Gene Expression Prediction Using Deep Neural Networks

Raju Bhukya and Achyuth Ashok
Department of Computer Science and Engineering, National Institute of Technology, India

Abstract: *In the field of molecular biology, gene expression is a term that encompasses all the information contained in an organism's genome. Although, researchers have developed several clinical techniques to quantitatively measure the expressions of genes of an organism, they are too costly to be extensively used. The NIH LINCS program revealed that human gene expressions are highly correlated. Further research at the University of California, Irvine (UCI) led to the development of D-GEX, a Multi Layer Perceptron (MLP) model that was trained to predict unknown target expressions from previously identified landmark expressions. But, bowing to hardware limitations, they had split the target genes into different sets and constructed separate models to profile the whole genome. This paper proposes an alternative solution using a combination of deep autoencoder and MLP to overcome this bottleneck and improve the prediction performance. The microarray based Gene Expression Omnibus (GEO) dataset was employed to train the neural networks. Experimental result shows that this new model, abbreviated as E-GEX, outperforms D-GEX by 16.64% in terms of overall prediction accuracy on GEO dataset. The models were further tested on an RNA-Seq based 1000G dataset and E-GEX was found to be 49.23% more accurate than D-GEX.*

Keywords: *Gene expression, regression, deep learning, autoencoder, multilayer perceptron.*

Received April 25, 2018; accepted October 28, 2018
https://doi.org/10.34028/iajit/17/3/16

1. Introduction

An all inclusive picture of cellular function can be obtained by measuring the activity (also known as expression) of thousands of genes collectively. In molecular biology, this significant domain of work is termed as gene expression profiling [4]. Gene expression can alternatively be viewed as the direct mapping from the genotype to the phenotype, which is the observable trait. Functional gene products, such as proteins and functional RNAs are synthesized using the information obtained from the gene expression. There exist several clinical techniques to measure the expression of a gene. For instance, DNA microarrays measure the relative activity of previously identified target genes [8]. Sequence based techniques like RNA-Seq provide information on the sequences of genes in addition to their expression level. Genes are expressed by being transcribed into RNA, which in turn may be translated into protein. The genetic code of each organism stored in its DNA is quantitatively described using gene expression. The properties of the expression govern the traits that are showcased by the organism. Its shape, size, structure and behaviour are controlled by the proteins that are synthesized by its DNA. Regulation of gene expression is thus critical to an organism's development. Its cellular response under a wide variety of conditions such as diseases, genetic mutations, intake of medicines and drugs etc., can thus be studied through the complete profiling of gene expressions [15].

2. Related Work

There are many techniques that are employed to measure the gene expression levels of an organism, and they can be broadly classified under two categories, clinical and statistical. The Connectivity Map (CMap) project is one that comes under the former [20]. Even with the advent of latest technological innovations, whole genome expression profiling is too expensive to be used in an academic set up. The CMap project in the initial phase was able to perform just 564 genome-wide expression profiles using the Affymetrix GeneChip microarrays. On the contrary, statistical methods rely on inferring the expression of several genes using the clinically measured expressions of a select number of genes. It was observed that although the human genome consisted of approximately 22000 genes, the expression levels of most of them were highly correlated. Researchers from the Library of Integrated Network-based Cellular Signatures (LINCS) program employed this observation to formulate a cost effective alternative to whole genome expression profiling [25]. Researchers from the NIH LINCS program employed Principal Component Analysis (PCA) and discovered that about 1000 genes explained close to 80% of the variance in the genome-wide expression. They decided to call these 978 genes as landmark genes, since they could be used to predict the expression levels of the rest [8]. Using the L1000 Luminex bead technology, the expressions of these landmark genes were measured and normalized [25].

This landmark expression signature was considered to accurately represent the cellular state at any given time. To obtain expression values for all the genes, it was assumed that the expression of an unmeasured gene can be predicted from the measured landmark genes using Linear Regression (LR). These clinically unmeasured but statistically inferred genes were termed as target genes [8].

However, the relative number of the target genes (about 21000) with respect to the landmark genes (978) vexed the researchers. Assuming linear relationship between the target and landmark gene expressions, they employed multi-task LR to predict the levels of the target genes. They had to build close to 21000 models, each one fitted for a specific target gene, in order to profile the genome wide gene expression and compare with the expression profiles from the L1000 project. Although the linear model was highly scalable, it failed to capture the non-linear relationships between some genes.

Most of the existing machine learning methods encounter issues like improper separation of feature extraction and model training, manually defined features. Most of the applied features requires domain expert in order to reduce the complexity of the data. Sometimes these features are extremely laborious and expensive Deep learning, as the modern, leading-edge machine learning method has received ardent audience in the field of artificial intelligence. It has the ability to automatically extracts features from sequences and discovers complex representations of data patterns

Poor scalability of non-linear techniques like kernel machines steered the researchers away from such alternatives [8]. Researchers at the University of California, Irvine (UCI) decided that a machine learning approach was the way forward [8]. They argued that it would enhance the scalability and usher in better data representation. The expansive field of deep learning that could account for complex hierarchical nonlinear relationships was identified as an ideal solution. Deep learning, also called hierarchical learning uses multiple layers of abstraction to learn complex data representations [21]. The study at UCI culminated in the formulation of D-GEX, which is a multi-task multi-layer feed-forward neural network. The neural network architecture is akin to that of a multi-layer perceptron (MLP). The best performing model with 5 layers (1 input layer, 3 hidden layers of 9000 nodes each, 1 output layer) was reported to be 15% more accurate than the LR model [8].

D-GEX contains 943 units in the input layer corresponding to the 943 landmark genes. D-GEX also configured with 9520 units in the output layer corresponding to the 9520 target genes. Due to memory constrains D-GEX hidden layers are not configured. Therefore, random partition was done with 9520 target genes into two sets that each contains 4760 target genes. Two separate neural networks with each output layer corresponding to one half of the target genes was built and constructed series of different architecture with varying depths.

This paper identifies certain shortcomings of the D-GEX project, some of which are general to any deep neural network and others that are specific to the human genome dataset based on Microarrays [1]. A prime culprit belonging to the former category is the vanishing gradient problem that plagues any deep neural network [14, 15]. Theoretically, deeper and deeper networks are able to capture complex non-linear relationships within the data. But as the depth of the network grows, it becomes increasingly difficult to train the network. This is chiefly due to the fact that the errors propagated by the back propagation algorithm keeps on diminishing as it traverses towards the first layer. Thus there is little or no change in weights of the first few layers. As a result, these layers are poorly trained. Hence, although deeper networks may give better results, training them is cumbersome. Thus, the number of hidden layers that can be employed is severely restricted. The Gene Expression Omnibus (GEO) dataset that is used for training consist of 112634 samples of 978x21290 genes (978 landmarks versus 21290 targets) [12]. Training networks using such a large dataset is computationally expensive. Large number of features results in extremely slow training [5]. The problem is compounded by the fact that large feature size makes it much harder to find a good solution too. This problem is known as the curse of dimensionality. High dimensional datasets are prone to become too sparse, i.e., two instances randomly chosen are highly likely to be tremendously far away from each other [14]. Previously unseen instances may diverge from the instances that the network was trained on and thus predictions will be inaccurate. Having greater dimensions also bring forth the risk of over fitting the data [5]. Ideally, the number of training instances for training such high dimensional datasets should be sufficiently large, but increasing the sample size is not always possible [14]. The hidden units are activated using the hyperbolic tangent function in the D-GEX model [8]. Although hyperbolic tangent is better than logistic function because its mean is at 0, the function saturates to +1 or -1 as the input increases or decreases, respectively. Thus the derivatives are pulled to zero and thus the vanishing gradient problem kicks in [14]. As the number of hidden layers increase, the fitted model may be far from the ideal model. Another key drawback of D-GEX was that training was done separately for randomly segregated batches of target genes due to hardware limitations [8]. This in turn increases the overall training time. Also, since the batches were randomly chosen, the correlation information between the target genes was ignored.

Here, we present Encoded Gene Expression Predictor (abbreviated as E-GEX), which is a combination of an autoencoder and a multi-layer feed-forward neural network. E-GEX attempts to systematically alleviate the shortcomings of D-GEX by building an autoencoder to derive a concise encoded representation of target expression signature and then developing an MLP to predict the encoded target expression from the original landmark signature [8, 23]. We introduced some modifications to the MLP network of D-GEX by altering the optimization strategy and activation function. The performance of E-GEX on the Microarray based GEO data and RNA-Seq based 1000G data was determined to be better than that of D-GEX, when both were trained on the cross-platform quantile normalized expression profiles. Additionally, E-GEX was trained independently on the GEO data to efficiently predict entire target profiles (21290 target genes) using a single network, something that D-GEX failed to perform.

3. Proposed Model

The key idea of gene expression prediction is to treat it as a supervised learning problem [8]. Since the target values are expressions of target genes, the learning scenario can be simplified to multi-task regression. Expression profiles that were generated by the LINCS program are used to train and test models built for the expression inference.

3.1. Dataset

GEO is a publicly available database repository of high throughput gene expression data and hybridization arrays, chips, microarrays [12]. The GEO expression data was made publicly available by the LINCS program. It was curated by the Broad Institute from the publicly available GEO database. It consists of 112634 gene expression profiles from the Affymetrix microarray platform. The 978 landmark genes and 21290 target genes of the human genome are represented by 22268 probes per profile. The expression levels are normalized and fall in the range 4-15 [8].

A supplementary Illumina RNA-Seq platform dataset, titled as 1000 Genomes (1000G) RNA-Seq expression data is also used to train E-GEX. This dataset is not derived from the Affymetrix Microarray platform and thus is used for testing the cross platform viability of the models developed. The expression levels of 462 profiles of lymphoblastoid cell line samples measured in RPKM format based on Gencode V12 annotations constitute 1000G [8].

The two datasets that are used for training E-GEX are derived from different platforms. Also, one Gencode probe may include multiple probes of the Microarray platform [8]. Using the gene ID and name information from the two datasets, probes that were common to both platforms were identified and the rest were pruned off. After pruning, we were left with 943 landmark genes and 9520 target genes, which exactly matched the feature set that was used to train D-GEX. Since the range of values in GEO Expression dataset and 1000G differ, they were quantile normalized with standard normal distribution as reference [8]. The normalized version of the GEO Expression data, which is called GEO-norm, is used for training the models. The normalized 1000G dataset is set aside for testing and inferring the cross platform viability of E-GEX. The original GEO dataset is used to build a complete model that can predict the entire target signature from the landmark expression.

3.2. Encoding Target Expressions

Principal Component Analysis (PCA) is applied on the target gene set to identify whether the target genes are further correlated [8]. PCA projected the 21290 target genes into a 508 features, retaining about 99% of the variance in the target data, without significant reconstruction error (approximately 0.22).

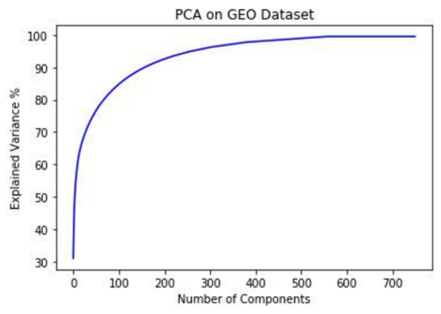


Figure 1. Principal components analysis on target expressions.

Hence, it is inferred that the target expression can ideally be represented using much less number of features than 21290, without compromising much on the accuracy. Although PCA is used to validate our attempt to encode the target signature, the projections made by PCA are based on an assumption of linearity [26]. Instead, autoencoders are used to find complex nonlinear encodings of the target expressions [6, 22].

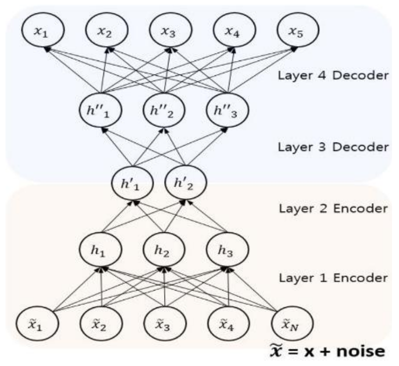


Figure 2. Deep Autoencoder with 3 hidden layers.

Autoencoders are artificial neural networks capable of learning efficient coding of the input data without any supervision [2]. Basically, an autoencoder learns to copy their inputs to their outputs. The coding can be viewed as the manifestation of the autoencoders attempt to learn the identity function under constraints like limiting the number of internal nodes [14]. An autoencoder typically consists of two phases - an encoder phase and a decoder phase. The number of neurons in the output layer must be equal to that of the number of inputs. By stacking autoencoders one above another to form a deep autoencoder, it can learn much more complex coding [14, 31]. A deep autoencoder is symmetrical about its central hidden layer. Training can be made easier by tying the weights, a technique in which the weights of the encoder phase are reused for the decoder phase after being transposed [14, 32].

Autoencoders learn by reducing the error in reconstructing the input data [2]. For E-GEX, two autoencoders are separately trained - one to encode the whole target set (21290 genes) and another to encode the target genes in GEO-norm. The target expressions are encoded into a 2000 feature representation. The chosen feature length is approximately 4 times the number of features suggested by PCA to account for nonlinearity. For ease of training, deep architectures with 3 hidden layers are selected. Different number of nodes are tried for the first and third layer (5000, 6000, 8000), with the innermost encoding layer consisting of 2000 neurons [8]. This is the first phase of E-GEX.

3.3. Predicting Target Expressions

The principal objective of our work is to find a single deep neural network model that can predict target expressions, given the corresponding landmark expressions. A Multilayer Perceptron (MLP), which is a multilayer feed-forward neural network is used for building this prediction model. Except the input layer nodes, each node in an MLP is a neuron with a nonlinear activation function. Formally, an MLP can be considered as a non-linear transformation $f : R^M \rightarrow R^N$, where M is the size of the input vector and N is the size of the output vector [8]. The inputs to the hidden layer are calculated as the weighted sum of the inputs. The input to the output layer is the weighted sum of the hidden layer activations. The number of hidden layers can be increased, forming a deep network that introduces greater levels of abstraction.

The second phase of E-GEX is an MLP with 943 nodes in the input layer and 2000 nodes in the output layer. The number of hidden layers and number of nodes per hidden layer are varied to find a best fit model. The following subsections describe the various parameters and constraints that were chosen while constructing the E-GEX MLP.

1) *Activation*: The hyperbolic tangent function used in D-GEX can capture the non-linearity, but it is prone to the vanishing gradients problem since it saturates to 1 for high/low input values [8]. The Restricted Linear Units (ReLU) were introduced as a compromise between fully linear units and nonlinear units [16]. But the output of ReLU is pulled to 0 for negative values. The Exponential Linear Unit (ELU) was derived as an alternative to ReLU in neural networks [10]. The exponential linear function is defined as

$$f(x)=\begin{cases} x & if\ x>0. \\ \alpha(e^x-1) & if\ x \le 0. \end{cases} \quad (1)$$

ELUs are not vulnerable to the vanishing gradient problem since they are continuous everywhere, even at the origin, as depicted in Figure 3. The calculation of the derivative is a bit more involved for ELUs than for hyperbolic tangents or ReLU, but they converges faster [14].

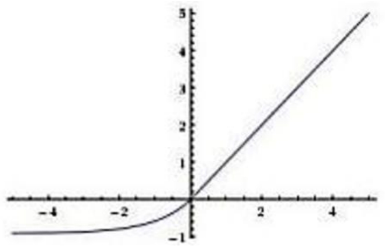


Figure 3. Exponential linear function.

In recent times, ELUs have become the norm in deep neural networks. All but the output layer in the E-GEX MLP are composed of ELUs. The output layer neurons are linear units since the output values are continuous [8].

2) *Initialization*: Randomized initialization of weights and biases is a huge deterrent in deep neural networks because it can direct the cost function to local minima [14]. Also, the training may go on indefinitely without finding an optimal solution. As an alternative, researchers suggested sampling from a normal distribution so that the total variance of the inputs of each layer and the total variance of the outputs are similar [11]. Also the gradients would have the same variance before and after flowing through a layer when such an initialization strategy is used. The fan-in and fan-out of each layer is used to select the mean and standard deviation of the Normal distribution from which sampling is done [14]. The Glorot Initialization scheme samples from a normal distribution with mean 0 and standard deviation,

$$\sigma = \sqrt{\frac{2}{n_{input} + n_{output}}} \quad (2)$$

or an uniform distribution between -r and +r where

$$r = \sqrt{\frac{6}{n_{input} + n_{output}}} \quad (3)$$

This generic scheme was modified through derivation of formulas suited specifically to specific activation functions [11]. The above mentioned r and were found suitable for logistic functions. For Rectified Linear Units (ReLU) and its variants like ELU the parameters r and are multiplied by a factor of √2. This variance scaling initializer scheme, also known as He initialization is used for the lower layers of the E-GEX MLP [14]. Since the MLP is used for regression, the initial values of output layer is sampled from a uniform distribution in the range [-1x10^{-4}, 1x10^{-4}] [8]. Selection of appropriate activation functions and initialization schemes can alleviate the vanishing gradient problem.

3) *Optimization*: Standard back-propagation is the most widely adopted technique for updating weights in neural networks [28]. Researchers have developed a variety of techniques to optimize back propagation and thus train neural networks faster. Gradient descent is the most common technique for optimization [14]. Momentum optimization method was derived as a way to accelerate the simple gradient descent optimization [14, 27]. Gradient descent simply updates the weights by subtracting the gradient of the cost function $J(\theta)$ multiplied by the learning rate [14].

$$\theta=\theta-\alpha\frac{\partial}{\partial\theta}(J(\theta)) \tag{4}$$

The key idea in Momentum optimization is to give sufficient weight to the previous gradients and not just the current derivative. Each epoch adds the local gradient to the momentum vector m and it updates the weights using this parameter [27].

$$m=\beta m+\alpha\frac{\partial}{\partial\theta}(J(\theta)), \tag{5}$$

Where β is momentum, and

$$\theta=\theta-m \tag{6}$$

Nesterov Accelerated Gradient (NAG) is a small variant of the momentum optimization, which has been empirically observed to be faster than the momentum technique [14, 24]. The chief idea is to measure the gradient of the cost function not at the local position, but slightly ahead in the direction of the momentum.

$$m=\beta m+\alpha\frac{\partial}{\partial\theta}(J(\theta+\beta m)) \tag{7}$$

$$\theta=\theta-m$$

In general the momentum vector will be pointing in the direction of the optimum and thus it will be slightly more accurate to use the gradient a bit farther in that direction than at the current position [24]. Although the modification is slight, it helps in rolling down the cost slope a bit faster [14]. Optimisation of all the feed-forward networks trained as part of this work is done using NAG.

4) *Regularization*: Neural networks have a tendency to overfit the data on which it was trained [15]. A collection of regularization techniques aims at creating the most generalized neural network models. Simple regularization techniques like L1 and L2 regularization imposes restrictions on the weights of the network by limiting their range. Dropout is the most popular regularization technique for deep neural networks [14, 30]. The approach is very simple - at every training step, every neuron (except the output neurons) is temporarily dropped out of the network with a probability p. The dropped neurons will be completely ignored during this training step, but may become active in the next one [3]. The hyperparameter p is termed as the dropout rate.

Dropout prevents neurons from co-adapting with its neighbours and thus they learn to become useful on their own. Also, they cannot rely on just a section of the input neurons and has to pay attention to all of them [14]. Thus, a more robust network that generalizes well will be obtained. Dropping is strictly restricted to the training phase. During the testing phase, each of the input connection weight is multiplied by a factor of (1-p) to adjust for the increase in the number of neurons [3]. Although dropout may delay convergence, it produces a good generalized model [14].

Early stopping is another form of regularization where training is pre-empted when the performance on a validation set (which the network does not see while training) keeps on reducing for a series of consecutive epochs [7]. The models are check pointed at regular intervals and when a better model is not obtained for k epochs, training is terminated and the previously check pointed model is restored for testing and further analysis [14]. Assigning small values for k can lead to extensive early stopping due to slight kinks in the cost function. Both early stopping and dropout regularization are used together while training the MLP networks.

5) *Hidden Layers*: Despite the widespread popularity of deep learning, there is still no universal consensus on the ideal number of hidden layers and the number of neurons per hidden layer to be used to construct the neural networks. Derivation of a global solution is cumbersome since these parameters are very much application dependent. Nevertheless, researchers have come up with certain workarounds where the ideal hidden layer configuration is deduced from other known parameters. For instance, irrespective of the nature of the data, the number of neurons per hidden layer can be derived using the cardinality of the dataset and the sample size. One popular rule of thumb is that a standard 2-layer feed-forward neural network with $\sqrt{(m+2)N}+2\sqrt{\frac{N}{m+2}}$ neurons in the first hidden layer and $m\sqrt{(\frac{N}{m+2})}$ neurons in the second hidden layer can represent N distinct input samples with any desired precision, where m is the number of output neurons [17]. For our work, such a 2 hidden layer model consisting of 11850 and 11826 nodes respectively is adopted (our chosen m=2000 and N≈70000, the number of training samples). An E-GEX model is also constructed bereft of the

autoencoder stage to compare the effectiveness of this configuration with the best performing D-GEX network consisting of 3 hidden layers of 9000 nodes each [8]. In this model, m = 4760 (one half of the target expressions) and N 70000, thus producing 2 hidden layers of approximately 18000 nodes each. Due to hardware limitations, ultimately only 12000 nodes are used in each of the hidden layers.

6) *Hyperparameters*: Parameters of the learning algorithms that are implicitly introduced into the model are collectively known as hyperparameters [14]. The step size in gradient descent is controlled by a hyperparameter called learning rate [29]. Learning rate of E-GEX MLP is initialized to 5x 10^{-4}. For optimal learning, this value is programmatically tuned using a decay rate of 0.9 until it reaches a minimum of $1x10^{-5}$ [8]. The tuning is done based on the prediction performance on a subset of the training set. For the acceleration using NAG, momentum hyperparameter is set as 0.9 [8, 14]. Dropout is performed with a dropout rate of 10%, which means that 90% of the internal nodes are retained during each epoch [8].

3.4. Model Selection

The target encoded GEO-norm dataset is partitioned into 3 disjoint sets - GEO-norm-tr, GEO-norm-val and GEO-norm-ts for training, validation and testing respectively. Instead of randomly segregating the sample profiles, we have adopted k-means clustering to ensure good representation and thus prevent overfitting [8, 13].

Appropriate value of k for clustering is normally identified as the elbow point in the plot of k versus squared error [13]. The plot in Figure 4 does not showcase a clear elbow and k was chosen as 20 because the error appears to be quite stagnant after that. From each cluster, a maximum of 4500 expression profiles are written onto GEO-norm-tr and 700 onto GEO-norm-val. The remnants of each cluster, if any, are added to GEO-norm-ts.

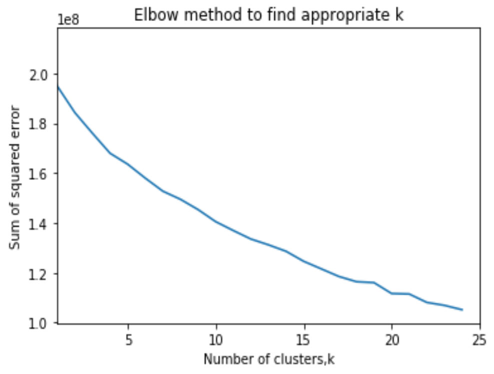


Figure 4. Plot of k vs sum of squared error.

The same approach is used to partition the original GEO dataset into GEO-tr, GEO-val and GEO-ts. In each case, the performance on the validation set is monitored every 5 epochs. Training will be stopped early if the latest model performs worse than the model 25 epochs prior to it. Thus the model with the least validation error will be chosen as the best performer. Mean Absolute Error (MAE) is the metric that is used to measure performance [8, 19]. For n samples,

$$MAE = \frac{\sum_{i=1}^{n} | y_i - \overline{y_i} |}{N}, \tag{8}$$

Where y_is are the actual values and $\overline{yi}$s are the predicted values [8, 19]. Different models are compared based on their MAE on the respective test sets. The quantile normalized 1000G dataset is exclusively used as a test set to validate cross platform application of E-GEX.

4. Experimental Results

Each of the autoencoder architectures were trained for 150 epochs and saved after their reconstruction errors were recorded. The encoded target expressions were written onto disk for training the prediction networks. Training of E-GEX MLP networks were done for a maximum of 250 epochs. The test sets were run through the saved MLP models to obtain the predicted target encodings, which were decoded using the saved autoencoder models. The final decoded target expressions were used to compute the MAE of the models.

Some MLP networks were also trained using the actual expression values, bypassing the autoencoder phase. These models serve the purpose of evaluating the performance improvement achieved solely by the modifications made to the MLP network [8, 17]. All the models were trained and tested using the Tensorflow library in Python on an i7-3770 CPU@ 3.40GHz x8 [14].

4.1. Performance of Models on GEO-norm

Two 3-layer feed-forward networks with 9000 nodes in each layer were trained on GEO-tr for 250 epochs, one for target genes 0-4760 and another for genes 4760-9520. The parameters were directly adapted from the best performing D-GEX model [8]. Activation of lower layers was done using hyperbolic tangent function. Output units had linear activation. Momentum technique was used for optimization. Dropout was performed with 10% dropout rate. This model served as the benchmark that was used to select the best performing E-GEX network.

Firstly, we measured the performance of the modified feed-forward network that addresses certain shortcomings of D-GEX. We constructed and trained two 2-hidden layer models with 12000 nodes in each layer. The activation function for lower layers was the exponential linear function, and linear function for the output layer. Early stopping regularization was used alongside dropout with 10% dropout rate. Optimization was done using NAG technique. MAE of both the networks was combined to produce the total error rate for genes 0-9520, which is described in Table 1. It can be observed that these modifications brought forth a 16.64% improvement in test error over D-GEX.

Table 1. Validation and test errors of different models.

Model	Error on GEO-val	Error on GEO-ts
D-GEX	1.1342	1.2927
E-GEX MLP	0.8988	1.0789
E-GEX	0.9833	1.0775

Secondly, we encoded the target genes into a 2000-feature representation using a deep autoencoder with 3 hidden layers. The number of nodes in the first and last hidden layers was varied to find the configuration that resulted in the least reconstruction error. From Table 2, it is evident that the 6000x2000x6000 configuration produced the best 3-layer encoding.

Table 2. Three-hidden layer autoencoders with tied weights.

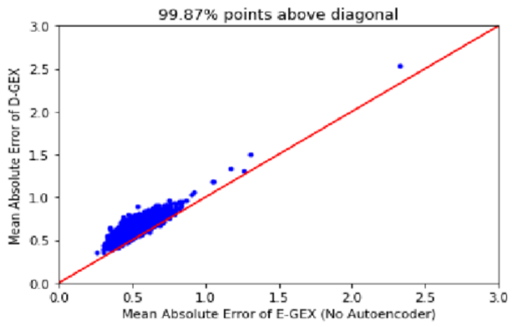
Number of Nodes in layers 1 and 3	Reconstruction Error	Training time per epoch (minutes)
5000	0.5346	40
6000	0.4468	45
8000	0.4971	55

The second phase of E-GEX is identical to the E-GEX MLP described above, with the exception of its hidden layer configuration which was tweaked to accommodate the change in the number of output nodes. The number of nodes in the first two hidden layers was 11850 and 11826, respectively [17]. The decoded target predictions were compared to the original target expressions to derive the final error rate mentioned in Table 1. The encoded prediction model showcased performance slightly superior to E-GEX MLP. It can be observed that the reconstruction error of 0.4478 can be reduced substantially by training a deeper autoencoder, which would further pull down the overall predictive error. Figure 5 establishes the superiority of E-GEX model over D-GEX, in terms of the validation errors.

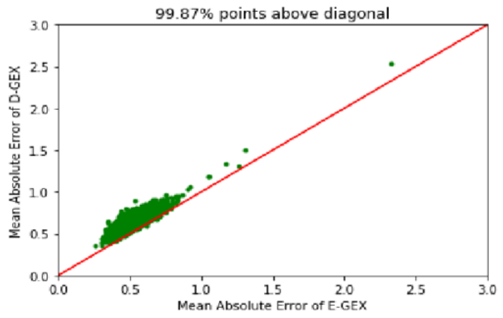
The expressions of each target gene predicted by E-GEX and D-GEX were compared, which is represented in Figure 8. E-GEX produced better predictions for 99.87% of the target genes, with or without the autoencoder phase.

4.2. Testing E-GEX on 1000G

All the three models described above were tested on the 1000G dataset, and the results are described in Table 3. Al-though the predictive errors of D-GEX and E-GEX MLP were found to be identical, encoding the target before prediction boosted prediction accuracy by 49.23%.



a) Without autoencoder.



b) With autoencoder.

Figure 5. Comparison of validation errors.

Table 3. Predictive errors on 1000G.

Model	Genes (0-4760)	Genes (4760-9520)	Total
D-GEX	0.7756	0.7757	1.5513
E-GEX MLP	0.7750	0.7754	1.5504
E-GEX	0.3794	0.4082	0.7876

From Figure 6 we observed that the predictive error on 1000G by D-GEX was almost halved by E-GEX, although the models were trained on GEO-norm. Thus, E-GEX can be employed across platforms, much more effectively than D-GEX.

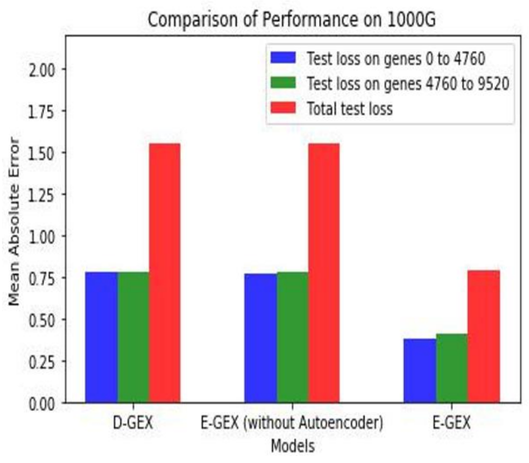


Figure 6. Performance of different models on 1000G dataset.

4.3. Performance of E-GEX on full GEO data

Encouraged by the performance of E-GEX on GEO-norm, a similar model was constructed and trained for the entire GEO dataset with 978 landmark genes and 21290 target genes. An autoencoder with 3 hidden layers of 8000, 2000 and 8000 nodes respectively was used to encode the target expressions. The parameters for the predictive network were directly adapted from the previous model. Several hidden layer configuration were tried, but it is clear from Table 4 and Figure 7 that the 11850 x11826 configuration was the best among the lot.

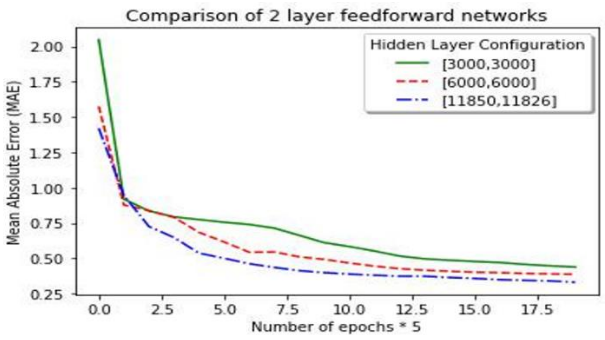


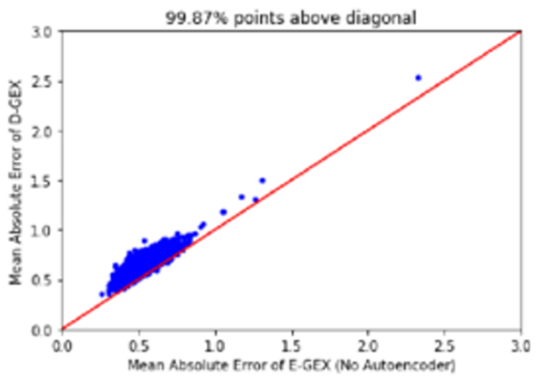
Figure 7. Effect of hidden layer configuration of E-GEX on validation loss.

Table 4. Validation Errors in 2-hidden layer feed-forward networks of E-GEX.

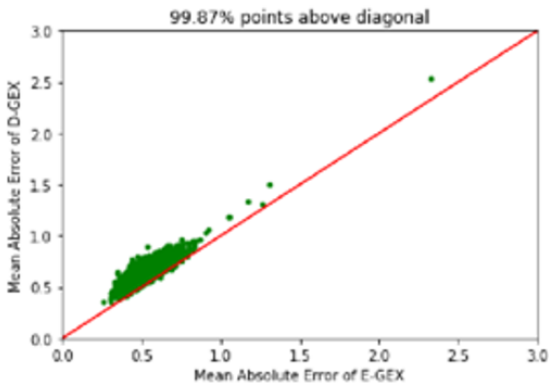
Number of nodes in layer 1	**Number of nodes in layer 2**	**Mean Absolute Error (MAE)**
3000	3000	0.464
3000	6000	0.423
6000	6000	0.389
6000	9000	0.351
9000	9000	0.334
11850	11826	0.257

The candidate configurations were adopted from the different models that were tried as part of the D-GEX project. The 3-hidden layer 9000x9000x9000 configuration produced the best result on normalized GEO data in D-GEX. Since the output nodes have been significantly reduced by the encoding phase, there was no need to look beyond 2-hidden layer feed-forward networks.

Deeper MLPs would usher in performance deficiencies in the form of the vanishing gradient problem since we are not employing higher order derivatives [15]. For the consolidated model that performs genome- wide microarray based profiling, the MAE on the test set was observed to be 1.8484. The principal contributor to this quantity was the reconstruction error of the autoencoder, which was close to 1.81. Although deeper encoders can produce better representations with minimum reconstruction error, hardware limitations hampered our attempts to increase the depth of the autoencoders.



a) Without autoencoder.



b) With autoencoder.

Figure 8. Comparison of Gene-wise errors.

5. Conclusions and Future Work

Characterising the gene expression patterns of genes under various biological environments is a fundamental problem in molecular biology. Cost constraints steer researchers away from explicit profiling through laboratorial techniques like the L1000 Luminex bead technology [25]. E-GEX presents a statistical approach to profile thousands of genes at once using deep neural networks. It is an extension of the D-GEX project, which used separately trained Multilayer Perceptron networks to predict expressions of randomly partitioned sets of target genes [8]. The autoencoder phase of E-GEX overcomes this redundancy and enables the construction of a single trained MLP to profile complete genome expressions. In addition to that, modifications are made to the MLP network in terms of its activation function, optimization technique, regularization strategy and hidden layer configuration. These alterations alone resulted in a 16.64% improvement in performance on the normalized GEO data. The MLP training procedure is simplified using the autoencoder phase. Since the mean absolute error on the RNA-Seq based 1000G data is lesser than that of the test error on GEO dataset, it can be inferred that good generalization has been achieved. The E-GEX model brought forth a 49.23% performance boost over D-GEX on the 1000G data. Thus we strongly believe that E-GEX is an efficient and accurate model for gene expression prediction. In the current implementation, the depth of the autoencoder is severely restricted due to hardware limitations. Architectures with more number of hidden layers with a gradual decrease in the number of neurons in each successive layer of the encoder phase can further reduce the reconstruction error which would result in better encoding [31]. Forage into recent developments in multi-GPU techniques would accelerate training and also enable us to accommodate deeper architectures [8, 9]. Deep networks could be efficiently trained layer by layer in a greedy manner using GPUs. Recurrence of the vanishing gradient problem in such deep networks could be tackled using the Batch Normalization technique for weight tuning [18].

References

[1] Arel I., Rose D., and Karnowski T., "Deep Machine Learning-A New Frontier in Artificial Intelligence Research," *IEEE Computational Intelligence Magazine*, vol. 5, no. 4, pp. 13-18 2010.

[2] Baldi P., "Autoencoders, Unsupervised Learning, and Deep Architectures," *in Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, Washington, pp. 37-50, 2012.

[3] Baldi P. and Sadowski P., "Understanding Dropout," *in Proceedings of Neural Information Processing Systems*, pp. 2814-2822, 2013.

[4] Bansal M., Belcastro V., Ambesi-Impiombato A., and Bernardo D., "How to infer gene networks from expression profiles," *Molecular Systems Biology*, vol. 3, no. 78, pp. 1-10, 2007.

[5] Bengio Y., "Learning Deep Architectures for AI," *Foundations and Trends® in Machine Learning*, vol. 2, no. 1, pp. 1-127, 2009.

[6] Bengio Y., Courville A., and Vincent P., "Representation Learning: A Review and New Perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798-1828, 2013.

[7] Caruana R., Lawrence S., and Giles L., "Overfitting in Neural Nets: Backpropagation, Conjugate Gradient, and Early Stopping," *Advances in Neural Information Processing Systems*, pp. 402-408, 2001.

[8] Chen Y., Li Y., Narayan R., Subramanian A., and Xie X., "Gene Expression Inference with Deep Learning," *Bioinformatics*, vol. 32, no. 12, pp. 1832-1839, 2016.

[9] Chen L., Villa O., Krishnamoorthy S., and Gao G., "Dynamic Load Balancing on Single- And Multi-GPU Systems," *in Proceedings of IEEE International Parallel and Distributed Processing Symposium*, Georgia, pp. 1-12, 2010.

[10] Clevert D., Unterthiner T., and Hochreiter S., "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUS)," *arXiv preprint arXiv:1511.07289*, pp. 1-14, 2015.

[11] De Sousa C., "An Overview on Weight Initialization Methods for Feedforward Neural Networks," *in Proceedings of the International Joint Conference on Neural Networks*, pp. 52-59, 2016.

[12] Edgar R., Domrachev M., and Lash A., "Gene Expression Omnibus: NCBI Gene Expression and Hybridization Array Data Repository," *Nucleic Acids Research*, vol. 30, no. 1, pp. 207-210, 2002.

[13] Forgy E., "Cluster Analysis of Multivariate Data: Efficiency versus Interpretability of Classifications," *Biometrics*, vol. 21, pp. 768-769 1965.

[14] Géron A., *Hands-On Machine Learning with Scikit-Learn and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems*, O'Reilly Media, 2019.

[15] Glorot X. and Bengio Y., "Understanding the Difficulty of Training Deep Feedforward Neural Networks," *in Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, Sardinia, pp. 249-256, 2010.

[16] Glorot X., Bordes A., and Bengio Y., "Deep Sparse Rectifier Neural Networks," *in Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, FL, pp. 315-323, 2011.

[17] Huang G., "Learning Capability and Storage Capacity of Two-Hidden-Layer Feedforward Networks," *IEEE Transactions on Neural Networks*, vol. 14, no. 2, pp. 274-281, 2003.

[18] Ioffe S. and Szegedy C., "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *arXiv preprint arXiv:1502.03167*, pp. 137-141, 2015.

[19] Kassam S., "Quantization Based on the Mean-Absolute-Error Criterion," *IEEE Transactions on Communications*, vol. 26, no. 2, pp. 267-270, 1978.

[20] Lamb J., Crawford E., Peck D., Model J., Blat I., Wrobel M., Lerner J., Brunet J., Subramanian A., Ross K., Reich M., Hieronymus H., Wei G., Armstrong S., Haggarty S., Clemons P., Wei R., Carr S., Lander E., and Golub T., "The Connectivity Map: Using Gene-Expression Signatures to Connect Small Molecules, Genes, And Disease," *Science*, vol. 313, pp. 1929-1935 2006.

[21] Lecun Y., Bengio Y., and Hinton G., "Deep Learning," *Nature*, vol. 521, no. 7553, pp. 436-444, 2015.

[22] Le Q., Ranzato M., Monga R., Devin M., Chen K., Corrado G., Dean J., and Ng A., "Building High-Level Features Using Large Scale Unsupervised Learning," *in Proceedings of the 29th International Conference on Machine Learning*, Scotland, pp. 8595-8598, 2011.

[23] Lin C., Jain S., Kim H., and Bar-Joseph Z., "Using Neural Networks For Reducing The Dimensions Of Single-Cell RNA-Seq Data," *Nucleic Acids Research*, vol. 45, no. 17, pp. 1-11, 2017.

[24] Nesterov Y., "A Method of Solving A Convex Programming Problem with Convergence Rate O(1/k^2)," *Doklady Mathematics*, vol. 27, no. 2, pp. 372-376, 1983.

[25] NIH LINCS Program. http://lincsproject.org/, Available at: www.lincsproject.org, Last Visited, 2018.

[26] Pierson E. and Yau C., "ZIFA: Dimensionality Reduction for Zero-Inflated Single-Cell Gene Expression Analysis," *Genome Biology*, vol. 16, no. 1, 2015.

[27] Polyak B., "Some Methods of Speeding Up the Convergence of Iteration Methods," *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1-17, 1964.

[28] Rumelhart E., Hinton E., and Williams J., "Learning Representations by Back-Propagating Errors," *Nature*, vol. 323, no. 6088, pp. 533-536, 1986.

[29] Senior A., Heigold G., Ranzato M., and Yang K., "An Empirical Study of Learning Rates in Deep Neural Networks for Speech Recognition," *in Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, Vancouver, pp. 6724-6728, 2013.

[30] Srivastava N., Hinton G., Krizhevsky A., Sutskever I., and Salakhutdinov R., "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929-1958, 2014.

[31] Vincent P. and Larochelle H., "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion Pierre-Antoine Manzagol," *Journal of Machine Learning Research*, vol. 11, pp. 3371-3408, 2010.

[32] Vincent P., Larochelle H., Bengio Y., and Manzagol P., "Extracting And Composing Robust Features With Denoising Autoencoders," *in Proceedings of The 25th International Conference on Machine Learning*, NY, pp. 1096-1103, 2008.

Raju Bhukya has received his B.Tech in Computer Science and Engineering from Nagarjuna University in the year 2003, M.Tech degree in Computer Science and Engineering from Andhra University in the year 2005 and P.hD in Computer Science and Engineering from National Institute of Technology (NIT) Warangal in the year 2014. He is currently working as an Assistant Professor in the Department of Computer Science and Engineering in National Institute of Technology, Warangal, Telangana, India. He is currently working in the areas of Bio-Informatics and Data Mining.

Achyuth Ashok is M.Tech student of CSE department at NIT Warangal. He has interest in analyzing information contained in genome sequences using deep learning to predic DNA sequences and time involved in DNA Sequence Analysis.