A Predictive Model of Gene Expression Using a Deep Learning Framework

Rui Xie *, Andrew Quitadamo †, Jianlin Cheng * and Xinghua Shi †
* Department of Computer Science, University of Missouri at Columbia
Columbia, MO, 65201, USA. Email: {rxpkd, chengji}@missouri.edu
† Department of Bioinformatics and Genomics, University of North Carolina at Charlotte
Charlotte, NC 28223, USA. Email: {aquitada, x.shi}@uncc.edu

***Abstract*—With an unprecedented amount of data available, it is important to explore new methods for developing predictive models to mine this data for scientific discoveries. In this study, we propose a deep learning regression model based on MultiLayer Perceptron and Stacked Denoising Auto-encoder (MLP-SAE) to predict gene expression from genotypes of genetic variation. Specifically, we use a stacked denoising auto-encoder to train our regression model in order to extract useful features, and utilize the multilayer perceptron for backpropagation. We further improve our model by adding a dropout technique to prevent overfitting. Our results on a real genomic dataset show that our MLP-SAE model with dropout outperform Lasso, Random Forests, and MLP-SAE without dropout. Our study provides a new application of deep learning in mining genomics data, and demonstrates that deep learning has great potentials in building predictive models to help understand biological systems.**

I. Introduction

Individuals of a species harbor genetic differences that can lead to phenotypic differences including different levels of gene expression. Genetic variation also provides an important repertoire for natural selection. There are many different types of genetic variation including single nucleotide polymorphisms (SNPs) which are one base pair substitutions, small insertions or deletions (indels), and structural variants (SVs) which can be as large as whole chromosomes. The biological community has generated genotypes of genetic variation in populations of different organisms. Two genetic variation databases at National Center for Biotechnology Information (NCBI), namely dbSNP [1] and dbVar [2], respectively store short genetic variation (SNPs and indels), and larger structural variation of various organisms. Studies have shown that genetic variants are associated with phenotypic traits ranging from growth to disease, but also molecular traits such as gene expression.

Expression quantitative trait locus (eQTL) mapping [3], [4] has been widely used to study the influence of genetic variants on gene expression in yeast [5], [6] and many other organisms. Genomic data is typically high dimensional, where the number of genetic variants or genes is much larger than the number of samples. The high dimensionality poses significant challenges for eQTL mapping due to the curse of dimensionality. Traditional eQTL frameworks use linear regression and correlation analysis to assess each pair of genetic variant and gene expression, and apply multi-test correction afterwards. Recently, machine learning methods emerging as powerful alternatives for performing eQTL anlaysis. Studies have shown that modern methods like Random Forests outperform legacy eQTL methods [7]. The model of Random Forests [8], [9], [10] is an ensemble learning method for classification, regression and other tasks. By constructing a multitude of decision trees at training time, Random Forests are capable of performing classification or mean prediction (regression) of the individual trees. In addition, Random Forests correct for decision trees' habit of overfitting to training sets[11].

Another set of approaches stem from Lasso [12] and multi-task Lasso [13], [14], [15].Lasso is a shrinkage and selection method for linear regression [16]. It minimizes the usual sum of squared errors, with a bound on the sum of the absolute values of the coefficients. Lasso is useful in some situations because of its tendency to prefer solutions with fewer parameter values. Therefore, Lasso simultaneously produces both an accurate and sparse model, which makes it an appropriate method for eQTL analysis.

In addition to providing improved eQTL mapping, these machine learning based models have potentials for building a predictive model of inferring gene expression from genetic variant genotypes. With this in mind, we plan to explore emerging methods in machine learning to address this biological question of predicting gene expression from genotypes.

Since the seminal work in [17], deep learning methods are breaking records in the areas of speech, signal, image, video and text mining and recognition by significantly improving state of the art classification accuracy [18]. With the deluge of big data in genomics, deep learning has the potential to model the hierarchical representations of the biological mechanisms encoded in rich datasets. Recently, we have witnessed a growing interest in applying deep learning models to genomic studies. For example, a deep neural network was developed to predict splicing patterns in individual tissues and differences in splicing patterns across tissues, inferred from mouse RNASeq data [19]. Other studies have used deep learning models to predict protein contact map [20], protein residue-residue contacts [21], [22], protein disorderedness [23], [24], protein secondary structures [25], [26], [27], protein properties [28], protein fold recognition [29], the functional effect of non-coding variants [30], the pathogenicity of variants [31], and the regulatory code of genomes [32], [33].

However, there is limited research in applying deep learning to predict a quantitative trait from genetic variation. In this

paper we develop a deep learning model based on MultiLayer Perceptron and Stacked Denoising Auto-encoder (MLP-SAE) to build a predictive model of gene expression from genotypes. Instead of using traditional neural networks, we propose a new method to use a a local unsupervised criterion to pre-train each layer in turn, and produce a useful higher-level representation from the lower-level representation output by the previous layer with stacked denoising auto-encoder. We employ the auto-encoder as hidden layers in a neural network for backpropagation. We further improve our model by adding a dropout technique to prevent overfitting. We evaluate the performance of our model, both with and without dropout, and compare it to Lasso and Random Forests. Our comparison analysis of these models suggests that our MLP-SAE model with dropout performs the best. We then apply this MLP-SAE model with dropout to a yeast data set and build a model to predict yeast gene expression from its genetic variants. Our study provides a software package that allows users to train the model with their own data and make predictions of various quantitative traits in not only yeast but also other organisms.

The rest of the paper is organized as follows. In the II, we formulate the problem and introduce the deep learning method based on MultiLayer Perceptron and Stacked Denoising Auto-Encoder. In III, we present our results on a real yeast genomic data and evaluate the performance of our model by comparing our method with two other methods. We then conclude the paper and point to future directions in Section IV.

II. Methods

To build a model for predicting gene expression from genotypes, we introduce a model based on the Multilayer Perceptron (MLP) and Stacked Denoising Auto-Encoder (DAE) as illustrated in Figure 1. The input data is SNP genotypes (features) which are fed into the model after pre-processing. The output layer is based on a regression model with the quantitatively predicted gene expression values as the output. The auto-encoder1 and auto-encoder2 serve as hidden layers for the prediction model and are trained using back propagation. The model is trained with given features and optimized through a backpropagation algorithm. The first training step is based on training the auto encoder with a stochastic gradient descent algorithm and the second training step utilizes the two auto encoders as two hidden layers and training them with MLP. A backpropagation algorithm is applied for optimization. After training, we use cross validation to select the optimal model and evaluate its performance on an independent data set. We then compare the results of our model with those from different methods including Lasso and Random Forests. Our model is implemented using pylearn2[34].

A. Data and Pre-processing

To evaluate and demonstrate the application of our model, we used a real genomic dataset on yeast widely used in the community. The genotypes and gene expression quantifications are measured using microarrays for 112 yeast samples [35]. We represent genotypes and gene expressions as input and output matrices for modeling. We extract the genotypes

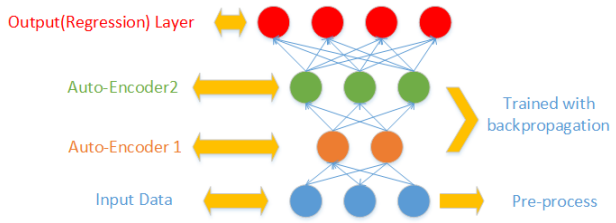

Fig. 1. An Illustration of Our Model for MLP and DAE.

of 2,956 SNPs and gene expression profiles of 7,085 genes in the 112 samples. We remove genes with 100% missing values in the expression matrix that result in 6,611 genes for modeling. To ensure model performance, we pre-process genotype data using the Scikit-Learn toolkit[36] including Imputer and MinMaxScaler[37]. These tools handle missing genotype data to minimize their negative effect during training. They also scale and translate each feature individually such that the data is in the given range on the training set, i.e. between zero and one.

B. Deep Learning Regression Model

Since we handle the quantitative output of continuous values of gene expressions, we use a regression model. The linear regression model of the final layer of our model can be represented as Equation 1:

$$f(x) = \omega^T x + b \tag{1}$$

where x represents the input or features. The w matrix is the weight and b is the bias, and are trained to minimize the objective function. The model first processes the input data and then performs pre-training. When it reaches the output layer, the model is finetuned with backpropagation. The algorithm stops when it reaches convergence.

C. Multilayer Perceptron

A Multilayer Perceptron (MLP) is a feedfoward network that tries to map input onto output. An MLP has multiple layers of nodes where each layer is fully connected with the next one. Each node of the hidden layers is operated with a nonlinear activation function. A backpropagation algorithm is used to train the network[38]. Now let's explain activation functions for training and learning through backpropagation.

1) Activation Function: Two activation functions are used for training. With a range from -1 to 1, a hyperbolic tangent is used as described in Equation 2.

$$y(v_i) = tanh(v_i) \tag{2}$$

Within the range from 0 to 1, a logistic function is used as seen in Equation 3.

$$y(v_i) = (1 + e^{-v_i})^{-1} \tag{3}$$

Here y_i is the output of the ith node (neuron) and v_i is the weighted sum of the input synapses.

2) Learning Through Backpropagation: After the data of each neuron is processed, an MLP network is learned through changing connection weights. With the amount of error in the output compared with the expected result, we are able to perform supervised learning.

Here is a simple example. We calculate error e in node j in the nth row of the training data by Equation 4.

$$error_j(n) = d_j(n) - y_j(n) \tag{4}$$

where y is the target value and d is the expected value. We then make corrections based on those corrections that minimize errors on the entire output, given by Equation 5.

$$\varepsilon(n) = \frac{1}{2}\sum_i error_j^2(n) \tag{5}$$

After gradient descent, the needed change for each weight can be represented as in Equation 6.

$$\Delta\omega_{ji}(n) = -\eta\frac{\partial\varepsilon(n)}{\partial v_j(n)}y_i(n) \tag{6}$$

where the output of the previous neuron is denoted as y and the learning rate is represented by η.

With a variety of induced local fields, the derivative can be calculated. The derivative for an output node is represented as in Equation 7.

$$-\frac{\partial\varepsilon(n)}{\partial v_j(n)} = error_j(n)\phi'(v_j(n)) \tag{7}$$

where ϕ' is the derivative of the activation function described above that does not vary. The analysis is more difficult for the change in weights to a hidden node, but it can be shown that the relevant derivative is represented by Equation 8.

$$-\frac{\partial\varepsilon(n)}{\partial v_j(n)} = \phi'(v_j(n))\sum_k -\frac{\partial\varepsilon(n)}{\partial v_k(n)}\omega_{kj}(n) \tag{8}$$

This relative derivative depends on the change in node weights, which is represented in the output layer. Therefore, in order to change the hidden layer weights, we must first change the output layer weights according to the derivative of the activation function. This analysis thus represents a backpropagation of the activation function.

D. Stacked Denoising Auto-encoder

An Auto-encoder [39] is a type of neural network that helps learning efficient codings. The main goal of an auto-encoder is to learn a compressed, distributed representation (encoding) for a set of data, typically for the purpose of dimensionality reduction. For each autoencoder, the network tries to reproduce the provided input data by using supervised learning, thus the backpropagation method is a suitable method in the supervised training of multi-layer networks[40].

1) Pre-training: Similar to the Multilayer Perceptron (MLP), a simple form of an autoencoder is a feedforward and non-recurrent neural net [41], with an input layer, an output layer and one or more hidden layers connecting them. Although an MLP is similar to a neural network, there is a difference between an MLP and an autoencoder. For an autoencoder, the output layer has the same number of nodes as the input layer. Instead of training the model to predict a target value y given input x, an autoencoder is trained to reconstruct its own input x through minimizing its objective function. The training algorithm can be summarized as follows:

- For each input x, do a feed-forward pass to compute the values of all nodes in hidden layers after activation, and obtain an output $\widehat{x}$ at the output layer.
- Measure the deviation of $\widehat{x}$ from input x (a common method is to use a squared error)
- Backpropagate the error through the network and perform weight updates (a common method is a stochastic gradient descent algorithm).

In conclusion, the activations of the final hidden layer can be regarded as a compressed representation of the input if the hidden layers have fewer nodes than the input/output layers. In addition, activation functions that are commonly used in MLPs can be used in autoencoders. Moreover, the optimal solution to an auto-encoder is strongly related to principal component analysis (PCA) [42] if linear activations or only a single sigmoid hidden layer are used[43]. An autoencoder can potentially learn the identity function and become useless, when the hidden layers are larger than the input layer. However, such autoencoders might still learn useful features [44]. As shown in Figure 2, input nodes are corrupted via process q. Then the encoder tries to map the corrupted input to Y via process f. Afterwards, Y can be reconstructed via process g_θ, resulting reconstructed Z. Finally, the reconstruction error is measured by $L_\theta(X, Z)$ as described next.

2) Corrupted Level: It is common to add noises to a model, so that the data is shuffled and a denoising auto-encoder can learn about that data by attempting to reconstruct it. The main goal of the model then is to tease out data features from noises. During training, a model is generated and an objective function is evaluated. Common objective functions include squared errors that measure the distance between that model and the benchmark. The objective function can be represented by $L(X, Z)$ or $L_H(X, Z)$ between the original X and a reconstruction Z, as in Equation 9 or Equation 10.

$$L(X, Z) = ||X - Z||^2 \tag{9}$$

$$L_H(X, Z) = -\sum_k^d [x_k log z_k - (1 - x_k) log(1 - log z_k)] \tag{10}$$

Equation 9 denotes the squared error objective for a real value X while Equation 10 represents the cross-entropy objective for a binary X[45]. Both methods attempt to minimize the loss function by resampling the shuffled input and reconstructing data, until it finds the input which brings its model closest to the truth. Figure 2 illustrates a corruption model, where the raw input X is corrupted via process q. Black nodes denote the corrupted input. Then the corrupted input is changed to Y via process f_θ. After that, Y tries to reconstruct the raw input via process g_θ, that results in the reconstructed Z. We then backpropagate via the reconstruction error $L_H(X, Z)$.

With randomly initialized parameters, a traditional MLP does not perform well by directly optimizing the supervised objective of interest through algorithms such as gradient descent. A more efficient way to achieve a better performance is to use a a local unsupervised criterion to pre-train each layer in turn, and produce a useful higher-level representation from the lower-level representation output by the previous layer. Thus, the gradient descent on the supervised objective leads to much better solutions in terms of generalization performance.[45]

E. Improving the Model by Applying Dropout

Although an MLP-SAE can produce comparable results when compared with other methods, we need to better control

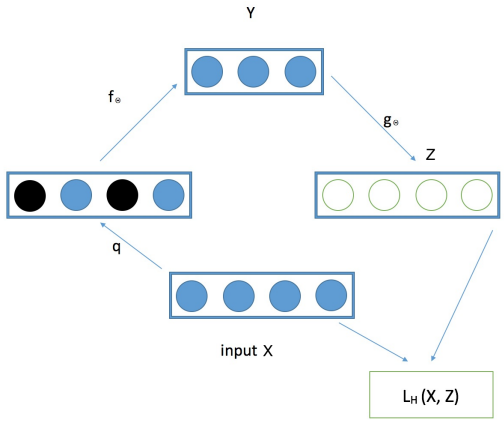

Fig. 2. An Illustration of an Auto-encoder Corruption Model.

overfitting in order to improve the performance. One strategy is by applying dropout, which prevents overfitting and provides a way of approximately combining exponentially many different neural network architectures efficiently [46]. The term dropout refers to drop out units (hidden and visible) in a neural network. To drop out a unit is to temporarily remove it from the network, along with all its incoming and outgoing connections. The choice of which units to drop is random. In the simplest case, each unit is retained with a probability p independent of other units, where p can be chosen using a validation set or simply set at 0.5, which seems to be close to optimal for a wide range of networks and tasks. For the input units, however, the optimal probability of retention is usually closer to 1 than to 0.5.

Hence, an intuitive goal of dropout regularization is to approximate the following concept: (1) Ignore units and their associated weights by a probability p for a particular training sample, and train with back propagation; (2) Repeat (ignoring any other random set of units and then train) and train training samples; (3) Average the weights across all these modified structures when doing predictions on new samples. In this study, we explore the multilayer perceptron combined with a stacked denoising autoencoder, with and without dropout to evaluate their performance.

III. Results

After establishing our model, we compare the multilayer perceptron with a stacked denoising autoencoder (MLP-SAE), with and without dropout, with other methods including Lasso and Random Forests, to evaluate their performance on a real yeast dataset. In our experimental setup, we split the dataset into three datasets, with a training dataset and validation dataset to be used in training phase, and an independent test dataset which does not participate in any training to avoid overfitting. Additionally, we take part of the training dataset as a validation dataset, which does not participate in training, and use five-fold cross validation on the training dataset to obtain our optimal model. Finally, we apply the trained model with learned parameters to an independent test dataset to obtain and compare the predictive results. To compare the performance of different models, we used mean square errors (MSE) for model evaluation defined in Equation 11.

$$MSE = \frac{1}{n}\sum_{i}^{n}(z_i - y_i)^2 \tag{11}$$

where z_i is the predicted output and y_i is the original output, with $i \in [1,n]$ (n is the number of samples).

Results suggest that our MLP-SAE model, with an optimal learning rate trained using cross-validation, outperform other classical methods like Lasso and Random Forests. The MLP-SAE model can be further improved using dropout. Since our model based on deep learning can easily incorporate evidence from other datasets such as epigenetic markers and functional elements, and achieve potentially scalable results on larger datasets, we argue that our study provides a new framework for building predictive models on genomic data.

A. Lasso and Random Forests

The first method for comparison is Lasso [12], which is a linear model with l_1 norm as regularizer. The objective function is to minimize the least-square penalty with an l_1 norm, as described in Equation 12.

$$\min \frac{1}{2n}||Xw - y||_2^2 + \alpha||w||_1 \tag{12}$$

where α is a constant and $||w||_1$ is the l_1-norm of the parameter vector. The sparsity of the model is controlled by α which can be learned through training. If α is large, the model is sparser and more coefficients will be shrunk to zero. Those features with non-zero coefficients will be selected as contributing features of the model. After cross validation, we use the optimal model trained to make predictions on our test dataset. Table I shows the MSEs of results by Lasso with different values of α after 5-fold cross validation.

TABLE I
MSE of the Results from Lasso with Varied α.

α	MSE
0.05	0.3516
0.1	0.3182
0.2	0.3002
0.3	0.2951
0.4	0.2930
0.5	0.2918
0.6	0.2914
0.7	0.2912
0.8	0.2912

The second method we compare with is Random Forests. Table II shows the MSE values of the results obtained from Random Forests after cross validation, with different number of estimators (i.e. trees). As expected, MSE values decrease and thus the model performs better as the number of estimators increase with the cost of increased computing resources.

B. MLP-SAE

We apply our model of Multilayer Perceptron with Stacked Denoising Auto-encoder (MLP-SAE) on the same dataset. Table III illustrates the MSE values of the results with different learning rates. According to cross validation, we select our optimal model with a learning rate of 0.1 on the yeast data. With this setting, our MLP-SAE model outperforms Lasso and Random Forest models.

TABLE II
MSE OF RESULTS FROM RANDOM FORESTS WITH VARIED NO. OF TREES.

Number of Estimators	MSE
10	0.3221
20	0.3127
30	0.3080
40	0.3001
50	0.2989
60	0.3003
70	0.2986
100	0.3003
150	0.2974
200	0.2967

TABLE III
MSE OF THE RESULTS ON MLP-SAE WITH VARIED LEARNING RATES.

Learning Rate	MSE
0.1	0.289001373
0.01	0.290938859
0.001	0.289527237
0.0001	0.290783006
0.00001	0.29175354

C. MLP-SAE with Dropout

Although the MLP-SAE model outperform other methods, we still need to better control overfitting to further improve its performance. We implement a MLP-SAE model with dropout[46] to handle overfitting. Our results show that dropout can produce better results on the same data set compared with MLP-SAE without dropout. The average MSE of MLP-SAE with Dropout is 0.3082, while that of MLP-SAE without dropout is 0.3093.

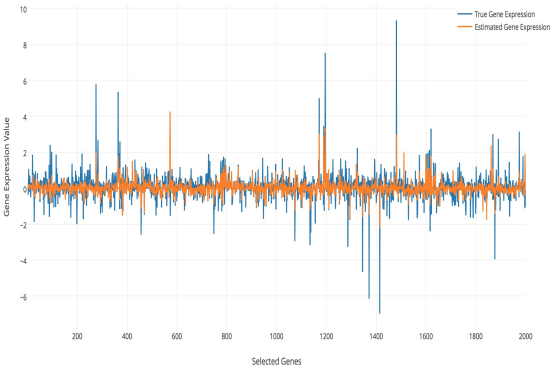

Fig. 3. Comparison between True and Estimated Gene Expression

Finally, we make predictions on the yeast data using a trained MLP-SAE model with dropout. Figure 3 visualizes true and our model estimated gene expression quantifications. Figure 4 zooms into a partial picture of the gene expression profiles that can be well predicted by our model. We observe that estimated values aligned well with true data. Although the true and estimated values might not be exactly the same for some genes, our model captures similar peaks in the data which suggests that the estimated gene expression quantifications encode similar up-regulated and down-regulated trend of gene expression, using only SNP genotypes as explanatory variables. We believe that our model can be further improved to recapitulate true gene expression signals by including more feature types (e.g. regulatory elements and pathways) and environmental conditions (e.g. media and temperature).

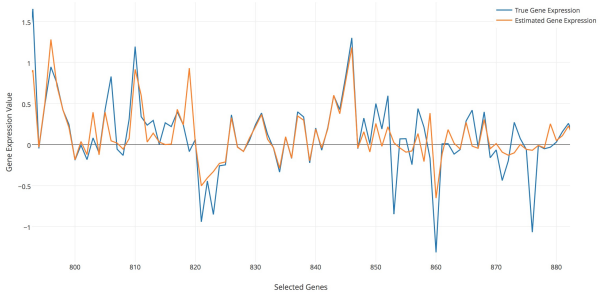

Fig. 4. Comparison between True and Estimated Gene Expression

To demonstrate that our model is scalable, we measure the training and prediction time usage of our model for different deep learning architecture as described in TableIV. The measurement is done on a computer with 7.8 GiB memory, Intel Core i7-2600 CPU 3.40GHz processor, Gallium 0.4 on AMD CEDAR graphics, 64-bit OS type and 445.5 GB disk. We can see that the time for prediction is far less than the training time. With a well trained model in hand, we believe that our model can handle big datasets with a large number of features and labels.

TABLE IV
TRAINING AND PREDICTION TIME COST FOR DIFFERENT CONFIGURATIONS.

Configuration	Training(s)	Prediction(s)
1500, 750	1683.24	4.65
2000, 1000	2350.67	6.13
3000, 1500	4045.95	7.782
4000. 2000	5884.088	10.99165

IV. CONCLUSION

In conclusion, we provide a new deep learning model, based on the Multilayer Perceptron with Stacked Denoising Auto-encoder (MLP-SAE) with dropout, that can be deployed to predict quantitative traits from genotype data. We use a stacked denoising auto-encoder to train our regression model in order to extract useful features, and then utilize the multilayer perceptron for backpropagation. Applying the MLP-SAE model to a well-established yeast dataset [35], we show that this model is well posed to predict gene expression values from SNP genotypes. We compare the MLP-SAE model with and without dropout, to other classical methods including Lasso and Random Forest. The comparison shows that MLP-SAE with dropout improves the performance of the MLP-SAE model and outperforms other models.

In this study, we demonstrate our method in a scenario where gene expression values are predicted from SNP genotypes, which implies the contribution of germline mutations to gene expression. Our model is built upon a flexible platform that is able to include other data types to further improve the model by including epigenetic, metabolic, and environmental factors. For example, protein quantifications [47], [48], metabolite screening [49], [50] and chromatin accessibility data [51], [52] are available for yeast. All these data can be directly incorporated and integrated in our model to predict a quantitative trait of interest, not limited to gene expression (e.g. growth rate).

Our study indicates that deep learning has great potential in solving predictive problems in biological systems. There are other deep learning architectures such as Restricted Boltzmann Machine [53] and the Recurrent Neural Network [54], which can be adopted to solve the quantitative trait prediction problem in this study. We anticipate vast room for improvement when more of the available data is used, and when a more detailed exploration of these architectures is performed.

ACKNOWLEDGMENT

This work is supported in part by US NIH grant (R01GM093123) and NSF grant (DBI1149224) to JX, and NSF grants (DGE-1523154 and IIS-1502172) to XS.

REFERENCES

[1] dbsnp: Database for short genetic variations. http://www.ncbi.nlm.nih.gov/snp/.

[2] dbvar: Database of genomic structural variation. http://www.ncbi.nlm.nih.gov/dbvar.

[3] Nica AC, Dermitzakis ET. Expression quantitative trait loci: present and future. *Philos Trans R Soc Lond B Biol Sci.*, 368(1620):20120362, 2013.

[4] Tian L, et al. Methods for population based eQTL Analysis in human genetics. *Tsinghua Science and Technology*, 19(6):624-634, 2014.

[5] Brem RB, et al. Genetic Dissection of Transcriptional Regulation in Budding Yeast. *Science*, 296(5568):752-755, 2002.

[6] Brem RB, et al. Genetic interactions between polymorphisms that affect gene expression in yeast. *436*, 296(5568):701703, 2005.

[7] Michaelson JJ, et al. Data-driven assessment of eqtl mapping methods. *BMC Genomics.*, 11:502, 2010.

[8] Breiman L. Random forests. *Machine learning*, 45(1):5–32, 2001.

[9] Breiman L, et al. Arcing classifier (with discussion and a rejoinder by the author). *The annals of statistics*, 26(3):801–849, 1998.

[10] Geurts P, et al. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

[11] Hastie T, et al. *The Elements of Statistical Learning*. Springer Series in Statistics. 2001.

[12] Tibshirani R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58(1):267–288, 1996.

[13] Lee S, Xing EP. Leveraging input and output structures for joint mapping of epistatic and marginal eqtls. *Bioinformatics*, 28(12):i137–i146, 2012.

[14] Chen X, et al. A two-graph guided multi-task lasso approach for eqtl mapping. In *ICML*, pages 208–217, 2012.

[15] Cheng W, et al. Graph-regularized dual lasso for robust eqtl mapping. *Bioinformatics*, 30(12):i139–i148, 2014.

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

[17] Hinton GE, et al. A fast learning algorithm for deep belief nets. *Neural Computation.*, 18(7):1527-1554, 2006.

[18] Bengio Y, et al. Representation learning: a review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35:1798-1828, 2013.

[19] Leung MK, et al. Deep learning of the tissue-regulated splicing code. *Bioinformatics.*, 30(12):i121–9, 2014.

[20] Di Lena P, et al. Deep architectures for protein contact map prediction. *Bioinformatics.*, 28(19):2449–57, 2012.

[21] Eickholt J, Cheng J. Predicting protein residue-residue contacts using deep networks and boosting. *Bioinformatics.*, 28:3066–3072, 2012.

[22] Adhikari B., Cheng J. Protein Residue Contacts and Prediction Methods. *Methods Mol Biol.*, 1415:463–76, 2016.

[23] Eickholt J and Cheng J. DNdisorder: predicting protein disorder using boosting and deep networks. *BMC Bioinformatics.*, 14:88, 2013.

[24] Wang S, et al. DeepCNF-D: Predicting Protein Order/Disorder Regions by Weighted Deep Convolutional Neural Fields. *Int J Mol Sci.*, 16(8):17315–30, 2015.

[25] Zhou J,Troyanskaya O. Deep Supervised and Convolutional Generative Stochastic Network for Protein Secondary Structure Prediction. *Deep Learning Workshop NIPS.*, 2013.

[26] Wang S, et al. Protein Secondary Structure Prediction Using Deep Convolutional Neural Fields. *Sci Rep.*, 6:18962, 2016.

[27] Spencer M, et al. A Deep Learning Network Approach to ab initio Protein Secondary Structure Prediction. *IEEE/ACM Trans Comput Biol Bioinform.*, 12(1):103112, 2015.

[28] Wang S, et al. RaptorX-Property: a web server for protein structure property prediction. *Nucleic Acids Res.*, gkw306, 2016.

[29] Jo T, et al. Improving protein fold recognition by deep learning networks. *Scientific Reports.*, 5:17573, 2015.

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

[31] Quang D, et al. DANN: a deep learning approach for annotating the pathogenicity of genetic variants. *Bioinformatics.*, 31(5):761–3, 2015.

[32] Kelley DR, et al. Basset: Learning the regulatory code of the accessible genome with deep convolutional neural networks. *Genome Res.*, pii(gr.200535):115, 2016.

[33] Xu W, et al. SD-MSAEs: Promoter Recognition in Human Genome based on Deep Feature Extraction. *J Biomed Inform.*, pii:S1532–0464, 2016.

[34] Goodfellow IJ, et al. Pylearn2: a machine learning research library. *arXiv preprint arXiv:1308.4214*, 2013.

[35] Brem RB, Kruglyak L. The landscape of genetic complexity across 5,700 gene expression traits in yeast. *PNAS*, 102(5):1572–1577, 2005.

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

[37] Buitinck L, et al. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[38] Rumelhart DE, et al. Learning internal representations by error propagation. Technical report, 1985.

[39] Vincent P, et al. Extracting and composing robust features with denoising autoencoders. In *Proceedings of ICML*, pages 1096–1103. ACM, 2008.

[40] Ng A. Sparse autoencoder. *CS294A Lecture notes*, 72, 2011.

[41] Graves A. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[42] Richardson M. Principal component analysis. 2009.

[43] Bourlard H, Kamp Y. Auto-association by multilayer perceptrons and singular value decomposition. *Biolog. cybernetics*, 59(4):291-294, 1988.

[44] Bengio Y. Learning deep architectures for AI. *Foundations and trends in Machine Learning*, 2(1):1–127, 2009.

[45] Vincent P, et al. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, 11:3371–3408, 2010.

[46] Srivastava N, et al. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[47] Albert FW, et al. Genetics of single-cell protein abundance variation in large yeast populations. *Nature.*, 506(7489):494–7, 2014.

[48] Picotti P, et al. A complete mass-spectrometric map of the yeast proteome applied to quantitative trait analysis. *Nature.*, 494(7436):266–70, 2013.

[49] Zhu J, et al. Stitching together Multiple Data Dimensions Reveals Interacting Metabolomic and Transcriptomic Networks That Modulate Cell Regulation. *PLoS Biology.*, 10:e1001301, 2012.

[50] Breunig JS, et al. Genetic basis of metabolome variation in yeast. *PLoS Genet.*, 10(3):e1004142, 2014.

[51] Connelly CF, et al. Evolution and genetic architecture of chromatin accessibility and function in yeast. *PLoS Genet.*, 10(7):e1004427, 2014.

[52] Lee K, et al. Genetic landscape of open chromatin in yeast. *PLoS Genet.*, 9(2):e1003229, 2013.

[53] Hinton G. A practical guide to training restricted boltzmann machines. *Momentum*, 9(1):926, 2010.

[54] Graves A, et al. Speech recognition with deep recurrent neural networks. In *Proceedings of ICASSP*, pages 6645–6649. IEEE, 2013.