A Distributed Semi-Supervised Platform for DNase-Seq Data Analytics using Deep Generative Convolutional Networks

Shayan Shams*
Louisiana State University
Baton Rouge, Louisiana
sshams2@cct.lsu.edu

Richard Platania*
Louisiana State University
Baton Rouge, Louisiana
rplata1@lsu.edu

Joohyun Kim
Louisiana State University
Baton Rouge, Louisiana
jhkim@cct.lsu.edu

Jian Zhang
Louisiana State University
Baton Rouge, Louisiana
zhang@csc.lsu.edu

Kisung Lee
Louisiana State University
Baton Rouge, Louisiana
lee@csc.lsu.edu

Seungwon Yang
Louisiana State University
Baton Rouge, Louisiana
seungwonyang@lsu.edu

Seung-Jong Park
Louisiana State University
Baton Rouge, Louisiana
sjpark@cct.lsu.edu

ABSTRACT

A deep learning approach for analyzing DNase-seq datasets is presented, which has promising potentials for unraveling biological underpinnings on transcription regulation mechanisms. Further understanding of these mechanisms can lead to important advances in life sciences in general and drug, biomarker discovery, and cancer research in particular. Motivated by recent remarkable advances in the field of deep learning, we developed a platform, Deep Semi-Supervised DNase-seq Analytics (DSSDA). Primarily empowered by deep generative Convolutional Networks (ConvNets), the most notable aspect is the capability of semi-supervised learning, which is highly beneficial for common biological settings often plagued with a less sufficient number of labeled data. In addition, we investigated a k-mer based continuous vector space representation, attempting further improvement on learning power with the consideration of the nature of biological sequences for features associated with locality-based relationships between neighboring nucleotides. DSSDA employs a modified Ladder Network for underlying generative model architecture, and its performance is demonstrated on the cell type classification task using sequences from large-scale DNase-seq experiments. We report the performance of DSSDA in both fully-supervised setting, in which DSSDA outperforms widely-known ConvNet models (94.6% classification accuracy), and semi-supervised setting for which, even with less than 10% of labeled data, DSSDA performs relatively comparable to other ConvNets using the full data set. Our results underscore, in order to deal with challenging genomic sequence datasets, the need of a better deep learning method to learn latent features and representation.

*This author contributed equally to this work.

KEYWORDS

DNase-Seq; Convolutional Networks; Semi-Supervised learning; Generative models; Continuous vector representation; Deep Learning

ACM Reference Format:
Shayan Shams, Richard Platania[1], Joohyun Kim, Jian Zhang, Kisung Lee, Seungwon Yang, and Seung-Jong Park. 2018. A Distributed Semi-Supervised Platform for DNase-Seq Data Analytics using Deep Generative Convolutional Networks. In *Proceedings of 9th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics (ACM-BCB'18)*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3233547.3233601

1 INTRODUCTION

In the past decade, high-throughput sequencing technologies, such as Next-Generation Sequencing (NGS), have significantly advanced life sciences, particularly with their genome-wide profiling capacities for genomic, transcriptomic, and even epigenomic signatures. Despite enormous information contents, the large amount of raw sequencing data sets produced, as well as inherently complex biological implications, pose considerable analytical challenges[7].

To tackle such challenges, life science domains have begun to embrace deep learning methodologies. This is encouraged by impressive success in a variety of fields, exhibiting particularly remarkable accuracy for image, textual, and speech datasets[18, 20, 32]. Already, some recent works have explored the potentials of deep learning to improve upon methods for biological sequences[2, 15, 41]. Notably, the most attractive aspect of deep learning appears to be its intrinsic capability for end-to-end solutions (e.g., allowing the direct use of biological data, such as nucleotide sequences, as input). This is in contrast to conventional machine learning methods relying upon hand-crafted feature sets, requiring a significant amount of domain knowledge. This capability is in fact closely related to powerful representation learning, leading to discovery of latent features of a given data set[5]. In addition, a convenient extension to incorporate multi-task learning underscores its great potential for combining heterogeneous information from multi-omics data sets[41].

Regarding life science applications, one of the major obstacles is the difficulty of producing labeled data. Experimentally labeling data is both costly and, in some cases, not feasible. This deficiency of labeled data causes particular difficulty with respect to traditional deep learning, which is typically performed in a supervised manner. To overcome this challenge, semi-supervised methods are highly desirable[38]. In fact, an impressive increase in throughput in sequencing technologies ironically hampers a step for labeling sequences, implying an urgent demand of methods that mitigate the problem.

In this work, we introduce DSSDA, a distributed semi-supervised deep learning method, and demonstrate its performance for cell-type classification task using sequences, identified as DNase I hypersensitive Sites (DHSs), obtained from DNase-Seq experiments. We design DSSDA based on the Ladder Network[29, 30, 37] architecture considering its core design strategy to carry out semi-supervised learning. Our platform is built based upon the Ladder Network by adding the ConvNet architecture, rather than traditional Multi-Layer Perceptron (MLP), resulting in the combination of the two effective methodologies in the field of deep learning. We also investigate two different input representations: one-hot vector and continuous vector space models[27, 28]. The latter is particularly considered as an attempt to examine the unknown nature of biological sequences. It hypothesizes that the local correlation over neighboring nucleotides is important for additional improvement of prediction.

Our main contributions can be divided into three parts: (1) We provide DSSDA, a distributed and semi-supervised method for biological sequence classification based on deep generative ConvNets. In particular, it uses a modified version of the ladder architecture that supports ConvNets. (2) DSSDA improves upon previous accuracy results in fully-supervised mode and achieves comparable accuracy by using only 10% of labeled data in semi-supervised mode. (3) We compare two input representations for sequence data, one-hot vector and probabilistic continuous vector-space representations, and present useful findings with our experiments for the latter.

The remainder of the paper is organized as follows. We first give insight towards the background of deep learning and our biological analysis task. Following this, we summarize related works. Then, our methods are detailed, primarily relating to the sequence classification problem, dataset, input representations, deep learning models, and DSSDA. After that, results are presented followed by discussions focusing on implications of the resulting strengths of DSSDA. Finally, we conclude the paper and state our plans for future works.

2 BACKGROUND

Our background is divided into two parts. First we discuss DNase-Seq data and analytics. The second part focuses on the background of supervised and semi-supervised deep learning with ConvNets.

2.1 DNase-Seq and Data Analytics

An understanding of gene expression mechanisms in a living cell is one of the holy grails in biology and has immense implications for life sciences, in general, and diseases such as cancer, in particular. While directly measuring gene expression levels with RNA-seq is still a common approach, unraveling epigenomic components is becoming increasingly recognized as a crucial task since deeper insights into deciphering the regulation of transcription can be obtained [7, 19]. Among various methods for probing epigenomic nature, DNase-Seq is a NGS platform for detecting directly open chromatin regions implicated as DNase I Hypersensitive Sites (DHSs) [11, 13, 36]. Experimentally identified DHSs from various cell types and conditions provide important clues towards how eukaryote genomes are conditionally organized via chromatin packaging with other molecules, such as histone proteins and various markers. This consequently informs regions only to be dynamically exposed for specific binding of regulatory molecules such as transcription factors[11, 13, 33].

The two major consortia, the Encyclopedia of DNA Element (ENCODE) [7] and Roadmap Epigenomics [19], have carried out large-scale DNase-Seq experiments against human samples of various cell and tissue types. Analyzing such large scale data sets in order to decipher underlying unknown implications remains challenging [41].

2.2 Supervised and Semi-Supervised Deep Learning with ConvNets

Supervised learning is the most common and frequently used method in many machine learning tasks and deep learning problems. On the other hand, semi-supervised learning is gaining popularity within domains plagued by large amounts of unlabeled data. Semi-supervised learning techniques enable training a model with data having limited labels[38]. While there is a typical sacrifice in accuracy when using semi-supervised learning, advanced techniques have begun to close the gap between fully-supervised and semi-supervised deep learning performance.

Among many deep learning architectures, ConvNet models have been remarkably successful for image classification and other problems such as speech recognition and natural language processing[18, 20]. Their success is rooted in their capacity with multiple stacked layers and efficient feature extraction with convolutional layers, often explained as a powerful representation learning method. In the last few years, the advent of ConvNets has been sensational, with new additions on top of the classical form, enabling deconvolutional architecture, object detection and image segmentation, generative models, and semi-supervised learning[10, 16, 17, 39]. One challenging problem with the ConvNet models is the difficulty of finding an optimal architecture and hyper-parameters for the best outcome[40].

In addition to the use of ConvNets, our work specifically focuses on the use of generative models for more powerful representation learning and its extension for semi-supervised learning. Generative deep learning models are one of great advances in the modern deep learning due to enabling unsupervised learning[14].

Various algorithms for generative models have been proposed recently, such as Variational Autoencoders (VAE)[16, 31] and Generative Adversarial Networks (GAN)[10], and thus can be used for our purpose. We found that the Ladder Network architecture was attractive with its own unique properties[30, 37]. For example, the architecture can be utilized with different feed-forward networks, and the lateral connections provide an interesting addition to the underlying denoising autoencoder-based generative model scheme[29].

3 RELATED WORKS

Machine learning methods have been actively developed for biological sequence problems and dominated by methods such as Support Vector Machine[9]. Several recent works have employed deep learning and demonstrated advantages over conventional methods in terms of scalability, support of multi-tasking, and other benefits [2, 12, 41].

Kelley et al. previously applied ConvNets to DNase-seq data [15]. One of their focuses, like our main task, was cell-type classification. Furthermore, they found that the features generated by early convolution layers are representative of sequence motifs of known transcription factor binding sites. Unlike their work, which is only based on a simple ConvNet model (LeNet[21]), our work, DSSDA, can employ any deep ConvNet models and incorporate them in a semi-supervised-capable architecture[29, 30, 37].

As for our methodological sides, the Ladder Network is the central idea and well explained in recent papers[29, 30, 37]. This network model is designed to implement the generative capabilities by following the methodology of the denoising autoencoder-based approach[6]. Semi-supervised learning gets less attention outside of computer sciences and statistical learning fields but has still been one of the active research areas[16, 24, 30, 38].

Input representation has been extensively studied for language modeling in Natural Language Processing. We employed an approach, *word2vec*, introduced by Mikolov et al.[27], which can be implemented in either of the two different schemes: the Continuous Bag-of-Words *(CBOW)* model and the Skip-Gram model.

4 METHODS

In this section, we will describe the sequence classification problem, the dataset used for this work, the two input representation techniques (namely, One-hot vector and continuous vector-space model), the deep learning models constituting DSSDA employed for supervised and semi-supervised learning modes, and our distributed training implementation.

4.1 Sequence Classification Problem

A sequence may be associated with multiple cell/tissue types. Hence it may have multiple labels. The classification problem is to determine, for a given sequence, all the labels that apply to the sequence. In other words, the classification is correct if and only if all possible labels for that sequence are correctly predicted. There are a total of 164 cell and tissue types from datasets. 125 cell types come from ENCODE[36] and 39 from the Roadmap Epigenomics Consortium [26]. These two datasets are merged to form our entire dataset. The cell or tissue types originating each sequence is its label for the main classification task.

For pre-processing the raw data, we followed the same process of Kelley et al. [15]. We downloaded BED files and merged them into one BED and activity table. Sequences with more than 200 *bp* overlap were merged and extended to 600 *bp*, using the hg19 reference genome.

4.2 Input Representation

4.2.1 One-hot vector representation. The default representation for our input is the one-hot vector format, and it is preprocessed as follows. Using the script provided by Kelley et al.[1], for a 600 *bp* sequence input, in the one-hot vector representation, each nucleotide position has a four-element vector representing A, T, C, G, with one nucleotideâĂŹs bit set to one, resulting in a vector of size 2,400 = $(1 \times 600 \times 4)$. Since ConvNets are mostly used for images with an input typically described as (width, height, channel), each sequence can be seen as an image with a width of 600, height of 1, and with 4 channels in comparison with RGB pictures which have 3 channels.

4.2.2 Probabilistic continuous vector-space model. A *k*-mer based approach is popular to deal with genomic data[9], facilitating to preserve the locality relationship between adjacent *k*-mers in the sequence. However, using a *k*-mer based approach that encodes each *k*-mer as a discrete arbitrary ID will produce limited information regarding the relationship between *k*-mers, structure, and semantics of sequence. As a result, the model is able to leverage very little of what it has learned about one *k*-mer when considering other *k*-mers. Furthermore, representing *k*-mers as random IDs causes data sparsity, which means that more data is required for successful training. Probabilistic embedding models represent each word in a continuous vector space, where semantically similar words are mapped to nearby points[4]. We believe this makes sense for a *k*-mer based approach and, more importantly, further increases the representation learning performance provided by ConvNet-based mapping between inputs and labels. We use *CBOW* to train the model with the objective of discriminating a target *k*-mer from a noise *k*-mer.

We consider each 3-mer as one word. Note that the size of the *k*-mer is in fact arbitrary and remains to be logically determined if possible. As a result, we have 598 3-mers for each sequence and a vocabulary length of $4 \times 4 \times 4$, since each position can have four characters. We consider a window size of one, which means the model should predict the one adjacent *k*-mer, and the context length considered is nine *k*-mers. We used stochastic gradient descent (SGD) using one *k*-mer at a time and a mini-batch size of 16 for optimization to produce embedding representation.

4.3 Deep ConvNets for Supervised learning

Since the overall architectural design of DSSDA supports any ConvNet model, we first tested well-known ConvNet models as standalone before being embedded in DSSDA, which are the winners of the ImageNet Large Scale Visual Recognition Competition (ILSVRC) [8]. The models include AlexNet and Inception*V1* [18, 34, 35], along with the historically well-known LeNet[22] model. Note that the

[1] https://github.com/davek44/Basset

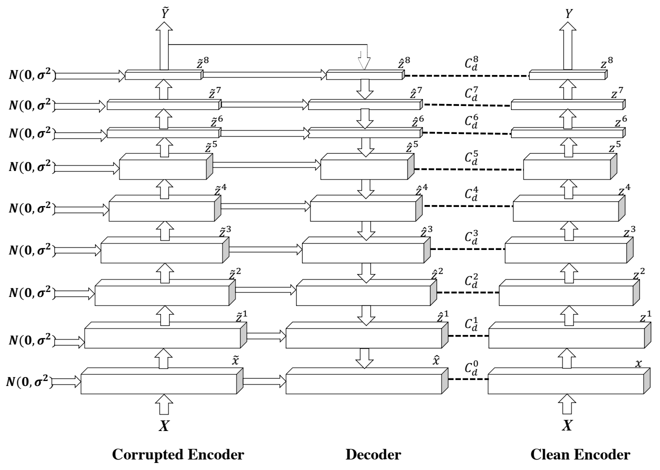


Figure 1: Schematic illustration of DSSDA architecture. Here, as an example, the employed ConvNet architecture is AlexNet.

LeNet architecture was used by Kelley et al. [15], and we refer to this model as the baseline. For the remaining models, our implementations followed the main architecture of the original models, such as the number of convolutional and pooling layers. However, hyper-parameters, such as filter size, stride, initialization method, batch normalization, number of neurons in fully connected layer, and batch size, were tuned to ensure the best performance of each model specifically for the problem at hand. All ConvNet models were designed and implemented in TensorFlow version 1.2[1].

The performance of each model is measured with True Positive Rate and False Positive Rate, and thus presented with the Area Under Receiver Operating Characteristic Curve (in short, AUC).

4.4 Deep Generative ConvNets for Semi-supervised Learning

DSSDA is able to run a supervised task but more importantly enables a high-performing semi-supervised learning task. As it is shown in Figure 1, DSSDA comprises two encoder paths and one decoder path (middle). The only difference between clean and corrupted encoder is that the corrupted encoder (left) adds Gaussian noise $N(0, \sigma^2)$ to all layers. Lateral connections for each layer of the decoder from the corresponding layer of the encoder have a potential role for more powerful representation learning by influencing the encoder function to the decoder function. Note that without such a mechanism, unsupervised learning is highly likely to be non-specific, as most approaches suffer. In brief, the central rational behind this architecture is its capacity such that (1) it can act as a generative model with a denoising autoencoder embedded in the model, (2) it provides simultaneous learning of discriminative (supervised) and generative model (unsupervised), and (3) it is a hierarchical latent model with skip connections. The decoder inverts the mappings on each layer of the corrupted encoder and supports unsupervised learning. It further uses a denoising function to reconstruct the activations of each layer given the corrupted version. The corrupted encoder function is defined in Eq.1, in which W^l is the weights of the convolution filters at layer l. $\tilde{h}^{l-1}$ is the output of the previous layer, and $Conv$ is the Convolution operation. $N(0, \sigma^2)$ is Gaussian noise, $\tilde{h}^l$ is the output of layer l, and $\tilde{h}^0 = x +$ noise is the original input x with noise added. (See Figure 1.)

$$\begin{aligned} \tilde{z}^l &= batchnorm(Conv(\tilde{h}^{l-1}, W^l) + Bias) + N(0, \sigma^2) \\ \tilde{h}^l &= RELU(\tilde{z}^l) \end{aligned} \tag{1}$$

The clean encoder follows the same weights and hyper-parameters, for both the convolutional and the fully connected layers, as the corrupted encoder except that no noise is added at each layer. We denote the output of the clean encoder at layer l as z^l. The final classification result is the output of the clean encoder. In the decoder path, we use deconvolution layers and a denoising function to reconstruct the output of the corrupted encoder for each layer. Algorithm 1 shows how the decoder works. In the Algorithm, V^l is the weights of the deconvolution filters. $\hat{z}^l$ is the output of the denoise function for layer l. We used the same denoise function as the one in Rasmus et al [30]. Furthermore, $\hat{z}^l_{BN}$ is the normalized $\hat{z}^l$ for each batch of data with μ^l and σ^l being the mean and the standard deviation for layer l, respectively.

Algorithm 1 Decoder path

```
for $l = L$ to 0 do
    if l=L then
        $u^l = batchnorm(\tilde{h}^l)$
    else
        $u^l = batchnorm(Deconv(\hat{z}^{l+1}, V^{l+1}) + Bias)$
    end if
    $\hat{z}^l = denoise(\tilde{z}^l, u^l)$
    $\hat{z}^l_{BN} = \frac{\hat{z}^l - \mu^l}{\sigma^l}$
end for
```

The reconstruction target for each layer of the decoder is the clean version of the activation provided by each layer of the clean encoder following Eq.1 but without $N(0, \sigma^2)$. The squared difference between the reconstruction and the clean version serves as the denoising cost of that layer, represented as C_d^l in Figure 1.

The supervised cost is calculated from the output of the corrupted encoder and the sequence label(s). Let N be the number of samples, S the set of classes ($|S|$ the number of classes), and $Y^c_{target}(n) \in \{0, 1\}$ be the indicator of whether the n-th sequence belongs to class c. The loss function used for calculating the supervised cost is defined in Eq. 2.

$$\begin{aligned} Y_{predict} &= sigmoid(\tilde{Y}) \\ Loss_{Supervised} &= -\frac{1}{N \cdot |S|} \sum_{n=1}^{N} \sum_{c \in S} \Big(Y^c_{target}(n) \cdot \log Y^c_{predict}(n) + \\ &\quad (1 - Y^c_{target}(n)) \cdot \log(1 - Y^c_{predict}(n)) \Big) \end{aligned} \tag{2}$$

The unsupervised cost is the sum of the denoising cost C_d^l across all layers l, scaled by a hyper-parameter that denotes the significance of each layer. Eq.3 shows how the unsupervised cost is calculated, where λ_l is the hyper-parameter determining denoising cost.

$$Cost_{unsupervised} = \sum_{l=0}^{L} \lambda_l C_d^l = \sum_{l=0}^{L} \frac{\lambda_l}{N} \sum_{l=1}^{N} (z^l(n) - \hat{z}_{BN}^l(n))^2 \tag{3}$$

The final cost, shown in Eq.4, is the sum of supervised and unsupervised cost. The whole network can be trained in fully-labeled or semi-supervised setting using stochastic gradient descent (SGD) to minimize the overall cost.

$$Cost_{overall} = Cost_{unsupervised} + Loss_{supervised} \tag{4}$$

Since our experiments with different supervised models, mentioned in Section 5.2, show that AlexNet has the best performance among others for this problem, we chose the AlexNet architecture as base for our semi-supervised implementation. As it is shown in Fig. 1, both encoders are using convolutional layers, and the decoder is using deconvolutional layers for decoding. However, there are some exceptions. In the encoding path, the first three convolutional layers are followed by 2×1 pooling layers, and in the decoder path, convolutional layers are replaced with deconvolutional layers (and pooling layers are replaced with unpooling layers). Also, we implement fully connected layers with convolutional layers for encoders and decovolutional layers for decoder to get a better performance and easier implementation.

Note that the first layer shown in Figure 1 is in fact the input and the output for the generative pathway, emphasizing the adding of noise and denoising in both the encoder and decoder paths. Consequently, with the AlexNet architecture, the decoder layer has five deconvolutional layers and three fully connected layers. We chose the noise hyper-parameter λ_l for all the layers to be 0.2 to make sure all the layers have the same significance when calculating loss. We trained DSSDA in fully-supervised mode using 100% labeled training set in Table 1. For the semi-supervised mode, 10% (25%) of labeled training set are used, and the remaining 90% (75%) are given as input for unsupervised learning without any labels.

4.5 Distributed Training

Training with large-scale data sets is challenging because the training can often take days or even weeks. As we will see in the results section, training DSSDA to convergence with even a single state-of-the-art Nvidia Tesla-P100 GPU takes nearly five days to complete. Because of this, it is important to consider parallel and distributed training techniques that improve upon the execution time. In this section, we describe a few of these techniques that are employed by DSSDA to make the training time more feasible.

For training DSSDA, we consider data parallelism during the training process. Data parallelism is fairly straightforward, consisting of duplicating the DSSDA model in each worker and dividing the input into mini-batches and distributing them among the different GPU workers. Each worker will receive a different mini-batch. The strategy for data parallelism is depicted in Figure 2.

The second technique we will consider is asynchronous training. When training a model with multiple workers, the gradients

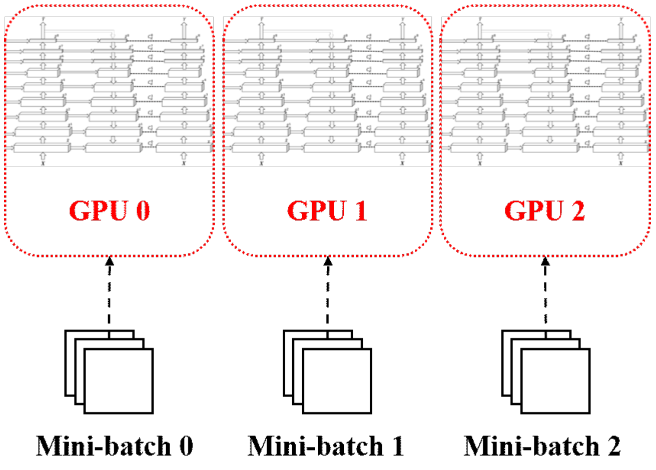


Figure 2: Data parallelism in DSSDA.

computed by each need to be sent to a parameter server. The server then updates the parameters and redistributes them to the workers. In synchronous training, this process is performed after training a single input batch. Asynchronous training can be introduced in order to limit this parameter communication and synchronization overhead. For DSSDA, we can specify how often the synchronization occurs. There are some trade offs that come with increasing the asynchronous nature of training. The longer the interval of time between synchronization, the shorter the average training time per batch will be. However, this also means that the training becomes more unstable and does not converge as quickly or easily as fully synchronous training. DSSDA is able to take advantage of both synchronous and asynchronous training.

5 RESULTS

In Section 5.1, we explain how the dataset was divided for training, test, and validation sets. In Section 5.2, we present the comparison result of our proposed model, DSSDA, with other ConvNets in the supervised mode. Then, in Section 5.3, we present the results with DSSDA in the semi-supervised mode. Following this, we give some results regarding our distributed training in Section 5.4. Our results for comparing the two input representations and visualization of regulatory motifs learned with filters in DSSDAare presented in Sections 5.5 and 5.6, respectively.

5.1 Dataset

In Table 1, the summary of our dataset used for this project is presented. It is important to note that many sequences originate from multiple classes (i.e., cell types), requiring multi-label classification, instead of commonly applied single-label classification. In the supervised setting, the whole training set is used along with their labels. For the semi-supervised setting, we divide the training set into labeled and unlabeled inputs, and the latter is used without labels for unsupervised learning. For the validation and testing, 70,000 and 71,886 are randomly selected, respectively.

Table 1: Summary of the DNase-Seq data set

Number of the total DHSs	2,071,886
Training Data	1,930,000
Validation Data	70,000
Testing Data	71,886
Number of nucleotides in each DHS	600 *bp*
Number of cell and tissue types	164

Table 2: Comparison of classification performance between DSSDA, other ConvNet models, and previously used SVM.

Network	Accuracy
gkm-SVM[23]	0.780
LeNet	0.895
Inception	0.911
AlexNet	0.929
DSSDA (all labels)	0.946
DSSDA (10% labeled)	0.822
DSSDA (25% labeled)	0.853

5.2 Supervised Learning

In Table 2, we give the comparison of the mean AUC of 164 classes between the ConvNet models we tested and DSSDA in the supervised mode. Individual AUC for each class is obtained using the validation dataset comprising of 70,000 sequences. Then, we calculate the mean AUC for all 164 classes. Note that the number of sequences for each class is not uniform, depending upon sizes of original experimental data and random selection for validation. As Table 2 clearly indicates, AlexNet showed the best performance among the ConvNet models, leading to our decision to choose this as a model for DSSDA . Importantly, DSSDA containing AlexNet as an internal ConvNet model outperforms all other models.

To understand how DSSDA outperforms conventional ConvNet models, we further compare the ROC curves for 10 random classes between AlexNet and DSSDA. As shown in Figure 3, this comparison reveals not only the fact that DSSDA outperforms AlexNet, but that its classification performance for each class is uniform. This aspect can be more clearly visualized with the results in Figure 4.

Here, we compare the two models again with respect to the calculated AUC for each class. While the results with the ConvNet model with AlexNet shows relatively distributed AUCs and occasionally high variation in some classes, DSSDA produces uniform AUCs. Calculated standard deviation of AUCs for all 164 classes is 1.02 for DSSDA and 1.85 for AlexNet. We will discuss this more later to argue the intrinsic advantages of DSSDA with these findings.

5.3 Semi-supervised Learning

Obtaining enough labeled data is one of the huge challenges when it comes to training a deep learning model. Hence, we are motivated to decrease the amount of labeled data in our training set with the goal of maintaining comparable AUC to results from using the whole labeled data set in the fully-supervised mode. Table 2 shows our main results for semi-supervised learning. In the case that only 10% of sequences in the training dataset with their labels are used, along with the remaining 90% of sequences are provided without labels, DSSDA achieves 0.822 as mean AUC, which is comparable to results with other ConvNet models. When we use the same model of Kelley et al.[15] (i.e., LeNet) with 100% of labels, the mean AUC is 0.895. Note that conventional ConvNets are unable to handle a data set with only such a small amount of labeled sequences.

5.4 Distributed Training Analysis

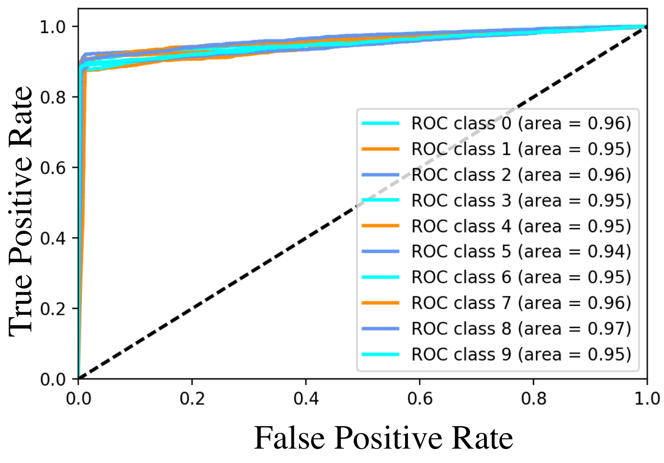
In this subsection, we give insight towards our distributed training analysis. To provide some form of a baseline, the time to train a batch of size 512 on one P100 GPU is 1.2 seconds for DSSDA. We discuss the effects of data parallelism and asynchronous training on performance. For these experiments, the time for one training iteration was calculated by taking the average of 100 iterations. Given this 1.2 seconds per training iteration with a single P100 GPU, 90 epochs to convergence, and 1,930,000 sequences, the total training time took approximately 4.7 days (113 hours).

The data parallelism employed by DSSDA is depicted in Figure 2. In this example, the model is replicated across three P100 GPUs, each receiving a mini-batch of size 170 (approximately 512/3). Note that this strategy can be applied to any number of two or more GPUs. With this strategy, the average time per iteration is reduced to approximately 0.55 seconds. This is a speedup of approximately 2.18x over the single GPU speed of 1.2 seconds. In an ideal situation, the speedup would be 3x since we are dividing the batch into three parts for each GPU. However, the extra communication overhead in parameter synchronization leads to reduced performance. Even with this extra communication overhead, the speedup with three GPUs results in a training time of approximately 2.18 days compared to 4.7 days with one GPU. Applying data parallelism to DSSDA results in multiple days saved in the overall training time.

In an attempt to reduce the communication overhead involved with parameter-server synchronization, we applied asynchronous training in DSSDA. To begin, we used the same setup as we had in the data parallelism experiment (3 GPUs with mini-batches of 170) and restricted the synchronization to only occur every 20 steps. While this does reduce the overall average time per training iteration by 2.3x (0.55 to 0.24), it introduces an instability in the model. We found that, with a more complex model like DSSDA, asynchronous training is not as feasible. After 90 epochs, the number of epochs at which a single GPU converged, the model was still unstable and not converged. This lead to a reduced testing accuracy of approximately 82%, which is significantly lower than DSSDA's best accuracy of 94.6%. We found that this trade off in reduced accuracy for reduced time was not worth the effort in the case of our model and application.

5.5 Input Representation

As explained in 4.2, the two input representations used are the one-hot vector format and the embedding vector format with 3-mer and neural probabilistic continuous vector space model with *word2vec*. Since continuous vector space model representation adds extra computation cost, discussed in detail later, we tested both input representations for the simple LeNet network architecture. The other architectures presented a challenge in training time with



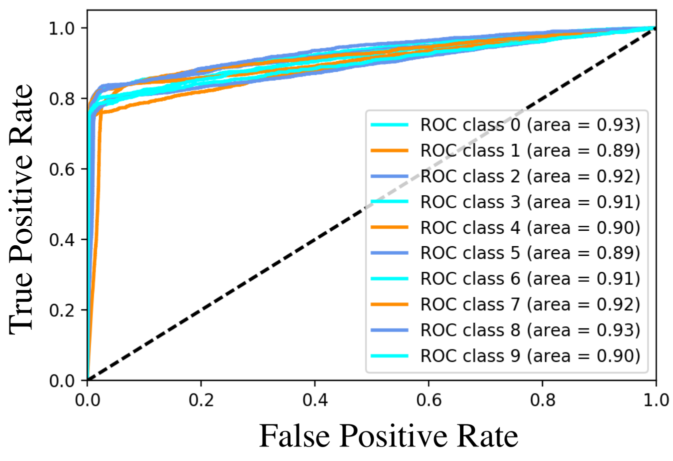
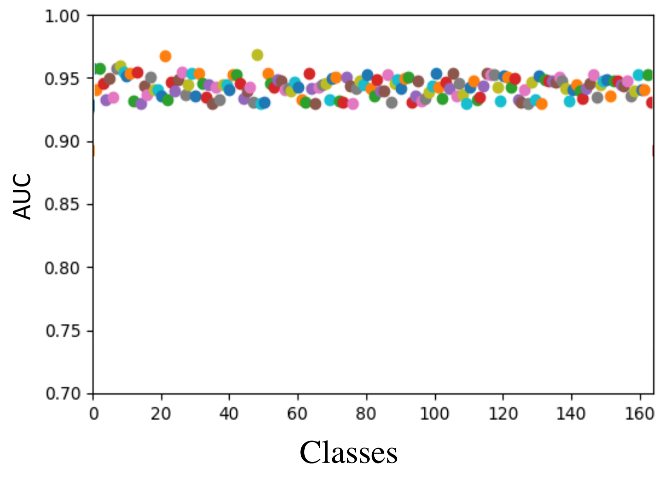


Figure 3: Comparison of classification AUC between DSSDA using the fully-supervised learning mode and a best performing ConvNet architecture, AlexNet (right)



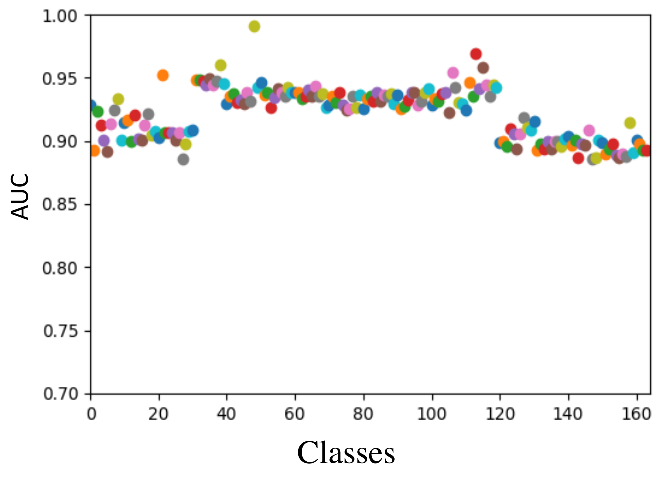


Figure 4: Comparison of classification performance between DSSDA (left) in supervised mode and AlexNet (right) with respect to AUC values for each cell type.

the different input representations since continuous vector space model representation adds extra computation cost to training. Although LeNet is a simple network, it can still give proper insight towards the quality of different input representations. Probabilistic continuous vector input representation achieved a final higher AUC of 0.904 in comparison to 0.89 obtained by one-hot vector. Another observation is that the embedding vector representation is found to converge faster. For example, it converged in 42 epochs, in comparison to one-hot vector representation that converged after 73 epochs. However, time for one training iteration for one-hot vector encoding is 0.31 second while it is 3.25 seconds for *word2vec* for LeNet model and the same batch size. Despite slightly better accuracy and potential advantages, the word embedding method requires additional computing cost, compared to one-hot vector input, which in turn limits the usage of word embedding.

5.6 Motif Vizualization

As shown by Kelley et al. [15], another interesting outcome with ConvNets is that learned features in filters of convolution layers could contain the information about real regulatory motifs, such as transcription factor binding sites. For example, they showed that the first convolution layer of their model recovers an extensive repertoire of known DNA binding motifs. Following the same process and using the scripts that they provided[2], we present the comparison of the recognized motifs from Tomtom's *Homo sapiens* motif database [3] and learned motifs by DSSDA for each input

[2]https://github.com/davek44/Basset

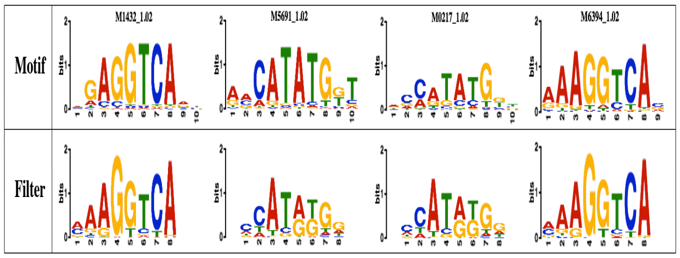


Figure 5: Visualization of learned features from our model. Known motifs from transcription factor binding site database (top) are found to be picked up by first layer filters (bottom).

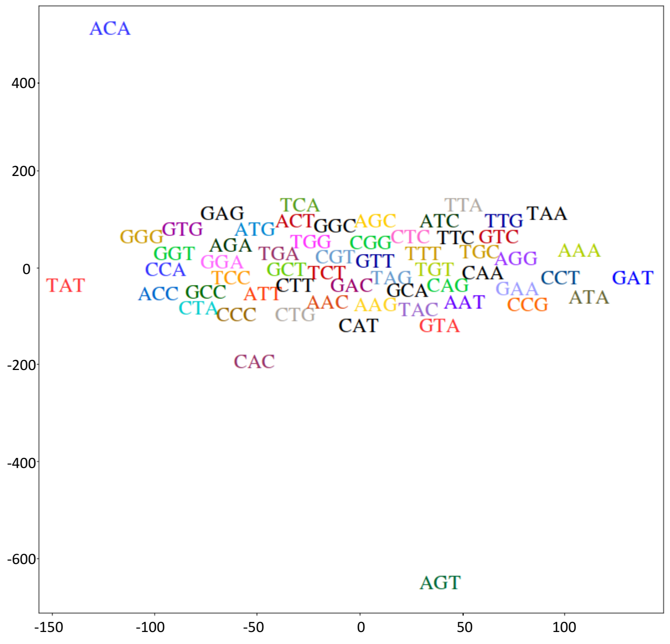


Figure 6: Visualization of learned embedding of different 3-mers via *t-SNE*.

sequence in Figure 5. The visualized patterns are for the first Convolution layer filters in the clean encoder for specific sequences. It is important to note that many of filters information cannot be interpreted to DNA binding motifs. Thus, we only visualized those which can be annotated by the software.

6 DISCUSSION

Here, we will discuss resulting implications from the strengths of DSSDA. These discussions are divided into two sections: DSSDA for biological sequence analysis and input representation with probabilistic continuous vector-space model.

6.1 DSSDA for Biological Sequence Analysis

In our comparison among the well-known architectures, AlexNet showed the best performance and was selected for DSSDA. DSSDA outperforms the model with the same deep ConvNet architecture (i.e., AlexNet). This implies further performance gains due to the learning mechanism with the generative model. Indeed, simultaneous training of the discriminative and generative models in encoder and decoder paths (unsupervised learning mechanism) facilitates the supervised learning task. In other words, DSSDA can leverage the features learned from sequences of other classes to learn hidden features related to different classes. We argue that the results shown in Figure 4 could support such a claim. The uniform AUC are likely to be associated with improved generalization from the contributions of other class sequences. Classification tasks of classes with an insufficient number of sequences could be improved, comparable to other classes with more samples in the data set. Another reason for the better accuracy with the Ladder architecture is, as argued by Valpola[37] and investigated by others[29], the unique lateral connection would be an additional mechanism to focus on features selectively learned by the supervised learning during the unsupervised learning pathway that generally suffers from a large representation space to be explored.

More importantly, the same argument is applied for its capability for semi-supervised learning. Indeed, despite a smaller number of labeled data, the representation learning power is increased from the information learned with unsupervised learning with the remaining data without labels and thus together lead to excellent classification results, comparable to the case where data has complete labels. Taken together, we found DSSDA is promising as a method for genomic data sets.

6.2 Input Representation with Probabilistic Continuous Vector-Space Model

The input representation with the neural probabilistic continuous vector space model is found to be efficient with respect to the better performance and convergence behavior. Of course, this is in the case when computational cost is not considered as the top priority. The increase in time for a training iteration is due to representing each 3-mer by a vector of size 16, which causes the input to become almost 16 times larger and increase the time needed to look up each 3-mer from the hash table. As a result, although the model converged in less number of epochs, the total time needed for training is almost 6 times longer for the same model and batch size, and this is without even considering the time needed for training the *word2vec* model.

Nonetheless, in Figure 6, we visualize the learned distribution for 3-mers by projecting them onto 2-dimensional coordinates using *t-SNE* technique[25]. Figure 6 proves our hypothesis about the existence of non-random semantics and relationships among different 3-mers since some are projected closer than others in 2-dimensional space. This is seemingly a key factor for the performance gain and training convergence, which is ignored with the one-hot vector input representation. In other words, learning proper manifolds for machine learning tasks might be beneficial if the input is represented with such vectors, as evidenced with successes of the same word embedding techniques for human language modeling.

In this line of direction, we could further consider multiple k-mers[9], instead of using a single k-mer, as multiple channels in input. The question is whether the required additional computing cost is practically justified for potential improvement in accuracy. Nonetheless, it is theoretically intriguing for detecting various regulatory motifs occurring in different length scales[9].

7 CONCLUDING REMARKS

A complete understanding of transcriptional mechanisms is a grand challenge in biology, and advancing novel application tools like DSSDA would contribute research efforts to tackle it effectively and efficiently. Our platform is promising, as shown in this work, to analyze the sequence patterns of DHSs across multiple conditions. This capacity can be easily extended to analyze other types of sequence data arising from RNA-Seq, DNA sequencing for whole genomes or exomes, ChIP-Seq, Methyl-Seq, and many others. We showed that the deep generative ConvNet that is part of DSSDA can have an excellent ability to perform supervised and semi-supervised learning with high performance. This suggests the potential of our approach for object detection and more challenging attribute-based learning (i.e., biological contexts such as specific type of motifs with different protein binding or cooperative roles of multiple DHSs). Surely, a complete understanding can be achieved when all aspects of regulation for transcription in genome, epigenome, transcriptome, and proteome are considered together, and our work contributes a promising platform for the case of sequence-dependent patterns with DHSs. Furthermore, we investigated different input representations for sequences and their effects on training ConvNets. In our future work, we plan to explore its potential for learning hidden attributes, such as biological nuance features, and efficiently detecting them, as well as to extend the platform for integrating multi-omics data sets.

ACKNOWLEDGMENTS

This work was partially funded by NIH grants (P20GM103458-10, P30GM110760-03, P20GM103424), NSF grants (MRI-1338051, IBSS-L-1620451, SCC-1737557, RAPID-1762600), LA Board of Regents grants (LEQSF(2016-19)-RD-A-08 and ITRS), and IBM faculty awards. The authors would like to thank NVIDIA for its generosity in allowing them to use their compute resources.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* (2016).

[2] Babak Alipanahi, Andrew Delong, Matthew T Weirauch, and Brendan J Frey. 2015. Predicting the sequence specificities of DNA-and RNA-binding proteins by deep learning. *Nature biotechnology* 33, 8 (2015), 831–838.

[3] Timothy L Bailey, Mikael Boden, Fabian A Buske, Martin Frith, Charles E Grant, Luca Clementi, Jingyuan Ren, Wilfred W Li, and William S Noble. 2009. MEME SUITE: tools for motif discovery and searching. *Nucleic acids research* 37, suppl_2 (2009), W202–W208.

[4] Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors.. In *ACL (1)*. 238–247.

[5] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence* 35, 8 (2013), 1798–1828.

[6] Yoshua Bengio, Li Yao, Guillaume Alain, and Pascal Vincent. 2013. Generalized denoising auto-encoders as generative models. In *Advances in Neural Information Processing Systems*. 899–907.

[7] ENCODE Project Consortium et al. 2012. An integrated encyclopedia of DNA elements in the human genome. *Nature* 489, 7414 (2012), 57–74.

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 248–255.

[9] Mahmoud Ghandi, Dongwon Lee, Morteza Mohammad-Noori, and Michael A Beer. 2014. Enhanced regulatory sequence prediction using gapped k-mer features. *PLoS computational biology* 10, 7 (2014), e1003711.

[10] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

[11] Eduardo G Gusmao, Manuel Allhoff, Martin Zenke, and Ivan G Costa. 2016. Analysis of computational footprinting methods for DNase sequencing experiments. *Nature methods* (2016).

[12] Hamid Reza Hassanzadeh and May D Wang. 2016. DeeperBind: Enhancing prediction of sequence specificities of DNA binding proteins. In *Bioinformatics and Biomedicine (BIBM), 2016 IEEE International Conference on*. IEEE, 178–183.

[13] Housheng Hansen He, Clifford A Meyer, Mei-Wei Chen, Chongzhi Zang, Yin Liu, Prakash K Rao, Teng Fei, Han Xu, Henry Long, X Shirley Liu, et al. 2014. Refined DNase-seq protocol and data analysis reveals intrinsic bias in transcription factor footprint identification. *Nature methods* 11, 1 (2014), 73–78.

[14] Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *science* 313, 5786 (2006), 504–507.

[15] David R Kelley, Jasper Snoek, and John L Rinn. 2016. Basset: learning the regulatory code of the accessible genome with deep convolutional neural networks. *Genome research* 26, 7 (2016), 990–999.

[16] Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. 2014. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*. 3581–3589.

[17] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[19] Anshul Kundaje, Wouter Meuleman, Jason Ernst, Misha Bilenky, Angela Yen, Alireza Heravi-Moussavi, Pouya Kheradpour, Zhizhuo Zhang, Jianrong Wang, Michael J Ziller, et al. 2015. Integrative analysis of 111 reference human epigenomes. *Nature* 518, 7539 (2015), 317–330.

[20] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444.

[21] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. 1989. Backpropagation applied to handwritten zip code recognition. *Neural computation* 1, 4 (1989), 541–551.

[22] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[23] Dongwon Lee, David U Gorkin, Maggie Baker, Benjamin J Strober, Alessandro L Asoni, Andrew S McCallion, and Michael A Beer. 2015. A method to predict the impact of regulatory variants from DNA sequence. *Nature genetics* 47, 8 (2015), 955.

[24] Lars Maaløe, Casper Kaae Sønderby, Søren Kaae Sønderby, and Ole Winther. 2016. Auxiliary deep generative models. *arXiv preprint arXiv:1602.05473* (2016).

[25] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9, Nov (2008), 2579–2605.

[26] Matthew T Maurano, Richard Humbert, Eric Rynes, Robert E Thurman, Eric Haugen, Hao Wang, Alex P Reynolds, Richard Sandstrom, Hongzhu Qu, Jennifer Brody, et al. 2012. Systematic localization of common disease-associated variation in regulatory DNA. *Science* 337, 6099 (2012), 1190–1195.

[27] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[28] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[29] Mohammad Pezeshki, Linxi Fan, Philemon Brakel, Aaron Courville, and Yoshua Bengio. 2016. Deconstructing the ladder network architecture. In *International Conference on Machine Learning*. 2368–2376.

[30] Antti Rasmus, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. 2015. Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems*. 3546–3554.

[31] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082* (2014).

[32] Jürgen Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural networks* 61 (2015), 85–117.

[33] Richard I Sherwood, Tatsunori Hashimoto, Charles W O'donnell, Sophia Lewis, Amira A Barkal, John Peter Van Hoff, Vivek Karun, Tommi Jaakkola, and David K Gifford. 2014. Discovery of directional and nondirectional pioneer transcription factors by modeling DNase profile magnitude and shape. *Nature biotechnology* 32, 2 (2014), 171–178.

[34] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[35] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1–9.

[36] Robert E Thurman, Eric Rynes, Richard Humbert, Jeff Vierstra, Matthew T Maurano, Eric Haugen, Nathan C Sheffield, Andrew B Stergachis, Hao Wang, Benjamin Vernot, et al. 2012. The accessible chromatin landscape of the human genome. *Nature* 489, 7414 (2012), 75–82.

[37] Harri Valpola. 2015. From neural PCA to deep unsupervised learning. *Advances in Independent Component Analysis and Learning Machines* (2015), 143–171.

[38] Jason Weston, Frédéric Ratle, Hossein Mobahi, and Ronan Collobert. 2012. Deep learning via semi-supervised embedding. In *Neural Networks: Tricks of the Trade*. Springer, 639–655.

[39] Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision*. Springer, 818–833.

[40] Haoyang Zeng, Matthew D Edwards, Ge Liu, and David K Gifford. 2016. Convolutional neural network architectures for predicting DNA–protein binding. *Bioinformatics* 32, 12 (2016), i121–i127.

[41] Jian Zhou and Olga G Troyanskaya. 2015. Predicting effects of noncoding variants with deep learning-based sequence model. *Nature methods* 12, 10 (2015), 931–934.