# Loss Analysis of Denoising Autoencoders on Gene Expression Data

Prikshet Sharma

May 1, 2019

## 1 Abstract

In this paper, we test various autoencoder models to denoise gene expression profiles. To train the denoising autoencoders, we create multiple datasets by adding various noise levels to the original data. The input values are taken from the noised datasets, whereas the target values are taken from the original dataset. Deep autoencoders of varying from two hidden layers to eight hidden layers are tested. Additionally, in order to find any spatial patterns in the datatset, a convolutional deep autoencoder is also tested.

## 2 Motivation

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.[1]
Unsupervised feature learning methods, such as autoencoders may provide some insights on this problem.[1]

## 3 Gene Expression

The amount of spliced mRNA that is produced in a cell is referred to as the level of gene expression, which is a function of the rate of synthesis of new mRNA and of its degradation. Gene expression is the process by which the genetic blueprint is translated into functional, biologically active units. This constellation of RNAs drives essential biological processes, such as transcriptional regulation, cellular differentiation, and when dysregulated, the development of complex disease.[2] For example, different types of cancer have been shown to have different RNA profiles.[3]

Several different ways have been developed to determine the RNA expression levels of all genes in the genome, called genome-wide expression analysis such as hybridization, microarray assays, RNA-Seq etc. [3] The data used for this paper was collected using RNA-Seq.

# 4 RNA-Seq

RNA-Seq delivers a high-resolution, base-by-base view of coding and noncoding RNA activity, providing a comprehensive picture of gene expression across the full transcriptome (sum total of all mRNA expressed) at a specific moment.[2]

# 5 The Data

The RNA-Seq data was taken from the Genotype-Tissue Expression (GTEx) database[4]. The data consists of gene expression profiles of the L1000 genes of 2921 cells and perturbations. L1000 genes or "landmark genes" are ones that computational analysis of large gene expression compendia has shown would be feasible to derive sufficient information about the complete transcriptional state of the cell.[5] 80% of the 2921 cells were used for training and validation, whereas the other 20% were used for testing. Ten additional training and test datasets were created by adding various noise levels to the original data. The ten noise factors in the noisy datasets ranged from 0.1 to 1.0 with 0.1 increments.

# 6 Autoencoders

An autoencoder neural network is an unsupervised learning algorithm that applies backpropagation, setting the target values to be equal to the inputs. In other words, it tries to learn an approximation to the identity function. The identity function seems a particularly trivial function to be trying to learn; but by placing constraints on the network, such as by limiting the number of hidden units, we can discover interesting structure about the data. If we impose a "sparsity" constraint on the hidden units, then the autoencoder will still discover interesting structure in the data, even if the number of hidden units is large.[6] If an autoencoder successfully learns the overall structure of the L1000 genes for some cells and perturbations, then it can denoise gene expression for the rest of the 21000 genes for those cells and perturbations.

## 6.1 Training

For a one-hidden-layer autoencoder, let,

$$\hat{\rho}_j = \frac{1}{m} \sum_{i=1}^{m} [a_j^{(2)}(x^{(i)})] \tag{1}$$

be the average activation of hidden unit $j$, (averaged over the training set), where $m$ is the total number of units in the network. $a_j^{(2)}(x)$ is the activation of hidden unit $j$ on input x. We would like to (approximately) enforce the constraint

$$\hat{\rho} = \rho, \tag{2}$$

where $\rho$ is a "sparsity parameter", typically a small value close to zero (say $\rho = 0.05$). In other words, we would like the average activation of each hidden neuron $j$ to be close to 0.05 (say). To satisfy this constraint, the hidden unit's activations must mostly be near 0.
To achieve this, we will add an extra penalty term to our optimization objective that penalizes $\hat{\rho}_j$ deviating significantly from $\rho$. Many choices of the penalty term will give reasonable results. We will choose the following:

$$\sum_{j=1}^{s_2} \rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_j} \tag{3}$$

Here, $s_2$ is the number of neurons in the hidden layer, and the index $j$ is summing over the hidden units in our network.
This penalty term is based on KL divergence, and can be written

$$\sum_{j=1}^{s_2} KL(D(\rho \parallel \hat{\rho}_j)), \tag{4}$$

where

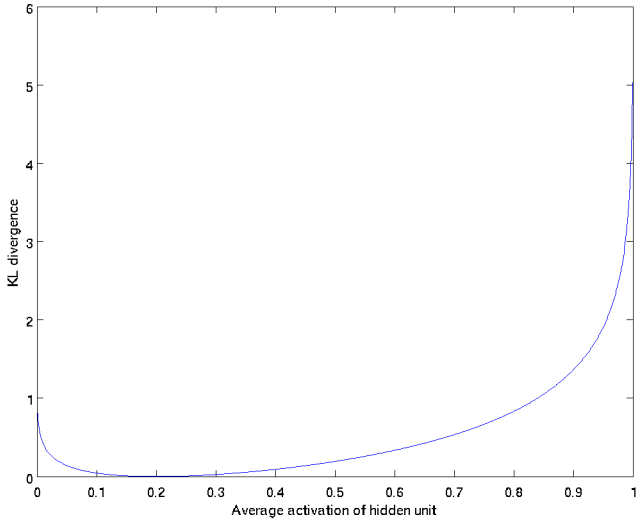
$$KL(D(\rho \parallel \hat{\rho}_j)) = \rho \log \frac{\rho}{\hat{\rho}_j} + (2-\rho) \log \frac{1-\rho}{1-\hat{\rho}_j} \tag{5}$$

is the Kullback-Leibler (KL) divergence between a Bernoulli random variable with mean $\rho$ and a Bernoulli random variable with mean $\rho_j$. KL-divergence is a standard function for measuring how different two different distributions are. This penalty function has the property that $KL(D(\rho \parallel \hat{\rho}_j)) = 0$ if $\hat{\rho}_j = \rho$, and otherwise it increases monotonically as $\hat{\rho}_j$ diverges from $\rho$.[6]

For example, in the figure below, we have set $\rho = 0.2$, and plotted $KL(D(\rho \parallel \hat{\rho}_j))$ for a range of values of $\hat{\rho}_j$.

We see that the KL-divergence reaches its minimum of 0 at

$$\hat{\rho}_j = \rho, \tag{6}$$




[6]

Figure 1: KL divergence vs average activation of hidden unit

and approaches inf as $\hat{\rho}_j$ approaches 0 or 1.

Our overall cost function is now,

$$J_{sparse}(W,b) = J(W,b) + \beta \sum_{j=1}^{s_2} KL(D(\rho \parallel \hat{\rho}_j)) \tag{7}$$

where $J(W,b)$ is the loss function of the autoencoder without the sparsity constraint, and $\beta$ controls the weight of the sparsity penalty term.

## 6.2 Denoising Autoencoder

Denoising autoencoders are autoencoders that, instead of inference, denoise the input data. Converting a regular autoencoder to a denoising autoencoder is simple. Instead of having the input values to be the same as the target values, the input values are the noisy values and the target values are the original values. For this study, five denoising autoencoders models were trained, validated and tested on the gene expression data for their denoising performance. The performance metric was the mean square error (MSE) between the original images and their denoised counterparts. Five deep autoencoder models and one deep convolutional autoencoder model were tested for their performance.

| Model | Hidden Layer(s) | Units in each layer |
|---|---|---|
| 1 | 2 | 943, 32, 32, 943 |
| 2 | 4 | 943, 64, 32, 32, 64, 943 |
| 3 | 6 | 943, 128, 64, 32, 32, 64, 128, 943 |
| 4 | 8 | 943, 256, 128, 64, 32, 32, 64, 128, 256, 943 |

Figure 2: Deep layers vs MSE

### 6.3 Deep Autoencoder Model Specifications

A deep autoencoder is one that has more than one hidden layer. Five Deep autoencoder models were tested. Their specifications are:

## 7 Deep Convolutional Autoencoder

A deep convolutional autoencoder is an autoencoder that has convolutional layers. Convolutional autoencoders are popular and perform well in computer vision tasks. This is because images have patterns or features. Convolutional neural networks exploit this property in that convolutional filters learn features and patterns in the images.
The deep convolutional autoencoder model was tested to find any patterns and features in the genomics data. If deep convolutional autoencoders significantly perform better than deep autoencoders, this would suggest that the genomics data have spatial features and patterns.
The autoencoder model used was the following:

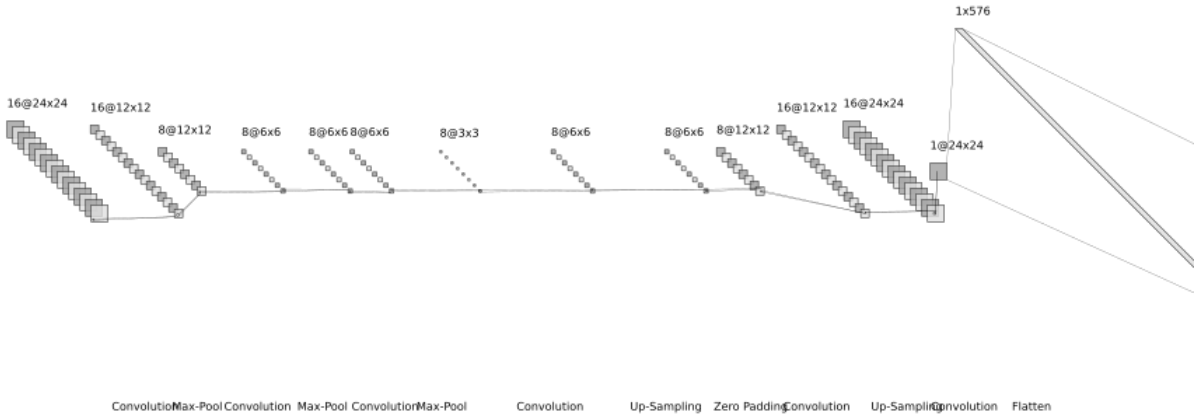



Figure 3: The convolutional autoencoder used for extracting spatial features in the data, if any. Its dimensionality is minimum in the middle and fans out to the left (encoder), and the right (decoder).

# 8 Mean Square Error Loss

Let the number of samples be $N$, and let the number of genes profiled in each sample be $M$. There are $N = 2921$ samples in the dataset, each sample profiled $M = 943$ special genes, known as the L1000 genes, on some cell with some perturbation. Let the set of all samples be $S$, where

$$S = \{s_i \mid i \in [0, N), s_i \in \mathbb{R}^M\} \tag{8}$$

where $s_i$ is the i'th sample.

The noisy input data is created by adding Gaussian white noise to the whole

dataset as follows:

$$\forall i, j \; s'_{ij} \leftarrow s_{ij} + \alpha\mathcal{N}(0,1) \tag{9}$$

$s'_{ij}$ is the profile for the i'th sample and the j'th gene, $\mathcal{N}(0,1)$ is the normal gaussian random variable, and $\alpha$ is the noise factor.
Therefore, the noisy set $S'$ is defined as:

$$S' = \{s'_i \mid i \in [0, N)\} \tag{10}$$

where $s_i$ is the i'th sample.

And we define the Mean Square Error or MSE between the set $S$ and $S'$ as follows:

$$MSE(S, S') = \frac{1}{MN} \sum_{i=0,j=0}^{M,N} (s_{ij} - s'_{ij})^2 \tag{11}$$

# 9 Performance

## 9.1 Varying the Number of Deep Layers

First we test whether adding more deep layers decreases the Mean Square Error, i.e., increases the denoising accuracy of the autoencoder. Following is the table of Mean Square Errors versus the number of deep layers. These results are for autoencoders that were trained with 0.5 noise factor images.

| Layers | MSE |
|---|---|
| 2 | 0.000216 |
| 4 | 0.00029524 |
| 6 | 0.00031907 |
| 8 | 0.00034364 |

Figure 4: Number of (deep) autoencoder layers vs Mean Square Error Table Plot

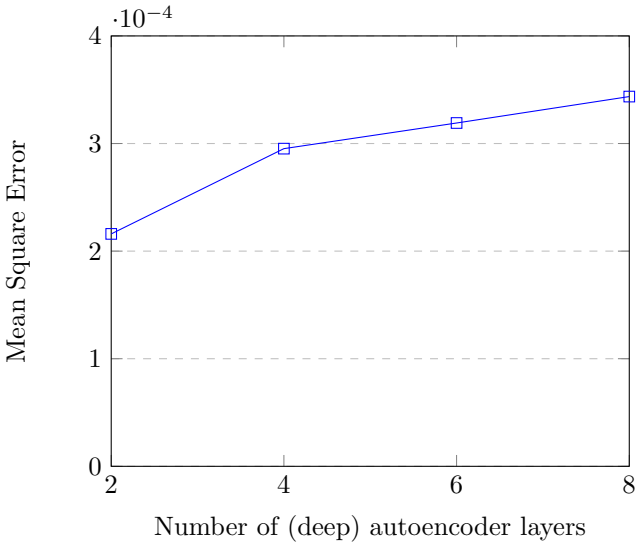



Figure 5: Number of (deep) autoencoder layers vs Mean Square Error

As the number of layers increases, the autoencoder's performance worsens when trained with 0.5 noise factor images. In fact, the same pattern appeared for all noise factors from 0.0 (no noise) to 1.0 with 0.1 increments. This indicates that the adding more autoencoder layers overfits the data.[7]. This motivates a reduction in the sparsity constraint, and/or the reduction of minimum dimensionality of hidden units down from 32. These modifications and their analysis are subject to future work.

## 9.2 Varying the Noise Levels

We plot the MSE vs noise factor for deep layers 2, 4, 6, 8.

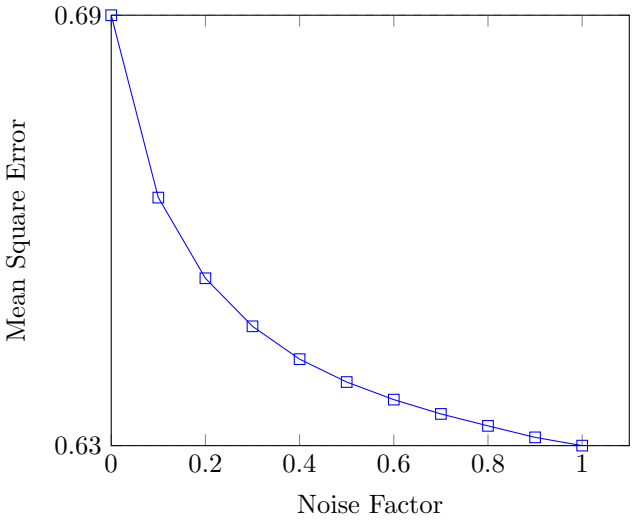



Figure 6: Two Layer Autoencoder MSE vs Noise Factor

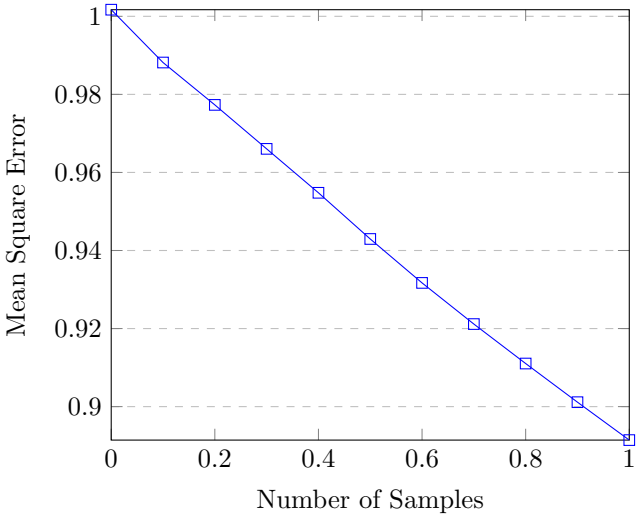



Figure 7: Four Layer Autoencoder MSE vs Noise Factor

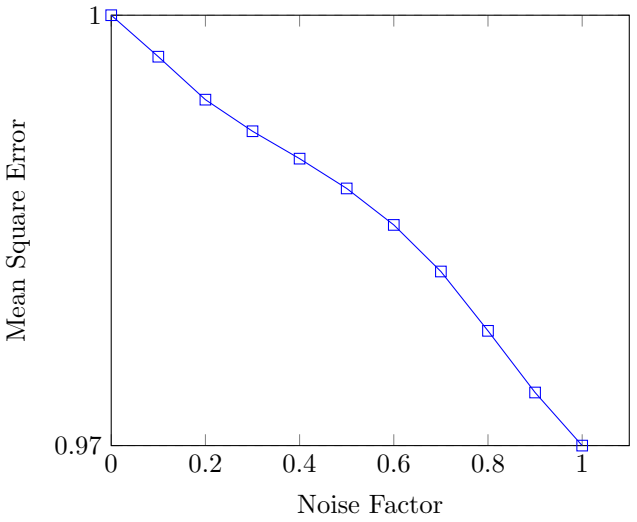



Figure 8: Six Layer Autoencoder MSE vs Noise Factor

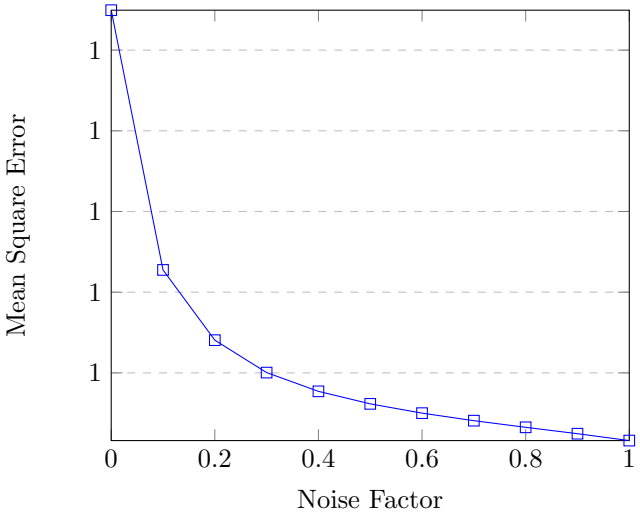



Figure 9: Eight Layer Autoencoder MSE vs Noise Factor

These graphs are especially confounding, because they show that for every model, adding more noise in the input training set increases the performance of denoising a test set that has the same noise level as the noise levels of the input training set. This suggests that the autoencoder overtrains on data with lower noise. A solution is to decrease the minimum dimensionality of the network while maintaining number of deep layers.

Additionally, we need to test the performance when the noise levels of the input data are different from the noise levels of the test set, and whether training the network with multiple noise levels gives better results.

### 9.3 Convolutional Autoencoder

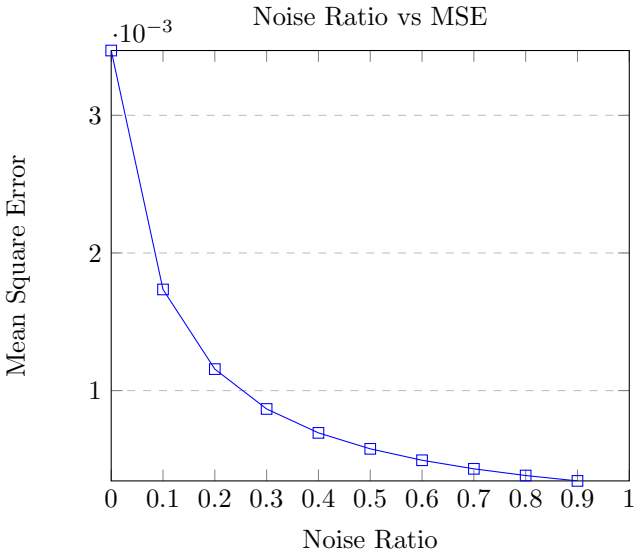



Figure 10: Noise Levels vs MSE of the Convolutional Autoencoder

This graph shows that the convolutional autoencoder also tends to overfit as the noise increases. However, all mean square values on the y-axis are two orders of magnitude less than the deep autoencoders. This suggests that there might be spatial features in the genomics data. Finding these features is a topic of further research and might involve novel visualizations of genomics samples.

## 10 Conclusion

Autoencoders are usually used to reduce overfitting, however in our case on all the models, increasing the noise level of the training dataset decreses the Mean Square Error, suggesting that the data overfitted. A solution proposed is reducing the minimum dimensionality of the networks.

On the other hand, the convolutional autoencoder performs better than deep autoencoders, suggesting that there are spatial patterns/features in the gene expression profiling dataset. Visualizations of these patterns and features can help us better understand what these features are.

# References


[1] "Gene expression inference with deep learning." https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4908320/. Accessed: 5/1/2019.

[2] "The next generation of gene expression profiling with rna-seq." https://www.illumina.com/content/dam/illumina-marketing/documents/gated/gene-expression-profiling-e-book-web.pdf. Accessed: 5/1/2019.

[3] P. N. Benfey, *Quickstart Molecular Biology*. Duke Center for Systems Biology, 2014.

[4] "Gtex portal." https://gtexportal.org/home/. Accessed: 5/1/2019.

[5] "What is the l1000 assay?." https://clue.io/connectopedia/what_is_l1000. Accessed: 5/1/2019.

[6] "Autoencoders." http://ufldl.stanford.edu/tutorial/unsupervised/Autoencoders/. Accessed: 5/1/2019.

[7] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.