OXFORD

Gene expression

Gene expression inference with deep learning

Yifei Chen[1,4,†], Yi Li[1,†], Rajiv Narayan[2], Aravind Subramanian[2] and Xiaohui Xie[1,3,*]

[1]Department of Computer Science, University of California, Irvine, CA 92697, USA, [2]Broad Institute of MIT And Harvard, Cambridge, MA 02142, USA, [3]Center for Complex Biological Systems, University of California, Irvine, CA 92697, USA and [4]Baidu Research-Big Data Lab, Beijing, 100085, China

*To whom correspondence should be addressed.

[†]The authors wish it to be known that, in their opinion, the first two authors should be regarded as joint First Authors.
Associate Editor: Inanc Birol

Abstract

Motivation: Large-scale gene expression profiling has been widely used to characterize cellular states in response to various disease conditions, genetic perturbations, etc. Although the cost of whole-genome expression profiles has been dropping steadily, generating a compendium of expression profiling over thousands of samples is still very expensive. Recognizing that gene expressions are often highly correlated, researchers from the NIH LINCS program have developed a cost-effective strategy of profiling only ~1000 carefully selected landmark genes and relying on computational methods to infer the expression of remaining target genes. However, the computational approach adopted by the LINCS program is currently based on linear regression (LR), limiting its accuracy since it does not capture complex nonlinear relationship between expressions of genes.

Results: We present a deep learning method (abbreviated as D-GEX) to infer the expression of target genes from the expression of landmark genes. We used the microarray-based Gene Expression Omnibus dataset, consisting of 111K expression profiles, to train our model and compare its performance to those from other methods. In terms of mean absolute error averaged across all genes, deep learning significantly outperforms LR with 15.33% relative improvement. A gene-wise comparative analysis shows that deep learning achieves lower error than LR in 99.97% of the target genes. We also tested the performance of our learned model on an independent RNA-Seq-based GTEx dataset, which consists of 2921 expression profiles. Deep learning still outperforms LR with 6.57% relative improvement, and achieves lower error in 81.31% of the target genes.

Availability and implementation: D-GEX is available at https://github.com/uci-cbcl/D-GEX.

Contact: xhx@ics.uci.edu

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

A fundamental problem in molecular biology is to characterize the gene expression patterns of cells under various biological states. Gene expression profiling has been historically adopted as the tool to capture the gene expression patterns in cellular responses to diseases, genetic perturbations and drug treatments. The Connectivity Map (CMap) project was launched to create a large reference collection of such patterns and has discovered small molecules that are functionally connected using expression pattern-matching (e.g. HDAC inhibitors and estrogen receptor modulators) (Lamb *et al.*, 2006).

Although recent technological advances, whole-genome gene expression profiling is still too expensive to be used by typical academic labs to generate a compendium of gene expression over a large number of conditions, such as large chemical libraries, genome-wide RNAi screening and genetic perturbations. The initial phase of the CMap project produced only 564 genome-wide gene

expression profiles using Affymetrix GeneChip microarrays (Lamb *et al.*, 2006).

Despite the large number of genes (∼22 000) across the whole human genome, most of their expression profiles are known to be highly correlated. Systems biologists have leveraged this idea to construct gene regulatory networks and to identify regulator and target genes (Bansal *et al.*, 2007). Researchers from the LINCS program (http://www.lincsproject.org/) analyzed the gene expression profiles from the CMap data using principal component analysis. They found that a set of ∼1000 carefully chosen genes can capture ∼80% of the information in the CMap data (http://support.lincscloud.org/hc/en-us/articles/202092616-The-Landmark-Genes). Motivated by this observation, researchers have developed the L1000 Luminex bead technology to measure the expression profiles of these ∼1000 genes, called the *landmark genes* (http://support.lincscloud.org/hc/en-us/articles/202092616-The-Landmark-Genes), with a much lower cost (∼$5 per profile) (Peck *et al.*, 2006). Therefore, researchers can use the expression signatures of landmark genes to characterize the cellular states of samples under various experimental conditions. If researchers are interested in the expression of a specific gene other than landmark genes, the expression profiles of the remaining ∼21 000 genes, called the *target genes*, can be then computationally inferred based on landmark genes and existing expression profiles. With the L1000 technology, the LINCS program has generated ∼1.3 million gene expression profiles under a variety of experimental conditions.

However, computationally inferring the expression profiles of target genes based on landmark genes is challenging. It is essentially a large-scale multi-task machine learning problem, with the target dimension (∼21 000) significantly greater than the feature dimension (∼1000). The LINCS program currently adopts linear regression (LR) as the inference method, which trains regression models independently for each target gene based on the Gene Expression Omnibus (GEO) (Edgar *et al.*, 2002) data. While LR is highly scalable, it inevitably ignores the nonlinearity within gene expression profiles that has been observed (Hasty *et al.*, 2001). Kernel machines can represent dexterous nonlinear patterns and have been applied to similar problems (Ye *et al.*, 2013). Unfortunately, they suffer from poor scalability to growing data size. Thus, a machine learning method enjoying both scalability and rich representability is ideal for large-scale multi-task gene expression inference.

Recent successes in deep learning on many machine learning tasks have demonstrated its power in learning hierarchical nonlinear patterns on large-scale datasets (Bengio *et al.*, 2013). Deep learning in general refers to methods that learn a hierarchical representation of the data through multiple layers of abstraction (e.g. multi-layer feedforward neural networks). A number of new techniques have been developed recently in deep learning, including the deployment of General-Purpose Computing on Graphics Processing Units (Ciresan *et al.*, 2012; Coates *et al.*, 2013), new training methodologies, such as dropout training (Baldi and Sadowski, 2013; Hinton *et al.*, 2012b) and momentum method (Sutskever *et al.*, 2013). With these advances, deep learning has achieved state-of-the-art performances on a wide range of applications, both in traditional machine learning tasks such as computer vision (Krizhevsky *et al.*, 2012), natural language processing (Socher *et al.*, 2011), speech recognition (Hinton *et al.*, 2012a) and in natural science applications such as exotic particles detection (Baldi *et al.*, 2014), protein structure prediction (Di Lena *et al.*, 2012), RNA splicing prediction (Leung *et al.*, 2014) and pathogenic variants identification (Quang *et al.*, 2014).

Here, we present a deep learning method for gene expression inference (D-GEX). D-GEX is a multi-task multi-layer feedforward

neural network. We evaluated the performances of D-GEX, LR (with and without different regularizations) and *k*-nearest neighbor (KNN) regression on two types of expression data, the microarray expression data from the GEO and the RNA-Seq expression data from the Genotype-Tissue Expression (GTEx) project (Ardlie *et al.*, 2015; Lonsdale *et al.*, 2013). GPU computing was used to accelerate neural network training so that we were able to evaluate a series of neural networks with different architectures. Results on the GEO data show that D-GEX consistently outperforms other methods in terms of prediction accuracy. Results on the GTEx data further demonstrate D-GEX, combined with the dropout regularization technique, achieves the best performance even where training and prediction were performed on datasets obtained from different platforms (microarray versus RNA-Seq). Such cross platforms generalizability implies the great potential of D-GEX to be applied to the LINCS program where training and prediction were also done separately on the microarray data and the L1000 data. Finally, we attempted to explore the internal structures of the learned neural networks with two different strategies and tried to interpret the advantages of deep learning compared with LR.

2 Methods

In this section, we first introduce three expression datasets we used in this study and formulate gene expression inference as a supervised learning problem. We then present D-GEX for this problem and explain a few key deep learning techniques to train D-GEX. Finally, we introduce several common machine learning methods that we used to compare with D-GEX.

2.1 Datasets

The GEO expression data were curated by the Broad Institute from the publicly available GEO database. It consists of 129 158 gene expression profiles from the Affymetrix microarray platform. Each profile comprises of 22 268 probes, corresponding to the 978 landmark genes and the 21 290 target genes. The original GEO data were accessed from the LINCS Cloud (http://www.lincscloud.org/), which has been quantile normalized into a numerical range between 4 and 15. Some of the expression profiles in the GEO dataset are biological or technical replicates. To avoid complications in the learning procedure, we removed duplicated samples (see Supplementary material), leaving 111 009 profiles in the end.

The GTEx expression data consist of 2921 gene expression profiles of various tissue samples obtained from the Illumina RNA-Seq platform (Ardlie *et al.*, 2015). The expression level of each gene was measured based on Gencode V12 annotations (Ardlie *et al.*, 2015) in the format of Reads Per Kilobase per Million (RPKM).

The 1000 Genomes expression data consist of 462 gene expression profiles of lymphoblastoid cell line samples from the Illumina RNA-Seq platform (Lappalainen *et al.*, 2013). The expression level of each gene was also measured based on Gencode V12 annotations (Lappalainen *et al.*, 2013) in the format of RPKM.

Since the gene expression values of the microarray platform and the RNA–Seq platform were measured in different units (probes versus Gencode annotations) and different numerical scales, we quantile normalized the three expression datasets jointly to retain the maximum information cross platforms. Because one Gencode annotation may include multiple microarray probes, 943 landmark genes and 9520 target genes in terms of Gencode annotations were left after joint quantile normalization. Details of joint quantile normalization are given in Supplementary materials. Finally, all the datasets were

standardized by subtracting the mean and dividing by the standard deviation of each gene.

2.2 Gene expression inference as multi-task regression

Assume there are L landmark genes, T target genes and N training samples (i.e. profiles); the training dataset is expressed as $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{N}$, where $\mathbf{x}_i \in \mathcal{R}^L$ denotes the expression values of landmark genes and $\mathbf{y}_i \in \mathcal{R}^T$ denotes the expression values of target genes in the ith sample. Our goal is to infer the functional mapping $\mathcal{F} : \mathcal{R}^L \rightarrow \mathcal{R}^T$ that fits $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{N}$, which can be viewed as a multi-task regression problem.

We use mean absolute error (MAE) to evaluate the predictive performance at each target gene t,

$$\text{MAE}_{(t)} = \frac{1}{N'} \sum_{i=1}^{N'} |y_{i(t)} - \hat{y}_{i(t)}|, \tag{1}$$

where N' is the number of testing samples and $\hat{y}_{i(t)}$ is the predicted expression value for target gene t in sample i. We define the overall error as the average MAE over all target genes, and use it to evaluate the general predictive performance.

For the microarray platform, we used the GEO data for training, validation and testing. Specifically, we randomly partitioned the GEO data into ~80% for training (88 807 samples denoted as GEO-tr), ~10% for validation (11 101 samples denoted as GEO-va) and ~10% for testing (11 101 samples denoted as GEO-te). The validation data GEO-va were used to do model selection and parameter tuning for all the methods.

For the RNA-Seq platform, we used GEO-tr for training, the 1000 Genomes data for validation (denoted as 1000G-va), and the GTEx data for testing (denoted as GTEx-te). The validation data 1000G-va was used to do model selection and parameter tuning for all the methods.

2.3 D-GEX

D-GEX is a multi-task multi-layer feedforward neural network. It consists of one input layer, one or multiple hidden layers and one output layer. All the hidden layers have the same number of hidden units. Units between layers are all fully connected. A hidden unit j in layer l takes the sum of weighted outputs plus the bias from the previous layer $l - 1$ as the input, and produces a single output o_j^l using a nonlinear activation function f.

$$o_j^l = f \left(\sum_{i=1}^{H} w_{i,j}^{l-1} o_i^{l-1} + b_j^{l-1} \right), \tag{2}$$

where H is the number of hidden units, $\{w_{i,j}^{l-1}, b_j^{l-1}\}_{i=1}^{H}$ are the weights and the bias associated with unit j that need to be learned. We adopt the hyperbolic tangent activation function to hidden units, which naturally captures the nonlinear patterns within the data. Linear activation function is applied to output units for the regression purpose. The loss function for training is the sum of mean squared error at each output unit, namely,

$$\mathcal{L} = \sum_{t=1}^{T} \left[\frac{1}{N} \sum_{i=1}^{N} \left(y_{i(t)} - \hat{y}_{i(t)} \right)^2 \right]. \tag{3}$$

D-GEX contains 943 units in the input layer corresponding to the 943 landmark genes. Ideally, we should also configure D-GEX with 9520 units in the output layer corresponding to the 9520 target genes. However, each of our GPUs has only 6 GB of memory, thus we cannot configure hidden layers with sufficient number of hidden units if all the target genes are included in one output layer.

Therefore, we randomly partitioned the 9520 target genes into two sets that each contains 4760 target genes. We then built two separate neural networks with each output layer corresponding to one half of the target genes. With this constraint, we were able to build a series of different architectures containing 1–3 hidden layers each and each hidden layer contains 3000, 6000 or 9000 hidden units. Supplementary Figure S1 shows an example of architecture of D-GEX with three hidden layers.

Training D-GEX follows the standard back-propagation algorithm (Rumelhart *et al.*, 1988) and mini-batch gradient descent, supplemented with advanced deep learning techniques. Detailed parameter configurations are given in Supplementary Table S1. For more descriptions about neural networks and their background please see Chen (2014). We discuss a few key training techniques as follows:

1. *Dropout* is a technique to perform model averaging and regularization (Hinton *et al.*, 2012b) for neural networks. At the training time, each unit along with its edges is temporarily dropped out with probability p for each training sample. Then the forward- and back-propagation are performed on a particularly 'thinned' network. For an architecture with n units performing dropout, there are $O\left(\frac{1}{(1-p)^n}\right)$ such thinned networks. At the testing time, all the units are retained with weights multiplied by $1 - p$. Therefore, dropout can be seen as model averaging of exponentially many different neural networks in an approximate but efficient framework. Dropout has been shown to suppress co-adaptation among units and force each unit to learn patterns that are more generalizable (Srivastava *et al.*, 2014). The dropout rate p serves as a tuning parameter that controls the intense of regularization. We applied dropout to all the hidden layers of D-GEX except for the outgoing edges from the input layer. The dropout rate was set to [0%, 10%, 25%] to compare the effect of different degrees of regularization.

2. *Momentum method* is a technique to accelerate gradient-based optimization. It accumulates a velocity in directions of gradients of the loss function across iterations and uses the velocity instead of the gradient to update parameters (Sutskever *et al.*, 2013). Given a loss function $\mathcal{L}$ with respect to the parameters Θ of the neural network, the momentum is given by

$$V^{(k+1)} = \mu V^{(k)} - \eta^{(k)} \nabla \mathcal{L}(\Theta^{(k)}),$$

$$\Theta^{(k+1)} = \Theta^{(k)} + V^{(k+1)}, \tag{4}$$

where $\mu \in [0, 1]$ is the momentum coefficient, η is the learning rate, V is the velocity and $\nabla \mathcal{L}(\Theta)$ is the gradient of the loss function. Momentum method has been shown to improve the convergence rate particularly for training deep neural networks (Sutskever *et al.*, 2013).

3. *Normalized initialization* is a technique to initialize the weights of deep neural networks (Glorot and Bengio, 2010). The weights of a unit is sampled from a uniform distribution defined by,

$$W \sim U \left[-\frac{\sqrt{6}}{\sqrt{n_i + n_o}}, \frac{\sqrt{6}}{\sqrt{n_i + n_o}} \right], \tag{5}$$

where n_i, n_o denote the number of fan-ins and fan-outs of the unit. It is designed to stabilize the variances of activation and back-propagated gradients during training (Glorot and Bengio, 2010). The uniform distribution of the output layer of D-GEX was set to be within a smaller range of $[-1 \times 10^{-4}, 1 \times 10^{-4}]$ as it was adopted with the linear activation function.

4. *Learning rate* was initialized to 5×10^{-4} or 3×10^{-4} depending on different architectures, and was decreased according to the

training error on a subset of GEO-tr for monitoring the training process. Specifically, the training error was checked after each epoch, if the training error increased, the learning rate was multiplied by a decay factor of 0.9 until it reached a minimum learning rate of 1×10^{-5}.

5. *Model selection* was performed based on GEO-va for the GEO data and 1000G-va for the GTEx data. Training was run for 200 epochs. The model was evaluated on GEO-va and 1000G-va after each epoch, and the model with the best performance was saved respectively.

D-GEX was implemented based on two Python libraries, Theano (Bergstra *et al.*, 2010) and Pylearn2 (Goodfellow *et al.*, 2013). Training was deployed on an Nvidia GTX TITAN Z graphics card with dual GPUs. The largest architecture of D-GEX (three hidden layers with 9000 hidden units in each hidden layer) contains ~427 million parameters. Training half of the target genes with the largest architecture took around 6 h. D-GEX is publicly available at https://github.com/uci-cbcl/D-GEX.

2.4 Linear regression

LR for multi-task gene expression inference trains a model, $\mathcal{F}_{(t)}(\mathbf{x}) = \mathbf{w}_{(t)}^T \mathbf{x} + b_{(t)}$, independently for each target gene t. $\mathbf{w}_{(t)} \in \mathcal{R}^L, b_{(t)} \in \mathcal{R}$ are the model parameters associated with each target gene t, and

$$\left(\mathbf{w}_{(t)}, b_{(t)}\right) = \arg \min_{\mathbf{w},b} \frac{1}{N} \sum_{i=1}^{N} \left(y_{i(t)} - \mathbf{w}_{(t)}^T \mathbf{x}_i - b_{(t)} \right)^2 \qquad (6)$$

L1 or L2 penalties can be further introduced for regularization purpose. In these cases,

$$\left(\mathbf{w}_{(t)}, b_{(t)}\right) = \arg \min_{\mathbf{w},b} \frac{1}{N} \sum_{i=1}^{N} \left(y_{i(t)} - \mathbf{w}_{(t)}^T \mathbf{x}_i - b_{(t)} \right)^2 + \lambda ||\mathbf{w}_{(t)}||_1 \quad (7)$$

or

$$\left(\mathbf{w}_{(t)}, b_{(t)}\right) = \arg \min_{\mathbf{w},b} \frac{1}{N} \sum_{i=1}^{N} \left(y_{i(t)} - \mathbf{w}_{(t)}^T \mathbf{x}_i - b_{(t)} \right)^2 + \lambda ||\mathbf{w}_{(t)}||_2 \quad (8)$$

LR (6) is currently adopted by the LINCS program. In our study, we evaluated both (6) and (7), (8) using scikit-learn (Pedregosa *et al.*, 2011). The regularization parameter λ was tuned based on the performance on GEO-va and 1000G-va.

2.5 KNN regression

KNN regression is a non-parametric and instance-based method. In standard KNN regression, a spatial data structure $\mathcal{T}$ such as the KD tree (Bentley, 1975) is built for training data in the feature space. Then, for any testing data, the k-nearest training samples based on a certain distance metric are queried from $\mathcal{T}$. The average of their values is computed as the prediction.

However, the standard KNN regression may be biased when duplicated samples frequently exist in the data, such as the GEO microarray data. Therefore, in gene expression inference, a commonly adopted alternative is to query the k-nearest genes rather than the k-nearest samples. Specifically, for each target gene, its Euclidean distances to all the landmark genes were calculated using the training samples. The k-landmark genes with the least Euclidean distances are determined as the k-nearest landmark genes of the target gene. Then the average of their expression values in the testing samples is computed as the prediction for the target gene. Such algorithm is also consistent with the basic assumption of the LINCS program that, the

expression of target genes can be computationally inferred from landmark genes. We call this algorithm the gene-based KNN (KNN-GE).

Due to the non-parametric and instance-based nature, KNN-GE does not impose any prior assumptions on the learning machine. Therefore, it is very flexible to model nonlinear patterns within the data. However, as performing prediction involves building and querying data structures that have to keep all the training data, KNN-GE suffers from poor scalability to growing data size and dimension. We evaluated KNN-GE in our study. The optimal k was selected based on the performance on GEO-va and 1000G-va.

3 Results

We have introduced two types of gene expression data, namely the GEO microarray data and the GTEx/1000G RNA-Seq data. We have formulated the gene expression inference as a multi-task regression problem, using the GEO data for training and both the GEO and the GTEx data for testing. We have also described our deep learning method D-GEX, and another two methods, LR and KNN regression, to solve the problem. Next, we show the predictive performances of the three methods on both the GEO data and the GTEx data.

3.1 Performance on the GEO data

D-GEX achieves the best performance on both GEO-va and GEO-te with 10% dropout rate (denoted as D-GEX-10%). Figure 1 and Table 1 show the overall performances of D-GEX-10% and the other methods on GEO-te. The complete performances of D-GEX with other dropout rates on both GEO-va and GEO-te are given in Supplementary Tables S2 and S3. The largest architecture of D-GEX-10% (three hidden layers with 9000 hidden units in each hidden layer, denoted as D-GEX-10%-9000 × 3) achieves the best performance on both GEO-va and GEO-te. The relative improvements of D-GEX-10%-9000 × 3 are 15.33% over LR and 45.38% over KNN-GE. Besides D-GEX-10%-9000 × 3, D-GEX-10% consistently outperforms LR and KNN-GE on all the other architecture as shown in Figure 1. One possible explanation is that deep architectures enjoy much richer representability than shallow architectures, thus learning complex features is much easier from the perspective of optimization (Bengio, 2009).

D-GEX also outperforms LR and KNN-GE for almost all of the target genes. Figure 2 shows the density plots of the predictive errors
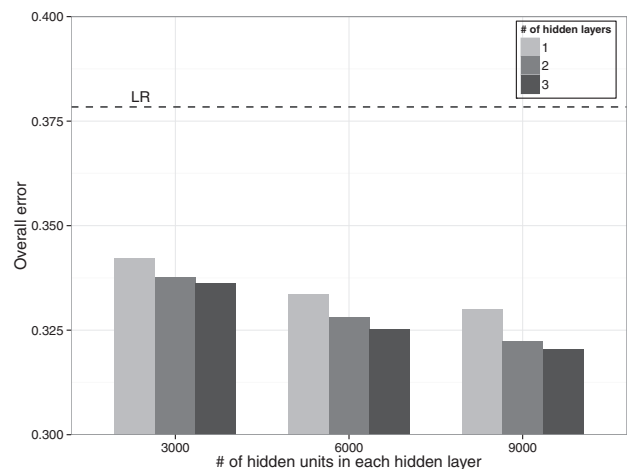

Fig. 1. The overall errors of D-GEX-10% with different architectures on GEO-te. The performance of LR is also included for comparison

Table 1. The overall errors of LR, LR-L1, LR-L2, KNN-GE and D-GEX-10% with different architectures on GEO-te

	Number of hidden units		
	3000	6000	9000
Number of hidden layers			
1	0.3421 ± 0.0858	0.3337 ± 0.0869	0.3300 ± 0.0874
2	0.3377 ± 0.0854	0.3280 ± 0.0869	0.3224 ± 0.0879
3	0.3362 ± 0.0850	0.3252 ± 0.0868	$\underline{\mathbf{0.3204} \pm \mathbf{0.0879}}$
LR		0.3784 ± 0.0851	
LR-L1		0.3782 ± 0.0844	
LR-L2		0.3784 ± 0.0851	
KNN-GE		0.5866 ± 0.0698	

Numerics after '±' are the standard deviations of prediction errors over all target genes. The best performance of D-GEX-10% is shown in bold font. The performance selected using model selection by GEO-va of D-GEX-10% is underscored.
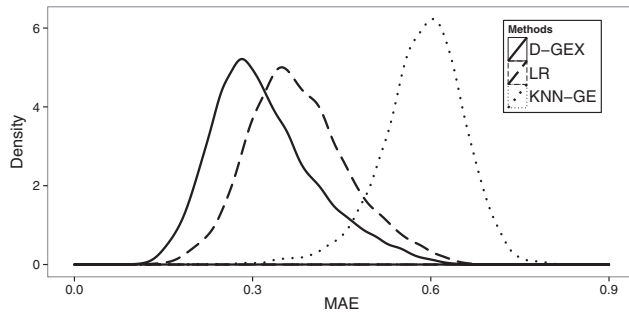
Fig. 2. The density plots of the predictive errors of all the target genes by LR, KNN-GE and GEX-10%-9000 × 3 on GEO-te

of all the target genes by LR, KNN-GE and GEX-10%-9000 × 3. Figure 3 shows a gene-wise comparative analysis between D-GEX-10%-9000 × 3 and the other two methods. D-GEX-10%-9000 × 3 outperforms LR in 99.97% of the target genes and outperforms KNN-GE in all the target genes. These results seem to suggest that D-GEX captured some intrinsic nonlinear features within the GEO data where LR and KNN-GE did not.

Regularization methods do not improve LR significantly. Table 1 shows the relative improvements of LR-L1 and LR-L2 over LR are 0.05 and 0.00%. Thus, it is most likely that LR is underfitting which means linear model is not complex enough to represent the data. Therefore, regularization techniques that reduce model complexity are not helpful.

KNN-GE performs significantly worse than the other methods. One possible explanation is that the *k*-nearest landmark genes for each target gene based on GEO-tr and GEO-te may not be fully consistent.

3.2 Performance on the GTEx data
Results on the GEO data demonstrate the significant improvement of D-GEX over LR and KNN-GE on the microarray platform. Yet in practice, the LINCS program trains regression models with the GEO data and performs gene expression inference on the L1000 data, which was generated with a different platform. Whether the significance of D-GEX preserves cross platforms requires further investigation. To explore the cross platforms scenario, we trained D-GEX with GEO-tr and evaluated its performances on GTEx-te

which was generated with the RNA-Seq platform (Lonsdale *et al.*, 2013).

However, new challenges arise in this scenario as the intrinsic distributions of the training data and the testing data may be similar but not exactly equivalent. Particularly in gene expression profiling, discrepancies between microarray and RNA-Seq data have been systematically studied (Wang *et al.*, 2014). Such discrepancies bring specific challenges to deep learning as the complex features it learn in the training data may not generalize well to the testing data, which leads to overfitting and reduces the prediction power. Therefore, more aggressive regularization may be necessary for deep learning to retain the maximum commonality cross platforms while avoiding platform-dependent discrepancies.

D-GEX-25%-9000 × 2 (with 25% dropout rate, two hidden layers with 9000 hidden units in each layer) achieves the best performance on both 1000G-va and GTEx-te. The relative improvements of D-GEX-25%-9000 × 2 are 6.57% over LR and 32.62% over KNN-GE. Table 2 shows the overall performances of D-GEX-25% and the other methods on GTEx-te. The complete performances of D-GEX with other dropout rates on both 1000G-va and GTEx-te are given in Supplementary Tables S4 and S5.

D-GEX still outperforms LR and KNN-GE in most of the target genes. Figure 4 also shows the gene-wise comparative analysis between D-GEX-25%-9000 × 2 and the other two methods. D-GEX-25%-9000 × 2 outperforms LR in 81.31% of the target genes and outperforms KNN-GE in 95.54% of the target genes. Therefore, the significance of D-GEX on the microarray platform basically preserves on the RNA-Seq platform. However, unlike the results on the GEO data, there is a noticeable number of target genes that D-GEX gets higher error than the other methods on the GTEx data. Thus, the expression patterns of these target genes D-GEX learned on the GEO data may be platform dependent and do not generalize well to the GTEx data. It is noteworthy that although the general performance of KNN-GE is still poor on the GTEx data, its errors on some of the target genes are significantly lower than D-GEX (dots in bottom right part of Fig. 4b). This is likely due to the gene-based aspect of KNN-GE that the numerical values predicted on target genes were not computed based on GEO-tr but based on GTEx-te itself. Therefore, the expression patterns captured by KNN-GE may be cross platforms invariant.

Dropout regularization effectively improves the performance of D-GEX on the GTEx data as shown in Figure 5. Without dropout, the overall error of D-GEX-9000 × 2 on GTEx-te slightly decreases at the beginning of training and then quickly increases, clearly implying overfitting. However, with 25% dropout rate, D-GEX-9000 × 2 achieves the best performance on both 1000G-va and GTEx-te.

3.3 Interpreting the learned neural network
We have demonstrated the performance of our deep learning method D-GEX on both the GEO microarray data and the GTEx RNA-Seq data. D-GEX outperforms LR on both types of expression data. On the other hand, interpreting the learned linear model from LR is straightforward as coefficients with large absolute value indicate strong dependencies between landmark genes and target genes. But for deep learning, currently there are no established methods to interpret the neutral networks learned from gene expression data. Next, we attempt to explore the learned neural networks with two strategies: (a) visualizing the major weights of the learned neural networks and (b) examining the nonlinearity captured by the hidden layers.
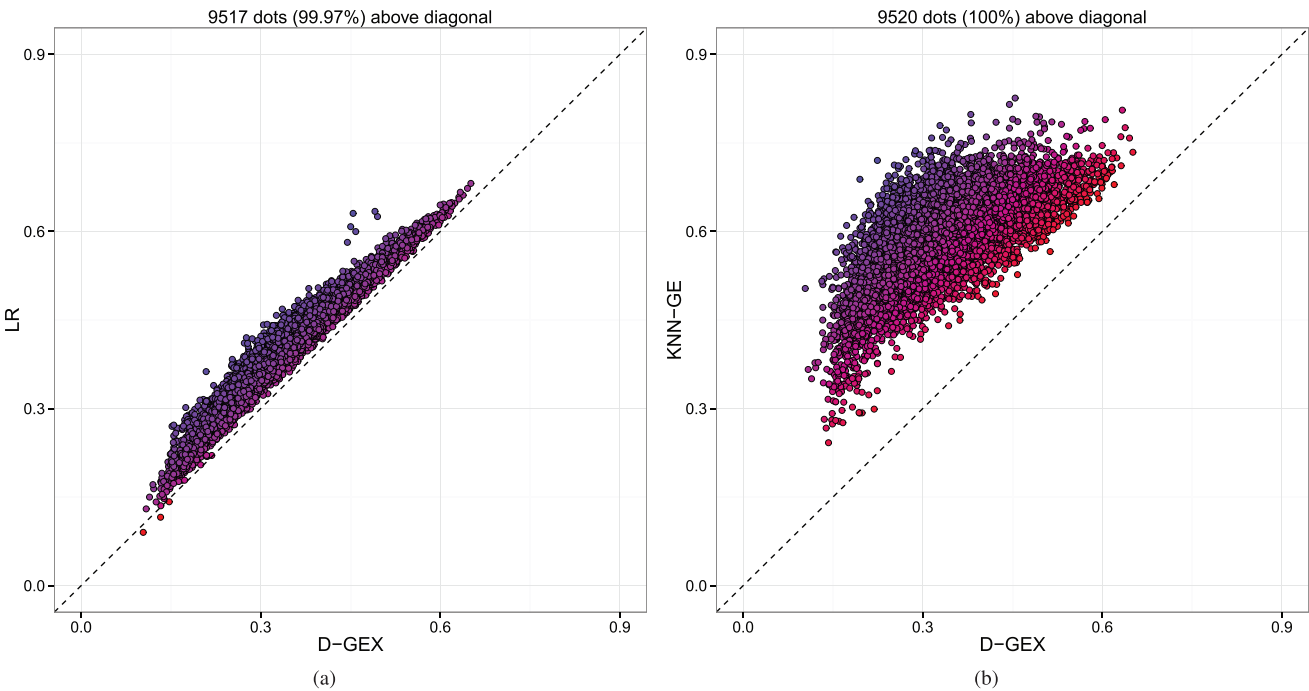
Fig. 3. The predictive errors of each target gene by GEX-10%-9000 $\times$ 3 compared with LR and KNN-GE on GEO-te. Each dot represents one out of the 9520 target genes. The *x*-axis is the MAE of each target gene by D-GEX, and the *y*-axis is the MAE of each target gene by the other method. Dots above diagonal means D-GEX achieves lower error compared with the other method. (**a**) D-GEX verse LR; (**b**) D-GEX verse KNN-GE

Table 2. The overall errors of LR, LR-L1, LR-L2, KNN-GE and D-GEX-25% with different architectures on GTEx-te

	Number of hidden units		
	3000	6000	9000
Number of hidden layers			
1	0.4507 ± 0.1231	0.4428 ± 0.1246	0.4394 ± 0.1253
2	0.4586 ± 0.1194	0.4446 ± 0.1226	<u>0.4393 ± 0.1239</u>
3	0.5160 ± 0.1157	0.4595 ± 0.1186	0.4492 ± 0.1211
LR		0.4702 ± 0.1234	
LR-L1		0.5667 ± 0.1271	
LR-L2		0.4702 ± 0.1234	
KNN-GE		0.6520 ± 0.0982	

Numerics after '$\pm$' are the standard deviations of prediction errors over all target genes. The best performance of D-GEX-25% is shown in bold font. The performance selected using model selection by 1000G-va of D-GEX-25% is underlined.

(a) *Visualizing the major weights* is a strategy inspired by the method of interpreting linear model that coefficients with large absolute value indicate strong dependencies between inputs and targets. Similarly, we examined the weights of the learned neural network of D-GEX-10%-3000 $\times$ 1 that was trained based on half of the target genes of GEO-tr and GEO-va. The weights from input to hidden units were randomly initialized with dense connections. However, after learning, the connections became so sparse that each input unit was primarily connected to only a few hidden units with the weights to the rest of hidden units decayed to near 0. Similar patterns were also observed for connections from the hidden to the output layer. Therefore, we created a visualization map of the learned connections by removing those with weights near zeros. Specifically, for each input unit (landmark gene), we calculated the mean and the standard deviation of the weights of the connections between the input unit and the

3000 hidden units. Then, we only retained the major weights that were four standard deviations away from the mean. Similarly, we used a threshold of five standard deviations to retain the major weights of the connections between the output units (target genes) and the hidden units. We colored the weights differently so that red indicates positive weights and blue indicates negative weights. Supplementary Figure S3 shows the final visualization map. From the visualization map, we noticed two interesting observations: (a) Most of the units in the input layer and the output layer have connections to the hidden layer. In contrast, only a sparse number of units in the hidden layer have connections to the input and the output layer. Specially, the connections to the output layer are dominated by a few hidden units, which we refer to as the 'hub units'. (b) Lots of the 'hub units' seem to have only one type of connections to the output layer, e.g. some of them only have positive connections (red edges), while some other units only have negative connections (blue edges). It seems that these 'hub units' may have captured some strong local correlations between the landmark genes and target genes.

(b) *Examining the nonlinearity* is a strategy to show that the intermediate hidden layers have captured some nonlinearity within the raw expression data. The neural networks we used are quite complex, containing several layers and many hidden units, each of which is activated through a nonlinear transfer function. To dissect the nonlinear contribution, we took a relatively simple approach by focusing on the representation (activations) from the last hidden layer. Each of the hidden unit in that layer can be viewed as a feature generated through some nonlinear transformation of the landmark genes. We then studied whether an LR based on these nonlinear features can achieve better performance than an LR based solely on the landmark genes. For this purpose, we measured the linear correlation between the activations from the last hidden layer of D-GEX-10%-9000 $\times$ 3 and the final targets (the expression of target genes), and compared it with the linear correlation between the raw inputs
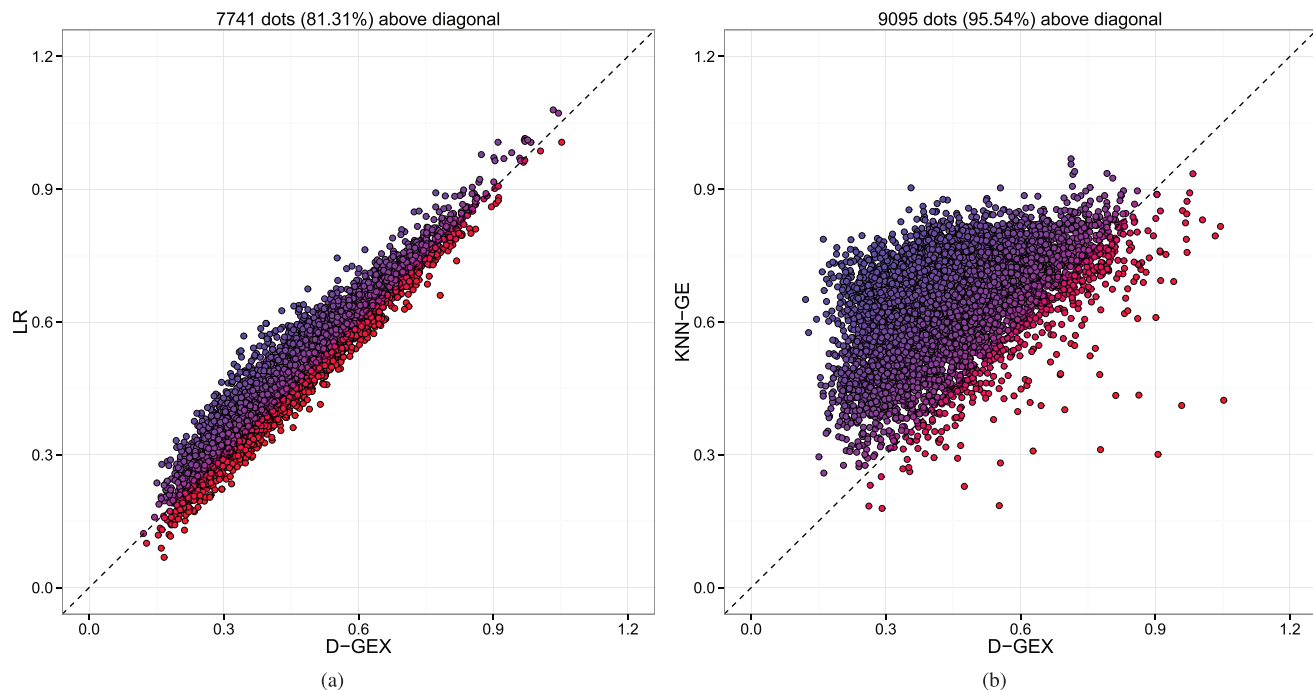
Fig. 4. The predictive errors of each target gene by GEX-25%-9000 × 2 compared with LR and KNN-GE on GTEx-te. Each dot represents one out of the 9520 target genes. The x-axis is the MAE of each target gene by D-GEX, and the y-axis is the MAE of each target gene by the other method. Dots above diagonal means D-GEX achieves lower error compared with the other method. (a) D-GEX versus LR; (b) D-GEX versus KNN-GE
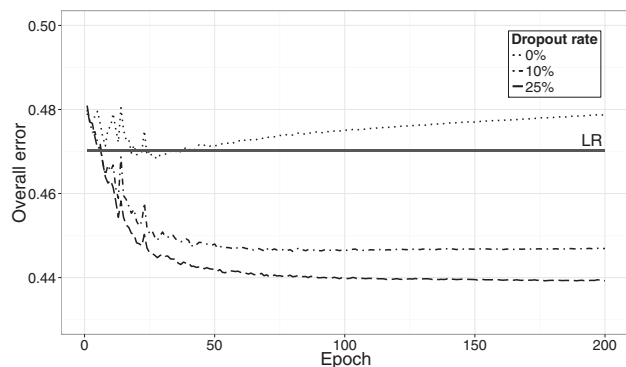

Fig. 5. The overall error decreasing curves of D-GEX-9000 × 2 on GTEx-te with different dropout rates. The x-axis is the training epoch and the y-axis is the overall error. The overall error of LR is also included for comparison

and the final targets. Normally, coefficient of determination (R^2) is used to compare the fitnesses of different linear models. Since the dimensionality has changed from the raw inputs to the transformed activations, we used adjusted R^2 (Theil, 1958) to specifically account for the change in dimensionality. We calculated the adjusted R^2 of both the raw inputs and the transformed activations for each target gene based on GEO-tr. Supplementary Figure S2 shows the gene-wise comparison of adjusted R^2 between the raw inputs and the transformed activations. The transformed activations have a larger adjusted R^2 than the raw inputs in 99.99% of the target genes. It seems to indicate that the intermediate hidden layers have systematically captured some nonlinearity within the raw expression data that would be ignored by simple LR. After the nonlinear transformation through the hidden layers, the activations fit the final targets significantly better than the raw inputs using a simple linear model. The analysis seems to suggest that most of the target genes

benefit from the additional nonlinear features, although to a different extent as characterized by the adjusted R^2.

3.4 Inference on the L1000 data

The LINCS program has used the L1000 technology to measure the expression profiles of the 978 landmark genes under a variety of experimental conditions. It currently adopts LR to infer the expression values of the 21 290 target genes based on the GEO data. We have demonstrated our deep learning method D-GEX achieved significantly improvement on prediction accuracy over LR on the GEO data. Therefore, we have re-trained GEX-10%-9000 × 3 using all the 978 landmark genes and the 21 290 target genes from the GEO data and inferred the expression values of unmeasured target genes from the L1000 data. The full dataset consists of 1 328 098 expression profiles and can be downloaded at https://cbcl.ics.uci.edu/pub lic_data/D-GEX/l1000_n1328098x22268.gctx. We hope this dataset will be of great interest to researchers who are currently querying the LINCS L1000 data.

4 Discussion

Revealing the complex patterns of gene expression under numerous biological states requires both cost-effective profiling tools and powerful inference frameworks. While the L1000 platform adopted by the LINCS program can efficiently profile the ∼1000 landmark genes, the linear-regression-based inference does not fully leverage the nonlinear features within gene expression profiles to infer the ∼21 000 target genes. We presented a deep learning method for gene expression inference that significantly outperforms LR on the GEO microarray data. With dropout as regularization, our deep learning method also preserves cross platforms generalizability on the GTEx RNA-Seq data. In summary, deep learning provides a

better model than LR for gene expression inference. We believe that it achieves more accurate predictions for target gene expressions of the LINCS dataset generated from the L1000 platform.

Interpreting the internal representation of deep architectures is notoriously difficult. Unlike other machine learning tasks such as computer vision, where we can visualize the learned weights of hidden units as meaningful image patches, interpreting the deep architectures learned by biological data requires novel thinking. We attempted to interpret the internal structures of the neural networks learned from gene expression data using strategies that were inspired by linear model. Yet, more systematic studies may require advanced computational frameworks that are specifically designed for deep learning. Unsupervised feature learning methods, such as autoencoder (Vincent *et al.*, 2008) and restricted Boltzmann machine (Hinton, 2010) may provide some insights on this problem.

In the current setting, target genes were randomly partitioned into multiple sets, and each set was trained separately using different GPUs due to hardware limitations. Alternatively, we could first cluster target genes based on their expression profiles, and then partition them accordingly rather than randomly. The rationale is that target genes sharing similar expression profiles share weights in the context of multi-task neural networks. Ultimately, the solution is to jointly train all target genes, either by using GPUs with larger memory such as the more recent Nvidia Tesla K80, or by exploiting multi-GPU techniques (Coates *et al.*, 2013).

References

Ardlie,K.G. *et al.* (2015) The genotype-tissue expression (gtex) pilot analysis: multitissue gene regulation in humans. *Science*, 348, 648–660.

Baldi,P. and Sadowski,P.J. (2013) Understanding dropout. In *Advances in Neural Information Processing Systems*, pp. 2814–2822.

Baldi,P. *et al.* (2014) Searching for exotic particles in high-energy physics with deep learning. *Nat. Commun.*, 5, 4308.

Bansal,M. *et al.* (2007) How to infer gene networks from expression profiles. *Mol. Syst. Biol.*, 3, 1.

Bengio,Y. (2009) Learning deep architectures for AI. *Found. Trends Mach. Learn.*, 2, 1–127.

Bengio,Y. *et al.* (2013) Representation learning: a review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.* 35, 1798–1828.

Bentley,J.L. (1975) Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18, 509–517.

Bergstra,J. *et al.* (2010) Theano: a CPU and GPU math expression compiler. In: *Proceedings of the Python for Scientific Computing Conference (SciPy)*, Austin, US.

Chen,Y. (2014) Machine learning for large-scale genomics: algorithms, models and applications. PhD Thesis, University of California, Irvine, ProQuest, UMI Dissertations Publishing.

Ciresan,D. *et al.* (2012) Multi-column deep neural networks for image classification. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Providence, US, pp. 3642–3649.

Coates,A. *et al.* (2013) Deep learning with cots hpc systems. In: *Proceedings of the 30th International Conference on Machine Learning*, Atlanta, US, pp. 1337–1345.

Di Lena,P. *et al.* (2012) Deep architectures for protein contact map prediction. *Bioinformatics*, 28, 2449–2457.

Edgar,R. *et al.* (2002) Gene expression omnibus: NCBI gene expression and hybridization array data repository. *Nucleic Acids Res.*, 30, 207–210.

Glorot,X. and Bengio,Y. (2010) Understanding the difficulty of training deep feedforward neural networks. In: *International Conference on Artificial Intelligence and Statistics*, Sardinia, Italy, pp. 249–256.

Goodfellow,I.J. *et al.* (2013) Pylearn2: a machine learning research library. *arXiv Preprint arXiv:1308.4214*.

Hasty,J. *et al.* (2001) Computational studies of gene regulatory networks: in numero molecular biology. *Nat. Rev. Genet.*, 2, 268–279.

Hinton,G. (2010) A practical guide to training restricted Boltzmann machines. *Momentum*, 9, 926.

Hinton,G. *et al.* (2012a) Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. *IEEE Signal Process. Mag.*, 29, 82–97.

Hinton,G.E. *et al.* (2012b) Improving neural networks by preventing co-adaptation of feature detectors. *arXiv Preprint arXiv:1207.0580*.

Krizhevsky,A. *et al.* (2012) Imagenet classification with deep convolutional neural networks. In: *Advances in Neural Information Processing Systems*, Lake Tahoe, US, pp. 1097–1105.

Lamb,J. *et al.* (2006) The connectivity map: using gene-expression signatures to connect small molecules, genes, and disease. *Science*, 313, 1929–1935.

Lappalainen,T. *et al.* (2013) Transcriptome and genome sequencing uncovers functional variation in humans. *Nature*, 501, 506–511.

Leung,M.K. *et al.* (2014) Deep learning of the tissue-regulated splicing code. *Bioinformatics*, 30, i121–i129.

Lonsdale,J. *et al.* (2013) The genotype-tissue expression (GTEX) project. *Nat. Genet.*, 45, 580–585.

Peck,D. *et al.* (2006) A method for high-throughput gene expression signature analysis. *Genome Biol.*, 7, R61.

Pedregosa,F. *et al.* (2011) Scikit-learn: machine learning in Python. *J. Mach. Learn. Res.*, 12, 2825–2830.

Quang,D. *et al.* (2014) DANN: a deep learning approach for annotating the pathogenicity of genetic variants. *Bioinformatics*,

Rumelhart,D.E. *et al.* (1988) Learning representations by back-propagating errors. *Cogn. Model.*, 5, 1.

Socher,R. *et al.* (2011) Parsing natural scenes and natural language with recursive neural networks. In: *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 129–136.

Srivastava,N. *et al.* (2014) Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15, 1929–1958.

Sutskever,I. *et al.* (2013) On the importance of initialization and momentum in deep learning. In: *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, Atlanta, US, pp. 1139–1147.

Theil,H. (1958) *Economic Forecasts and Policy*. North-Holland, Amsterdam.

Vincent,P. *et al.* (2008) Extracting and composing robust features with denoising autoencoders. In: *Proceedings of the 25th International Conference on Machine Learning*, ACM, Helsinki, Finland, pp. 1096–1103. ACM.

Wang,C. *et al.* (2014) The concordance between RNA-Seq and microarray data depends on chemical treatment and transcript abundance. *Nat. Biotechnol.*, 32, 926–932.

Ye,G. *et al.* (2013) Low-rank regularization for learning gene expression programs. *PloS One*, 8, e82146.