. W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Säckinger, and R. Shah. Signature verification using a "siamese" time delay neural network. *IJPRAI*, 7(4):669–688, 1993.

[19] J. D. Buenrostro, P. G. Giresi, L. C. Zaba, H. Y. Chang, and W. J. Greenleaf. Transposition of native chromatin for fast and sensitive epigenomic profiling of open chromatin, dna-binding proteins and nucleosome position. *Nature methods*, 10(12):1213–1218, 2013.

[20] J. D. Buenrostro, B. Wu, H. Y. Chang, and W. J. Greenleaf. Atac-seq: A method for assaying chromatin accessibility genome-wide. *Current protocols in molecular biology*, pages 21–29, 2015.

[21] O. Cappé and E. Moulines. On-line expectation–maximization algorithm for latent data models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(3):593–613, 2009.

[22] C. Chen, B. Schmidt, L. Weiguo, and W. Müller-Wittig. Gpu-meme: Using graphics hardware to accelerate motif finding in dna sequences. In *IAPR International Conference on Pattern Recognition in Bioinformatics*, pages 448–459. Springer, 2008.

[23] Y. Chen, Y. Li, R. Narayan, A. Subramanian, and X. Xie. Gene expression inference with deep learning. *Bioinformatics*, Feb 2016.

[24] F. Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.

[25] G. M. Cooper, E. A. Stone, G. Asimenos, NISC Comparative Sequencing Program, E. D. Green, S. Batzoglou, and A. Sidow. Distribution and intensity of constraint in mammalian genomic sequence. *Genome Res*, 15(7):901–13, Jul 2005.

[26] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[27] Crawford, G. et al. Genome-wide mapping of dnase hypersensitive sites using massively parallel signature sequencing (mpss). *Genome Res*, 16(1):123–31, Jan 2006.

[28] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[29] ENCODE Project Consortium. An integrated encyclopedia of dna elements in the human genome. *Nature*, 489(7414):57–74, Sep 2012.

[30] J. Ernst and M. Kellis. Chromhmm: automating chromatin-state discovery and characterization. *Nature methods*, 9(3):215–216, 2012.

[31] J. Ernst and M. Kellis. Large-scale imputation of epigenomic datasets for systematic annotation of diverse human tissues. *Nat Biotechnol*, 33(4):364–76, Apr 2015.

[32] V. Franc and S. Sonnenburg. Optimized cutting plane algorithm for large-scale risk minimization. *The Journal of Machine Learning Research*, 10:2157–2192, 2009.

[33] W. Fu et al. Analysis of 6,515 exomes reveals the recent origin of most human protein-coding variants. *Nature*, 493(7431):216–20, Jan 2013.

[34] Y. Fu, Z. Liu, S. Lou, J. Bedford, X. J. Mu, K. Y. Yip, E. Khurana, and M. Gerstein. Funseq2: a framework for prioritizing noncoding regulatory variants in cancer. *Genome biology*, 15(10):480, 2014.

[35] M. Gardiner-Garden and M. Frommer. CpG islands in vertebrate genomes. *Journal of molecular biology*, 196(2):261–282, 1987.

[36] M. Ghandi, D. Lee, M. Mohammad-Noori, and M. A. Beer. Enhanced regulatory sequence prediction using gapped k-mer features. *PLoS Comput Biol*, 10(7):e1003711, Jul 2014.

[37] P. G. Giresi, J. Kim, R. M. McDaniell, V. R. Iyer, and J. D. Lieb. Faire (formaldehyde-assisted isolation of regulatory elements) isolates active regulatory elements from human chromatin. *Genome research*, 17(6):877–885, 2007.

[38] A. Graves, N. Jaitly, and A.-R. Mohamed. Hybrid speech recognition with deep bidirectional lstm. In *Automatic Speech Recognition and Understanding, 2013 IEEE Workshop on*, pages 273–278, Dec 2013.

[39] A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610, 2005.

[40] B. Gulko, M. J. Hubisz, I. Gronau, and A. Siepel. A method for calculating probabilities of fitness consequences for point mutations across the human genome. *Nature genetics*, 47(3):276–283, 2015.

[41] S. Gupta, J. A. Stamatoyannopoulos, T. L. Bailey, and W. S. Noble. Quantifying similarity between motifs. *Genome Biology*, 8(2):R24, 2007.

[42] J. Harrow, A. Frankish, J. M. Gonzalez, E. Tapanari, M. Diekhans, F. Kokocinski, B. L. Aken, D. Barrell, A. Zadissa, S. Searle, et al. Gencode: the reference human genome annotation for the encode project. *Genome research*, 22(9):1760–1774, 2012.

[43] J. R. Hesselberth, X. Chen, Z. Zhang, P. J. Sabo, R. Sandstrom, A. P. Reynolds, R. E. Thurman, S. Neph, M. S. Kuehn, W. S. Noble, S. Fields, and J. A. Stamatoyannopoulos. Global mapping of protein-dna interactions in vivo by digital genomic footprinting. *Nat Methods*, 6(4):283–9, Apr 2009.

[44] L. A. Hindorff, P. Sethupathy, H. A. Junkins, E. M. Ramos, J. P. Mehta, F. S. Collins, and T. A. Manolio. Potential etiologic and functional implications of genome-wide association loci for human diseases and traits. *Proc Natl Acad Sci U S A*, 106(23):9362–7, Jun 2009.

[45] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[46] M. M. Hoffman, O. J. Buske, J. Wang, Z. Weng, J. A. Bilmes, and W. S. Noble. Unsupervised pattern discovery in human chromatin structure through genomic segmentation. *Nature methods*, 9(5):473–476, 2012.

[47] S. John, P. J. Sabo, T. K. Canfield, K. Lee, S. Vong, M. Weaver, H. Wang, J. Vierstra, A. P. Reynolds, R. E. Thurman, et al. Genome-scale mapping of dnase i hypersensitivity. *Current protocols in molecular biology*, pages 21–27, 2013.

[48] D. S. Johnson, A. Mortazavi, R. M. Myers, and B. Wold. Genome-wide mapping of in vivo protein-dna interactions. *Science*, 316(5830):1497–502, Jun 2007.

[49] J. Keilwagen and J. Grau. Varying levels of complexity in transcription factor binding motifs. *Nucleic acids research*, page gkv577, 2015.

[50] D. R. Kelley, J. Snoek, and J. L. Rinn. Basset: learning the regulatory code of the accessible genome with deep convolutional neural networks. *Genome Res*, 26(7):990–9, Jul 2016.

[51] W. J. Kent, C. W. Sugnet, T. S. Furey, K. M. Roskin, T. H. Pringle, A. M. Zahler, and D. Haussler. The human genome browser at ucsc. *Genome research*, 12(6):996–1006, 2002.

[52] W. J. Kent, A. S. Zweig, G. Barber, A. S. Hinrichs, and D. Karolchik. Bigwig and bigbed: enabling browsing of large distributed datasets. *Bioinformatics*, 26(17):2204–2207, 2010.

[53] P. V. Kharchenko, M. Y. Tolstorukov, and P. J. Park. Design and analysis of chip-seq experiments for dna-binding proteins. *Nature biotechnology*, 26(12):1351–1359, 2008.

[54] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[55] M. Kircher, D. M. Witten, P. Jain, B. J. O'roak, G. M. Cooper, and J. Shendure. A general framework for estimating the relative pathogenicity of human genetic variants. *Nature genetics*, 46(3):310–315, 2014.

[56] D. Lee, D. U. Gorkin, M. Baker, B. J. Strober, A. L. Asoni, A. S. McCallion, and M. A. Beer. A method to predict the impact of regulatory variants from dna sequence. *Nature genetics*, 47(8):955–961, 2015.

[57] D. Lee, D. U. Gorkin, M. Baker, B. J. Strober, A. L. Asoni, A. S. McCallion, and M. A. Beer. A method to predict the impact of regulatory variants from dna sequence. *Nat Genet*, 47(8):955–61, Aug 2015.

[58] R. Leslie, C. J. O'Donnell, and A. D. Johnson. Grasp: analysis of genotype-phenotype results from 1390 genome-wide association studies and corresponding open access database. *Bioinformatics*, 30(12):i185–94, Jun 2014.

[59] M. K. K. Leung, H. Y. Xiong, L. J. Lee, and B. J. Frey. Deep learning of the tissue-regulated splicing code. *Bioinformatics*, 30(12):i121–9, Jun 2014.

[60] Y. Li and X. Xie. Deconvolving tumor purity and ploidy by integrating copy number alterations and loss of heterozygosity. *Bioinformatics*, 30(15):2121–9, Aug 2014.

[61] X. Liu, C. Wu, C. Li, and E. Boerwinkle. dbnsfp v3. 0: A one-stop database of functional predictions and annotations for human nonsynonymous and splice-site snvs. *Human mutation*, 37(3):235–241, 2016.

[62] J. Lonsdale, J. Thomas, M. Salvatore, R. Phillips, E. Lo, S. Shad, R. Hasz, G. Walters, F. Garcia, N. Young, et al. The genotype-tissue expression (gtex) project. *Nature genetics*, 45(6):580–585, 2013.

[63] P. Machanick and T. L. Bailey. MEME-ChIP: motif analysis of large DNA datasets. *Bioinformatics*, 27(12):1696–7, Jun 2011.

[64] A. Mathelier, O. Fornes, D. J. Arenillas, C.-y. Chen, G. Denay, J. Lee, W. Shi, C. Shyr, G. Tan, R. Worsley-Hunt, A. W. Zhang, F. Parcy, B. Lenhard, A. Sandelin, and W. W. Wasserman. Jaspar 2016: a major expansion and update of the open-access database of transcription factor binding profiles. *Nucleic Acids Research*, 44(D1):D110–D115, 2016.

[65] R. McLendon, A. Friedman, D. Bigner, E. G. Van Meir, D. J. Brat, G. M. Mastrogianakis, J. J. Olson, T. Mikkelsen, N. Lehman, K. Aldape, et al. Comprehensive genomic characterization defines human glioblastoma genes and core pathways. *Nature*, 455(7216):1061–1068, 2008.

[66] A. Medina-Rivera, M. Defrance, O. Sand, C. Herrmann, J. A. Castro-Mondragon, J. Delerce, S. Jaeger, C. Blanchet, P. Vincens, C. Caron, D. M. Staines, B. Contreras-Moreira, M. Artufel, L. Charbonnier-Khamvongsa, C. Hernandez, D. Thieffry, M. Thomas-Chollier, and J. van Helden. Rsat 2015: Regulatory sequence analysis tools. *Nucleic Acids Res*, 43(W1):W50–6, Jul 2015.

[67] A. Mortazavi, S. Pepke, C. Jansen, G. K. Marinov, J. Ernst, M. Kellis, R. C. Hardison, R. M. Myers, and B. J. Wold. Integrating and mining the chromatin landscape of cell-type specificity using self-organizing maps. *Genome research*, 23(12):2136–2148, 2013.

[68] A. Mortazavi, B. A. Williams, K. McCue, L. Schaeffer, and B. Wold. Mapping and quantifying mammalian transcriptomes by rna-seq. *Nature methods*, 5(7):621–628, 2008.

[69] J. Mueller and A. Thyagarajan. Siamese recurrent architectures for learning sentence similarity. In *AAAI*, pages 2786–2792, 2016.

[70] Neph, S. et al. An expansive human regulatory lexicon encoded in transcription factor footprints. *Nature*, 489(7414):83–90, Sep 2012.

[71] Y. Pan and S. Phan. Threshold for positional weight matrix. *Engineering Letters*, 16(4):498–504, 2009.

[72] F. Pedregosa et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[73] R. Pique-Regi, J. F. Degner, A. A. Pai, D. J. Gaffney, Y. Gilad, and J. K. Pritchard. Accurate inference of transcription factor binding from dna sequence and chromatin accessibility data. *Genome Res*, 21(3):447–55, Mar 2011.

[74] Q. Qin and J. Feng. Imputation for transcription factor binding predictions based on deep learning. *PLoS computational biology*, 13(2):e1005403, 2017.

[75] D. Quang, Y. Chen, and X. Xie. Dann: a deep learning approach for annotating the pathogenicity of genetic variants. *Bioinformatics*, 31(5):761–3, Mar 2015.

[76] D. Quang and X. Xie. Extreme: an online em algorithm for motif discovery. *Bioinformatics*, 30(12):1667–73, Jun 2014.

[77] D. Quang and X. Xie. Danq: a hybrid convolutional and recurrent deep neural network for quantifying the function of dna sequences. *Nucleic Acids Res*, 44(11):e107, Jun 2016.

[78] D. X. Quang, M. R. Erdos, S. C. J. Parker, and F. S. Collins. Motif signatures in stretch enhancers are enriched for disease-associated genetic variants. *Epigenetics Chromatin*, 8:23, 2015.

[79] A. Raj, H. Shim, Y. Gilad, J. K. Pritchard, and M. Stephens. mscentipede: Modeling heterogeneity across genomic sites and replicates improves accuracy in the inference of transcription factor binding. *PLoS One*, 10(9):e0138030, 2015.

[80] F. Ramírez, F. Dündar, S. Diehl, B. A. Grüning, and T. Manke. deeptools: a flexible platform for exploring deep-sequencing data. *Nucleic acids research*, 42(W1):W187–W191, 2014.

[81] J. E. Reid and L. Wernisch. STEME: efficient EM to find motifs in large data sets. *Nucleic Acids Res*, 39(18):e126, Oct 2011.

[82] H. S. Rhee and B. F. Pugh. Comprehensive genome-wide protein-dna interactions detected at single-nucleotide resolution. *Cell*, 147(6):1408–1419, 2011.

[83] G. R. Ritchie, I. Dunham, E. Zeggini, and P. Flicek. Functional annotation of non-coding sequence variants. *Nature methods*, 11(3):294–296, 2014.

[84] Roadmap Epigenomics Consortium et al. Integrative analysis of 111 reference human epigenomes. *Nature*, 518(7539):317–30, Feb 2015.

[85] A. Roberts and L. Pachter. Streaming fragment assignment for real-time analysis of sequencing experiments. *Nat Methods*, 10(1):71–3, Jan 2013.

[86] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[87] T. Saito and M. Rehmsmeier. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, 10(3):e0118432, 2015.

[88] A. Sandelin, W. Alkema, P. Engström, W. W. Wasserman, and B. Lenhard. JASPAR: an open-access database for eukaryotic transcription factor binding profiles. *Nucleic Acids Res*, 32(Database issue):D91–4, Jan 2004.

[89] G. K. Sandve, M. Nedland, Ø. B. Syrstad, L. A. Eidsheim, O. Abul, and F. Drabløs. Accelerating motif discovery: Motif matching on parallel hardware. In *Algorithms in Bioinformatics*, pages 197–206. Springer, 2006.

[90] R. I. Sherwood, T. Hashimoto, C. W. O'Donnell, S. Lewis, A. A. Barkal, J. P. van Hoff, V. Karun, T. Jaakkola, and D. K. Gifford. Discovery of directional and nondirectional pioneer transcription factors by modeling dnase profile magnitude and shape. *Nat Biotechnol*, 32(2):171–8, Feb 2014.

[91] A. Shrikumar, P. Greenside, and A. Kundaje. Learning important features through propagating activation differences. *arXiv preprint arXiv:1704.02685*, 2017.

[92] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[93] A. Smit, R. Hubley, and P. Green. RepeatMasker Open-3.0, 1996-2010.

[94] N. Srivastava. *Improving neural networks with dropout*. PhD thesis, University of Toronto, 2013.

[95] G. D. Stormo, T. D. Schneider, L. Gold, and A. Ehrenfeucht. Use of the perceptronalgorithm to distinguish translational initiation sites in e. coli. *Nucleic acids research*, 10(9):2997–3011, 1982.

[96] M. Sundermeyer, T. Alkhouli, J. Wuebker, and H. Ney. Translation modeling with bidirectional recurrent neural networks. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing, October*, 2014.

[97] I. Sutskever et al. On the importance of initialization and momentum in deep learning. In *ICML-13*, pages 1139–1147, 2013.

[98] M. Thomas-Chollier, M. Defrance, A. Medina-Rivera, O. Sand, C. Herrmann, D. Thieffry, and J. van Helden. RSAT 2011: regulatory sequence analysis tools. *Nucleic Acids Res*, 39(Web Server issue):W86–91, Jul 2011.

[99] H. Thorvaldsdóttir, J. T. Robinson, and J. P. Mesirov. Integrative genomics viewer (igv): high-performance genomics data visualization and exploration. *Briefings in bioinformatics*, 14(2):178–192, 2013.

[100] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4, 2012.

[101] K. Wang, M. Li, and H. Hakonarson. Annovar: functional annotation of genetic variants from high-throughput sequencing data. *Nucleic acids research*, 38(16):e164–e164, 2010.

[102] D. Welter, J. MacArthur, J. Morales, T. Burdett, P. Hall, H. Junkins, A. Klemm, P. Flicek, T. Manolio, L. Hindorff, and H. Parkinson. The nhgri gwas catalog, a curated resource of snp-trait associations. *Nucleic Acids Res*, 42(Database issue):D1001–6, Jan 2014.

[103] J. W. Whitaker, Z. Chen, and W. Wang. Predicting the human epigenome from dna motifs. *Nature methods*, 12(3):265–272, 2015.

[104] X. Xie, J. Lu, E. J. Kulbokas, T. R. Golub, V. Mootha, K. Lindblad-Toh, E. S. Lander, and M. Kellis. Systematic discovery of regulatory motifs in human promoters and 3' UTRs by comparison of several mammals. *Nature*, 434(7031):338–45, Mar 2005.

[105] J. Zhou and O. G. Troyanskaya. Predicting effects of noncoding variants with deep learning-based sequence model. *Nat Methods*, 12(10):931–4, Oct 2015.

[106] W. Zhu, C. Lan, J. Xing, Y. Li, L. Shen, W. Zeng, and X. Xie. Co-occurrence feature learning for skeleton based action recognition using regularized deep lstm networks. *The 30th AAAI Conference on Artificial Intelligence (AAAI-16)*, 2016.