# ECE521 Assignment 2: Logistic Regression and Neural Networks

February 27, 2017

Raymond Ly (999959497)
Jianwei Sun (1000009821)

# 1. Logistic Regression

### 1.1.1 Learning

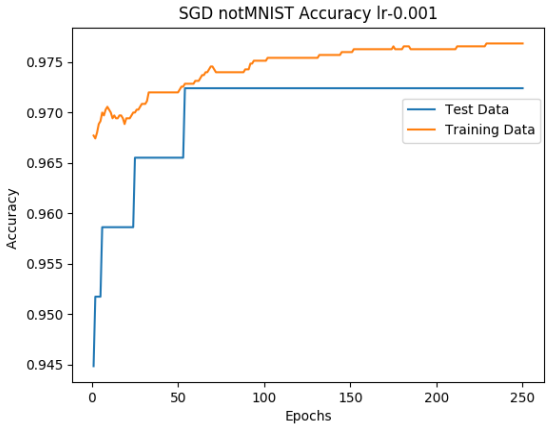


Figure 1.1.1a

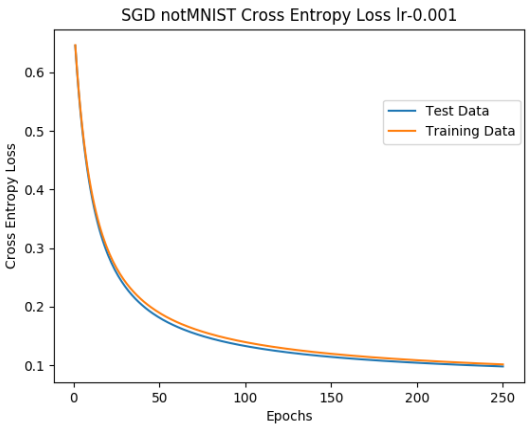


Figure 1.1.1b

Using the logistic regression model and with an optimal learning rate of 0.001, weight decay coefficient of 0.01, and a minibatch size of 500, the best classification accuracy on the test set was 97.25% which can be seen in the figure above. On the training set, the accuracy was 98%.

### 1.1.2 Beyond plain SGD

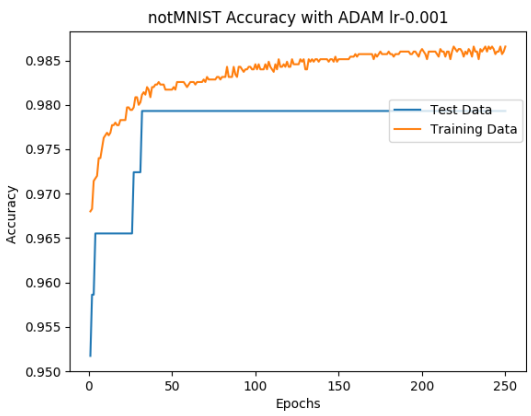


Figure 1.1.2a

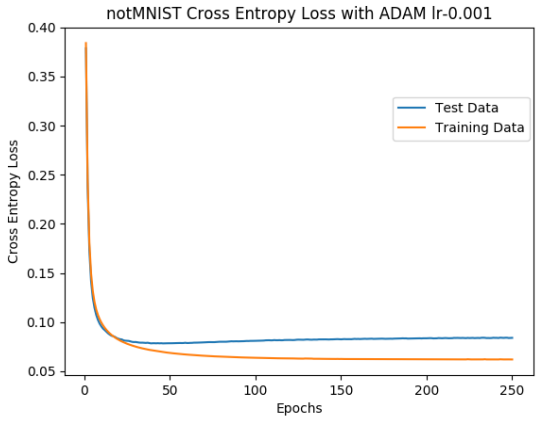


Figure 1.1.2b

The above two figures, generated using the Adam-Optimizer, have both a higher final classification accuracy and a lower cross entropy loss than the results from using SGD. The final training classification rate converges to approximately 98.5% and the final test classification accuracy is 98%. In addition, the convergence of the cross entropy loss is overall faster with the Adam-Optimizer.

The Adam-Optimizer keeps track of the exponentially decaying average of past squared gradient similar to momentum. The momentum allows the loss function to more quickly arrive at a local minimum which can be seen in the figure. The initial decrease in the cross entropy loss is much more steep.

### 1.1.3 Comparison with linear regression

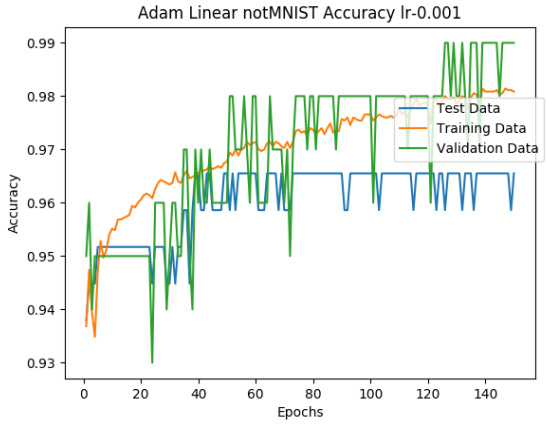


Figure 1.1.3a

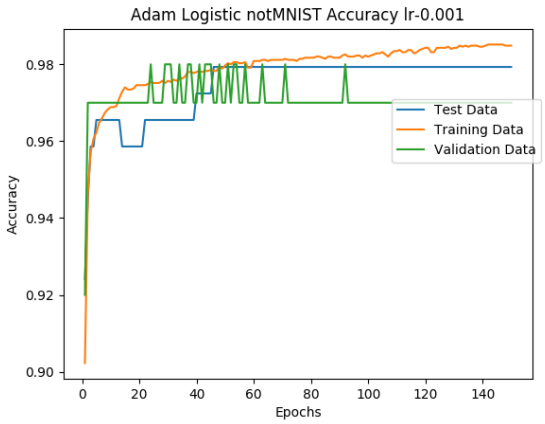


Figure 1.1.3b

Overall, without weight decay, the classification accuracy of the linear regression on the notMNIST dataset was worse than classification accuracy from the logistic regression. The final classification accuracy for the test set for logistic regression was 98% whereas the final classification accuracy is 96.5% in the linear regression case. The weights were also trained for a longer duration, 500 epochs, than the graphs reported above and though the training data accuracy continued to increase, the test and validation data accuracy stayed the same and the test and validation data loss had some decline.

Cross-entropy loss is more appropriate for this classification task. The loss value vs prediction for the Sum of Squares method and the Cross-entropy loss method is visualized in the figure below. Since the target value is zero, the loss at a prediction of zero is zero as well. As the prediction strays further and further from the target value, the loss value of the Cross-entropy method increases at a higher rate than the Sum of Squares method. This causes existing disparities between the target value and the predicted value to be more pronounced in the Cross-entropy method. As a result, the classification performance of the Cross-entropy loss was higher.


Figure 1.1.3c

### 1.1.4 Maximum likelihood estimation

From the Bernoulli distribution, we have:

$$P(y|x,w) = \hat{y}(x)^{y}(1-\hat{y}(x))^{(1-y)}$$

Taking the maximum log-likelihood of the probability distribution is equivalent to minimizing the cross-entropy loss, as shown below:

$$max\ log(P(y|x,w) = max(y log(\hat{y}(x)) + (1-y) log(1-\hat{y}(x)))$$
$$= min(-y log(\hat{y}(x)) - (1-y) log(1-\hat{y}(x)))$$

The expression in the minimum function is precisely the cross-entropy. Hence, we have shown that minimizing the cross-entropy loss is equivalent to maximizing the log-likelihood under a Bernoulli distribution.

### 1.2.1 Minimum-loss System

The expected loss is of the system is as follows:

$$E[L] = \sum_{k=1}^{m}\sum_{j=1}^{m} L_{kj} P(x \in R_j,\ C_k)$$

In the above equation, the expected loss is a weighted sum of the probabilities of misclassifying each input, x. If the penalty for misclassification is equal, then the antidiagonal of the penalty matrix, L, would have equal values. Keeping all the values to the minimum of unity would result in the following expression for the expected loss:

$$E[L] = \sum_{k=1}^{m} \sum_{j=1}^{m} L_{kj} P(x \in R_j,\ C_k) = P(mistake)$$

$$L_{kj} = 1 \text{ if } k + j = m + 1 \text{ and 0 otherwise}$$

By selecting the largest a posteriori probability, the term $P(W|x^{(1)},y^{(1)},...,x^{(m)},y^{(m)})$ is maximized. The probability of a correct classification is equal to one minus the probability of an incorrect classification: $P(x \in R_j,\ C_j) = 1 - P(mistake)$. As shown above, the probability of a misclassification is equal to the expected loss if the loss penalty is equal for each class. Hence, by selecting the largest a posteriori probability, the classifier represents a minimum-loss system is each class has equal misclassification penalty.

### 1.2.2 Complex Loss Matrix

The expected loss is a summation over all predictions and possible classes, with the loss matrix, L, representing the penalty for certain misclassifications. In order to minimize the total loss, the following quantity, representing the average loss with the common factor of $P(x \in R_j)$ eliminated, should be minimized:

$$\sum_{k} L_{kj} P(C_k,\ x \in R_j)$$

This expression essentially summarizes the total weighted error in misclassification. It states that the total error is the summation of the weighted probabilities of x belonging in class $C_k$ given that it is classified into class $C_j$. By ensuring that the total sum of erroneous classifications is minimized, we can conclude that the condition under which x is classified into class $C_j$ is valid only if the above expression is minimized.

### 1.2.3 Softmax Logistic Regression

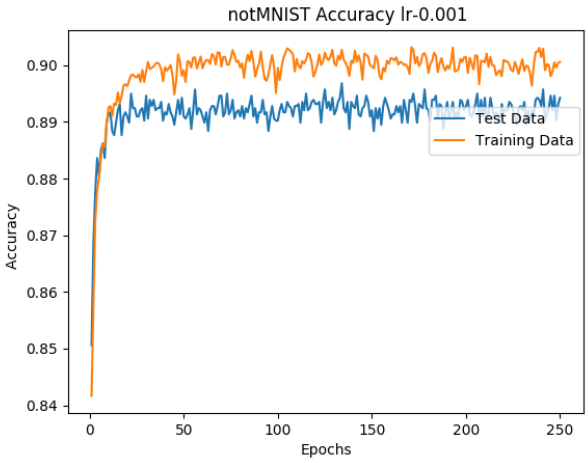


Figure 1.2.3a

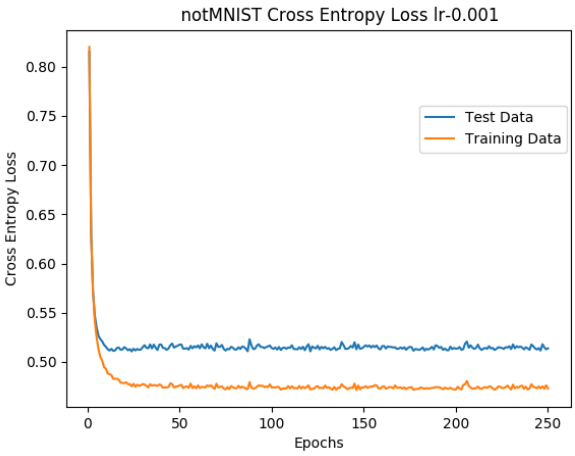


Figure 1.2.3b

The best test logistic regression classification accuracy of the the full notMNIST dataset with 10 classes and weight regulator of 0.01 is approximately 89%. In comparison to Part 1.1, the classification accuracy is a lot lower. Furthermore, the final Cross-entropy loss is also higher. With more classifications possible, each classification is given a measure of how 'likely' the data is the classification and the loss is calculated on each of these measures which introduces more loss. The classification task is much more difficult in comparison to Part 1.1. Rather than classify between two classes, this task involves classifying between 10 classes. Intuitively, the increase in classes adds a degree of classification difficulty as, because there are more classifications, it is easier to misclassify data.

# 2. Neural Networks

### 2.1.1 Linearly Separable Binary Classification

It is given that the dataset can be linearly separable into two classifications, so a sigmoid function is used. The prediction given a datapoint is generated from the following equation:

$$\widehat{y}(x) = \sigma(W^T x + b)$$

Through contradiction we can show that W will become unbounded. Let us assume towards contradiction that the W in the above equation is at an optimal, bounded value. Then, take an incremental step, $\Delta W$, in the same direction as W to compute the predictions:

$$\begin{aligned}\widehat{y}(x) &= \sigma((W + \Delta W)^T x + b) \\ &= \sigma(W^T x + b + \Delta W^T x)\end{aligned}$$

If $W^T x$ is a positive quantity, then $\widehat{y}(x)$ will tend towards unity due to the nature of the sigmoid function. Conversely, if $W^T x$ is a negative quantity, $\widehat{y}(x)$ will tend towards zero. Since $\Delta W$ is a step taking in the same direction as W, then $\sigma(W^T x + b + \Delta W^T x) > \sigma(W^T x + b)$, dealing with the positive case. Since the classification loss is a function of the difference between $\widehat{y}(x)$ and $y(x)$, the loss will decrease as $\sigma(W^T x + b + \Delta W^T x)$ tends towards unity. Furthermore, since unity is the supremum of $\sigma(W^T x + b + \Delta W^T x)$ in W, the classification loss will asymptotically tend towards zero as $W + \Delta W$ increases. Hence, a contradiction is reached: W is not the bounded optimal; and therefore, without regularization, W becomes unbounded and causes its Euclidean norm to also become unbounded.

### 2.1.2 Linearly Inseparable Binary Classification

The dataset being linearly inseparable means that a linear classification boundary cannot perfectly separate all the data points without error. This means there always exists at least one point that is misclassified. That one point has a misclassification error proportional to the difference between the actual classification and the predicted classification:

$$log(1 - \widehat{y}(x)) = log(1 - \sigma(W^T x + b))$$

The cross entropy loss is a function of the log of the difference between correct classifications and predictions. As W increases, the difference $1 - \sigma(W^T x + b)$ will decrease, which causes the logarithm to tend towards negative infinity, which will cause overall loss to grow. Hence, there exists a tradeoff between reducing the error for all the correctly classified data points and increasing the misclassification error for the set of misclassified points. Both extremes of W will result in a large loss value, so by the Mean Value Theorem there must exist a bounded optimal W between the extremes. Therefore, if W is bounded, then its Euclidean norm must also be bounded.

### 2.1.3 Linearly Inseparable Classification with Single Hidden Layer

Consider the simple dataset:
$x^{(1)} : (-1, 1),\ y^{(1)} = -1;\ x^{(2)} : (1, 1),\ y^{(2)} = 1;\ x^{(3)} : (-1, -1),\ y^{(3)} = 1;\ x^{(4)} : (1, -1),\ y^{(4)} = -1;$

This dataset is constructed such that it is linearly inseparable. Now, assuming that a single hidden layer neural net is used to classify under the cross entropy loss with no weight decay, the neural net can produce a set of W's, one corresponding to each hidden unit, that represents a set of hyperplanes in the input data space. If this set of hyperplanes is constructed so that they are aligned with the coordinate axes, then as the W's grow in magnitude, the error due to misclassification of a point in one quadrant is cancelled with the correct classification of a point in an adjacent quadrant. In this case, since there are no restrictions on W due to weight decay, the set of W's that are aligned with the coordinate axes will grow without bound. If a single set of weights becomes unbounded, then the Euclidean norm of all the weights stacked together will also be unbounded. On the other hand, a different local optimal solution could exist in which the weights for each hidden unit are identical. In this case, the locally optimal solution of a single neuron classifying this dataset (analogous to question 2.1.2) is mirrored across all the neurons. As we have shown in question 2.1.2, the weight corresponding to this one neuron for a linearly inseparable dataset without weight decay is bounded. Since this one neuron is duplicated, then stacking up all the weights from all the hidden units would result in a column vector whose Euclidean norm is also bounded.

### 2.2.1 Layer-wise Building Block

```
def add_layer(inputTensor, hiddenUnits):
    inputSize = inputTensor.get_shape().as_list()[1]
    W = tf.get_variable(name="weights", shape=[inputSize, hiddenUnits], initializer=tf.contrib.layers.xavier_initializer(uniform=False))
    b = tf.get_variable(name="biases", shape=[1, hiddenUnits], initializer=tf.constant_initializer(0))
    return tf.add(tf.matmul(inputTensor, W), b)
```

The function add_layer takes in two arguments: the input tensor of the hidden activations from the previous layer, and the number of hidden units in the current layer. The vectorized function initializes a weight tensor and a bias tensor with the appropriate dimensions and initializes the weight tensor using Xavier initialization. By using the tf.getvariable() function, the add_layer function allows these weight and bias tensors to later be accessed through named scopes for the other parts of this question.

### 2.2.2 Learning

A neural net with one hidden layer of 1000 hidden units of ReLU activation was constructed. The cross-entropy loss function along with weight decay to prevent overfitting were incorporated into the training. The neural net was then trained for learning rates of {0.1, 0.01, 0.001, 0.0001}. The plots can be seen below where lr stands for learning rate.

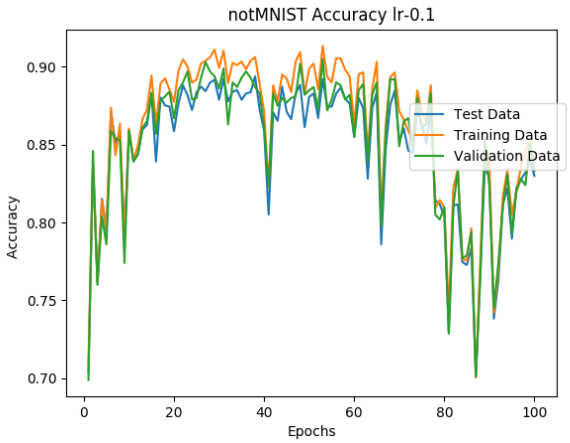


Figure 2.2.2a

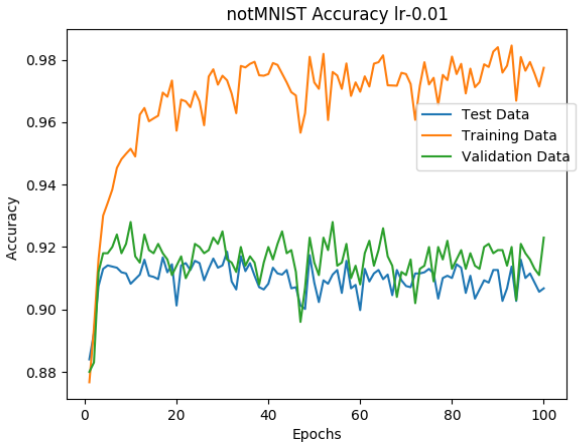


Figure 2.2.2b

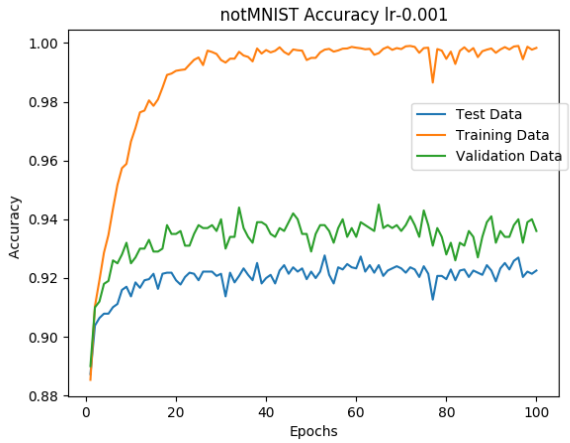


Figure 2.2.2c

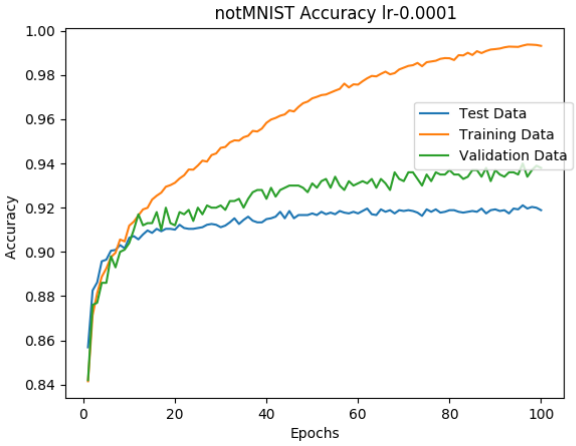


Figure 2.2.2d

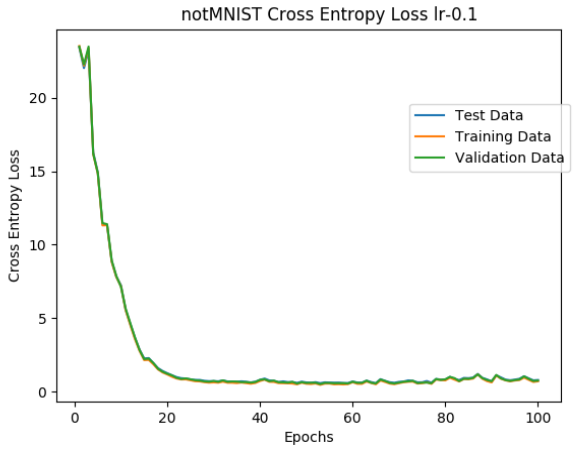


Figure 2.2.2e

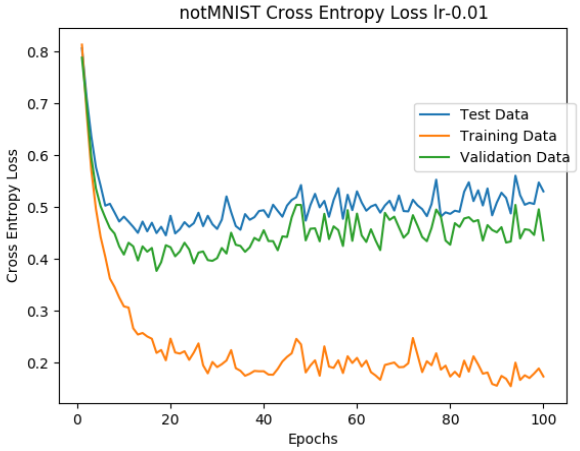


Figure 2.2.2f

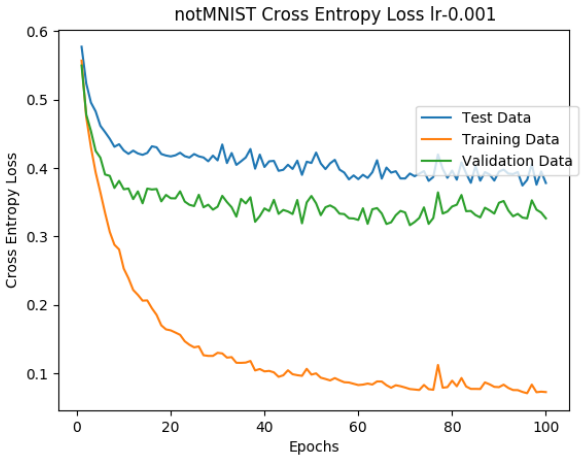


Figure 2.2.2g

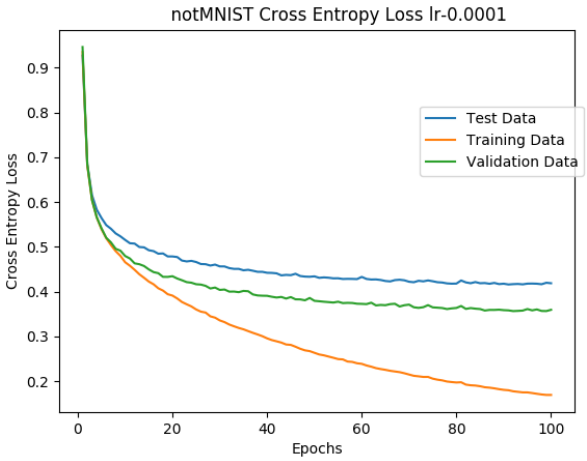


Figure 2.2.2h

From the above plots of the four different learning rates, it is evident that a learning rate of 0.001 produces the best results in terms of steady state cross entropy loss error, classification accuracy, and speed of convergence. For the learning rate of 0.001, results for the training, test, and validation accuracy and cross-entropy losses are plotted. From the accuracy plot, it is clear that the test classification error is simply the difference between its value and unity, with a smaller difference being representative of a smaller error. The number of epochs was chosen such that steady state is achieved during training.

### 2.2.3 Early Stopping

The effects of early stopping can be best noticed in the plots corresponding to a learning rate of 0.01. In the cross-entropy loss plot, it can be seen that the validation cross entropy loss starts increasing at the 25th epoch, while the training cross entropy loss continues to decrease. For better testing performance, early stopping should be applied at the 25th epoch. On the corresponding classification accuracy plot, there is a very subtle decrease in the accuracy for the testing and validation data, which agrees with the location of early stopping on the cross validation loss plot. The two plots suggest that the early stopping points are very close to each other. The cross entropy loss for the validation set should be used to determine the early stopping point, which terminates learning before overtraining occurs and the weights of the network are tuned specifically for the training data. If this overtraining occurs, the weights start to rely heavily on the idiosyncrasies of the training data and will not generalize well to other data such as the test data.

The purpose of a validation set is to tune the hyperparameters. The cross entropy loss plot should be used instead of the accuracy plot because the former reveals how “incorrect” the model is to the given data. The accuracy plot is determined by only if a classification is correct or wrong, with no measure of how far from the correct classification incorrect predictions are. Hence, a more accurate representation of the overall incorrectness of the classification is only revealed by the cross entropy loss plot.

### 2.3.1 Number of Hidden Units

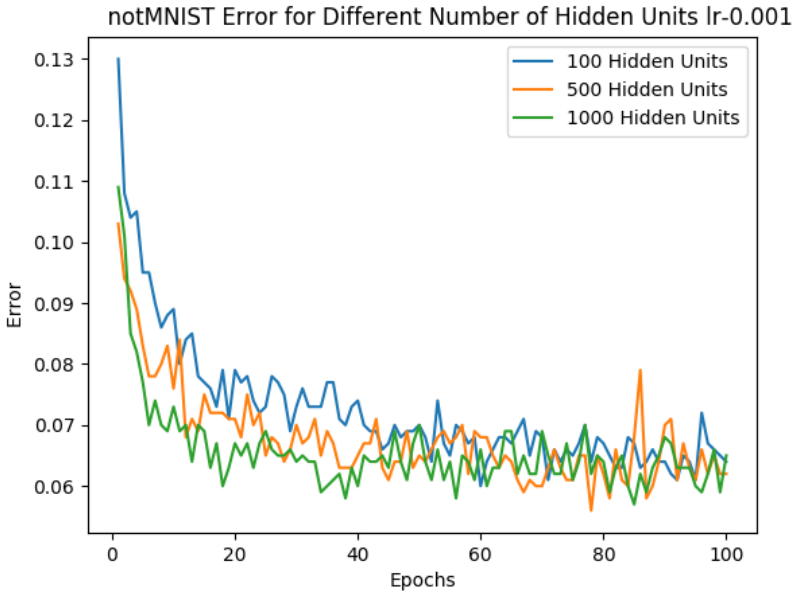


Figure 2.3.1a

The percentage of incorrectly categorized images of the validation set with various numbers of hidden units can be seen in the figure above. Overall, the classification error was lowest with the network with 1000 hidden units.

The test set accuracy can be seen in the figure from 2.2.2. The percentage error on the classification of the test set is 92%. As the number of hidden units increase, the classification performance increase as well.

### 2.3.2 Number of Layers

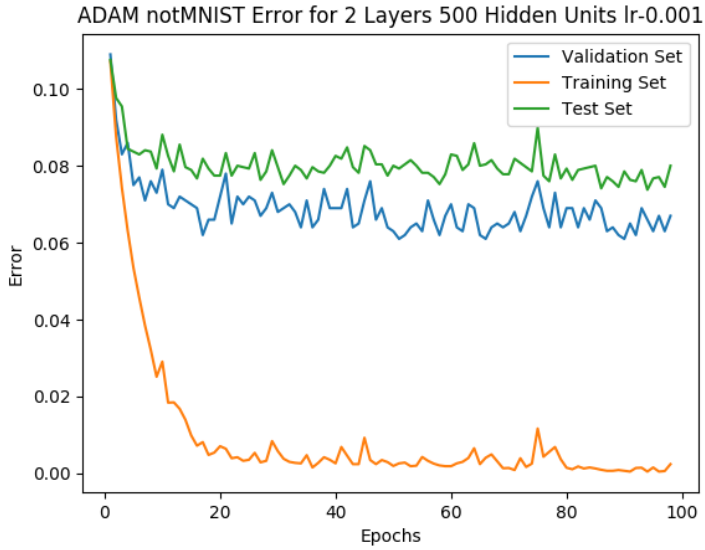


Figure 2.3.2a

The final validation error after training was 6.5% and the final test set error was 8%. In comparison to 2.2.2, a neural net with one layer of 1000 neurons, the test set performance was higher. From these results, by layering neurons and keeping the total number of hidden units constant, the performance improves. Multiple layers in the neural network allow for additional nonlinearity to be expressed which cannot be modeled by a single layer. The 2 layer neural network is able to produce a more complex model and hence, is better able to classify the dataset.

### 2.4.1 Dropout

**Without Dropout:**

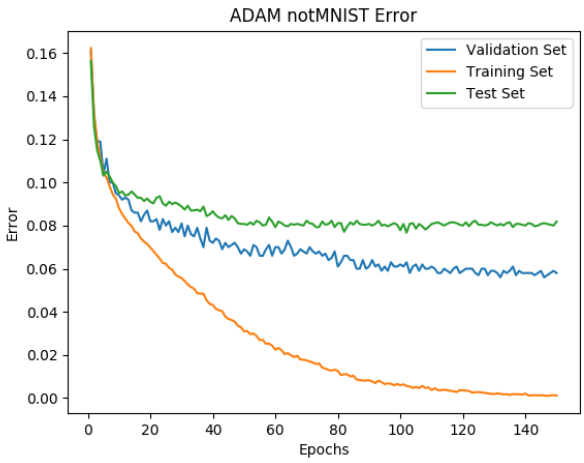


Figure 2.4.1a

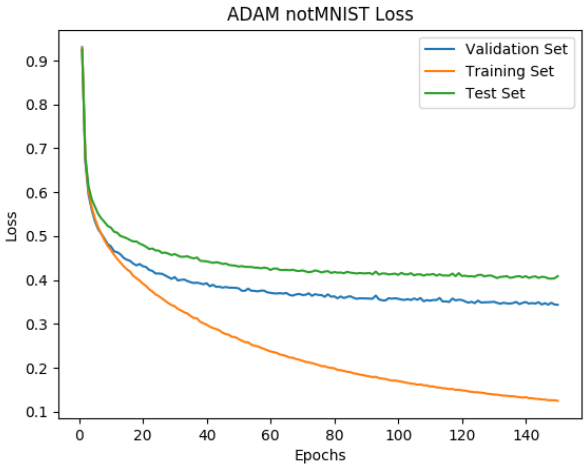


Figure 2.4.1b

**With Dropout:**

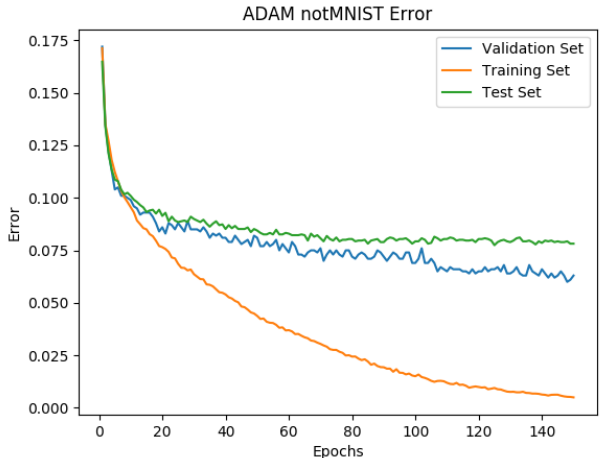


Figure 2.4.1c

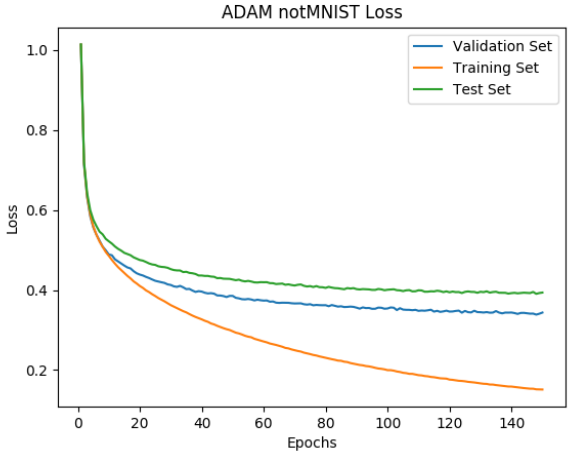


Figure 2.4.1d

The validation set classification error rate is 6%, training set classification error rate is 0.1% and test set classification error is 8.1% for training without dropout. With dropout, the validation error is 6%, training error is 0.5% and the test error is 7.8%. The classification error with dropout is higher for the test set because dropout encourages independence among hidden units by 'dropping' neurons during training. This prevents collaborations between neurons and prevents overfitting. As such, the weights are not able to overfit on the neural

net, which means that though it may not perform as well on the training set, it is better able to generalize to other data. This can be seen as the classification error with dropout is less than the classification error without dropout for the test set.

### 2.4.2 Visualization

**No Dropout:**

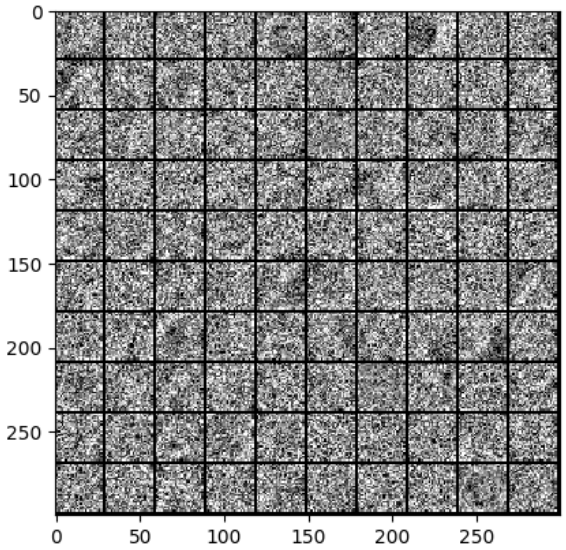


Figure 2.4.2a: Visualization of the Incoming Weights of 100 Neurons at 25% of the Early Stopping Point Without Dropout

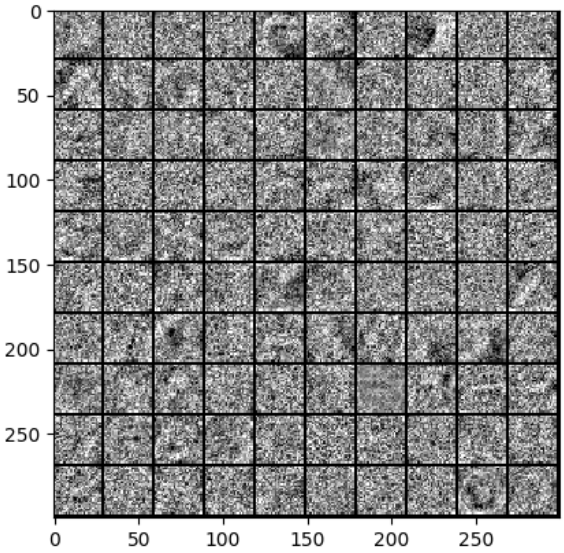


Figure 2.4.2b: Visualization of the Incoming Weights of 100 Neurons at 50% of the Early Stopping Point Without Dropout

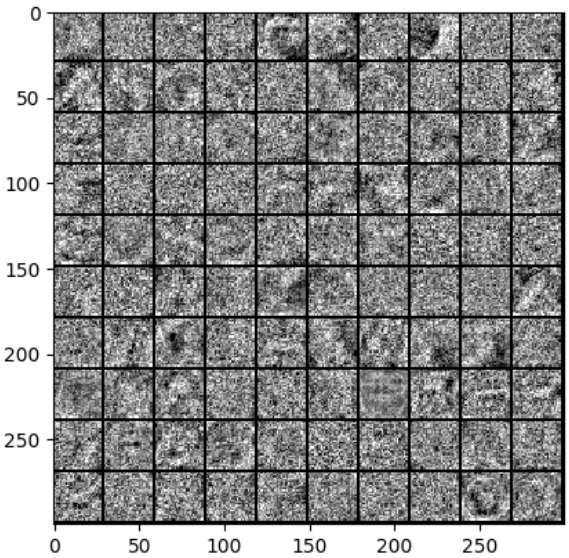


Figure 2.4.2c: Visualization of the Incoming Weights of 100 Neurons at 75% of the Early Stopping Point Without Dropout

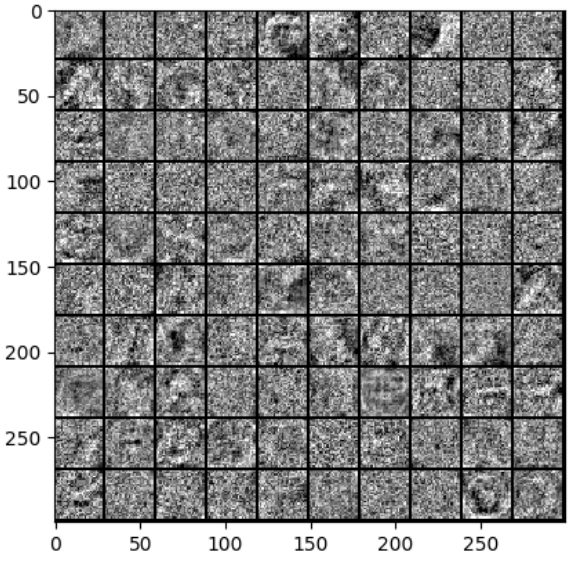


Figure 2.4.2d: Visualization of the Incoming Weights of 100 Neurons at 100% of the Early Stopping Point Without Dropout

**With Dropout:**

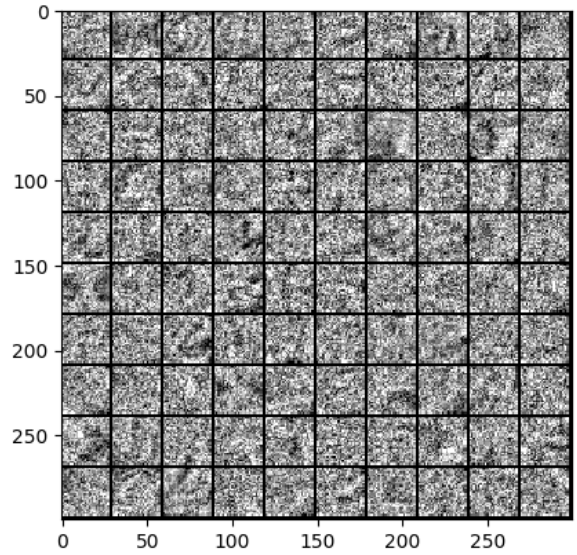


Figure 2.4.2e: Visualization of the Incoming Weights of 100 Neurons at 25% of the Early Stopping Point With Dropout

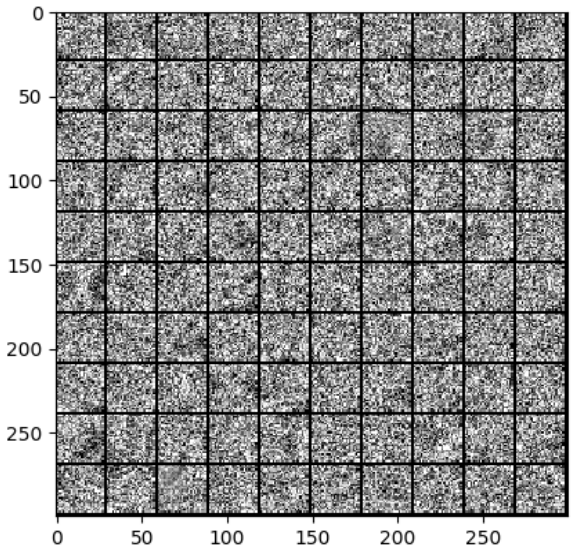


Figure 2.4.2f: Visualization of the Incoming Weights of 100 Neurons at 50% of the Early Stopping Point With Dropout

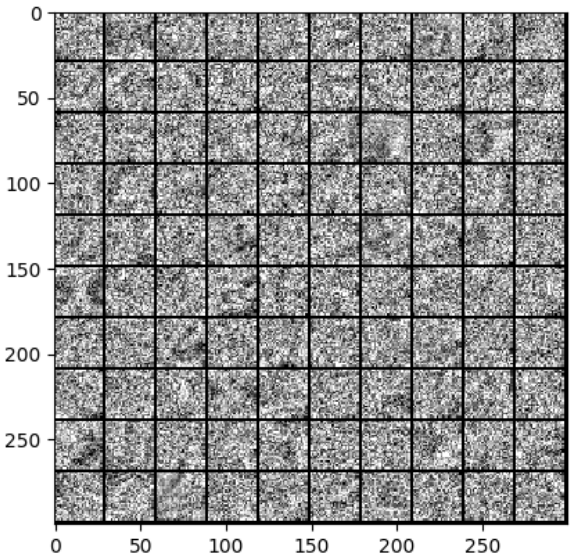


Figure 2.4.2g: Visualization of the Incoming Weights of 100 Neurons at 75% of the Early Stopping Point With Dropout

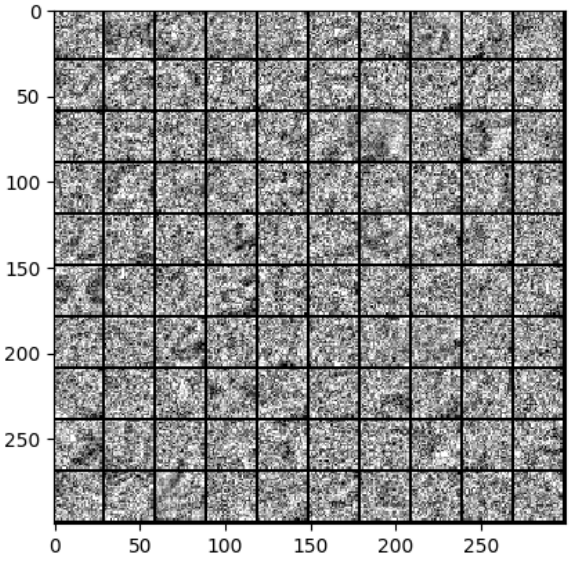


Figure 2.4.2h: Visualization of the Incoming Weights of 100 Neurons at 100% of the Early Stopping Point With Dropout

In the images above, each square is a visualization of the incoming weights to one neuron. As the training gets closer to the stopping point, the features visualized become more and more pronounced. At 25% of the early stopping point, there are very minimal distinctions between the neuron weights. As more epochs pass, the contrast between weights going into a neuron become more clear and some patterns begin to emerge. The visualization of without dropout clearly shows this as features become more prevalent.

There are fewer visually distinct features in the images of the weights trained with dropout. This observation results from the fact that with dropout enabled, each neuron needs to act more independently, which causes it to become a more generalized encapsulation of all input features. As a result, the corresponding weights appear less visually distinct. On the other hand, neurons that are trained without dropout are specialized to certain input features, resulting in their visually distinct shapes.

### 2.5.1 Random Search

**Trial A)** Learning rate: 0.001605 Hidden layers: 2 Hidden units: [187, 152] Weight decay: 0.0004935 Dropout: True

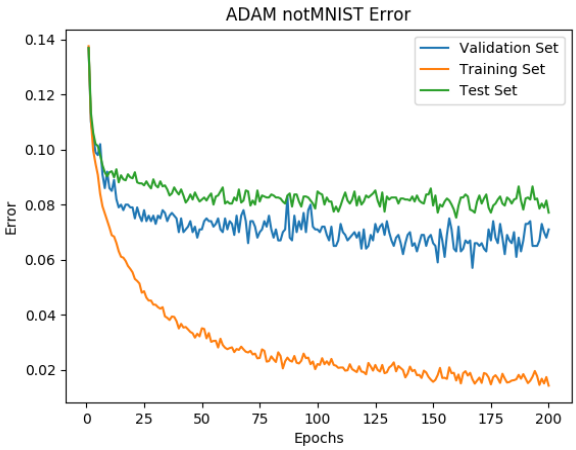


Figure 2.5.1a

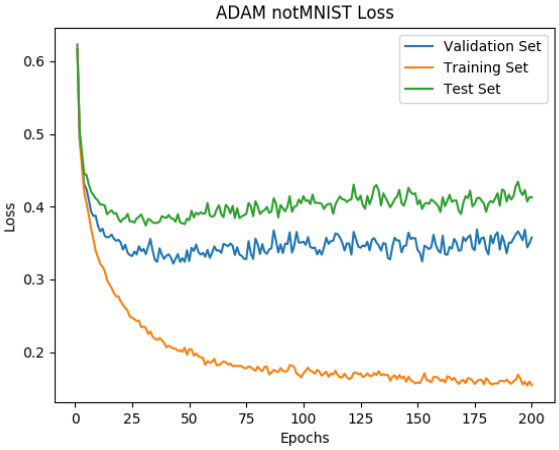


Figure 2.5.1b

**Trial B)** Learning rate: 0.003438 Hidden layers: 5 Hidden units: [373, 277, 344, 223, 434] Weight decay: 0.001277 Dropout: False

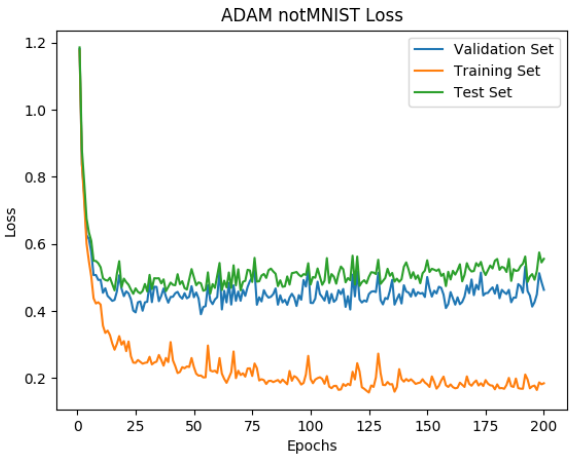


Figure 2.5.1c

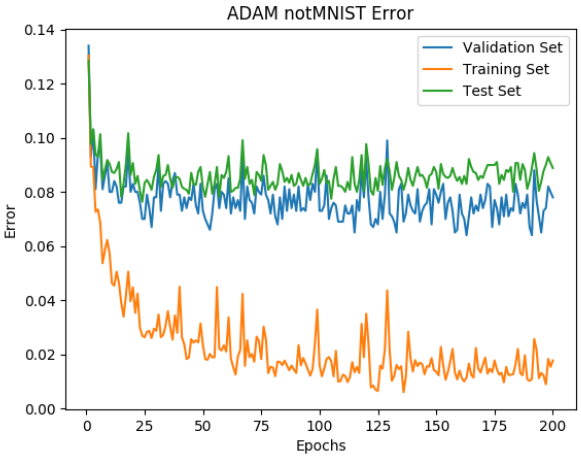


Figure 2.5.1d

**Trial C)** Learning rate: 0.00680 Hidden layers: 5 Hidden units: [405, 321, 485, 421, 137] Weight decay: 0.0.000159  Dropout: False

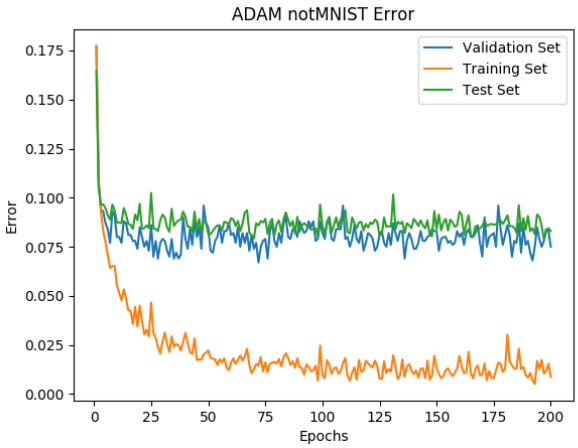


Figure 2.5.1e

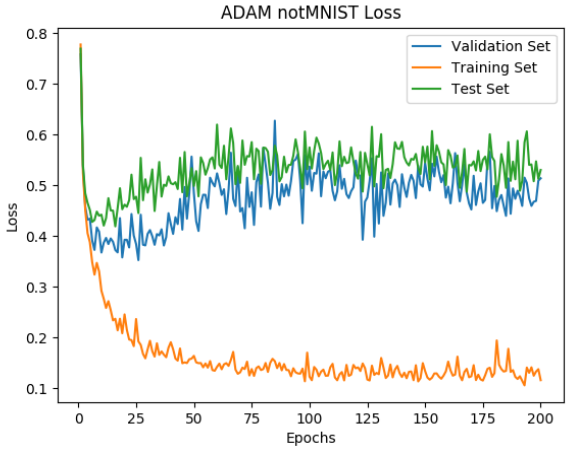


Figure 2.5.1f

**Trial D)** Learning rate: 0.000700 Hidden layers: 3 Hidden units: [384, 277, 411] Weight decay: 0.001286  Dropout: False

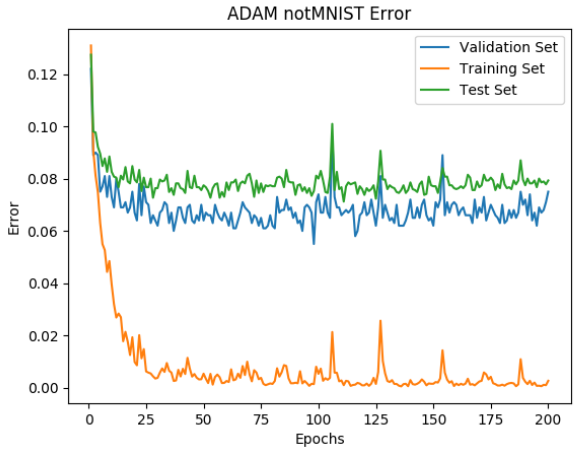


Figure 2.5.1g

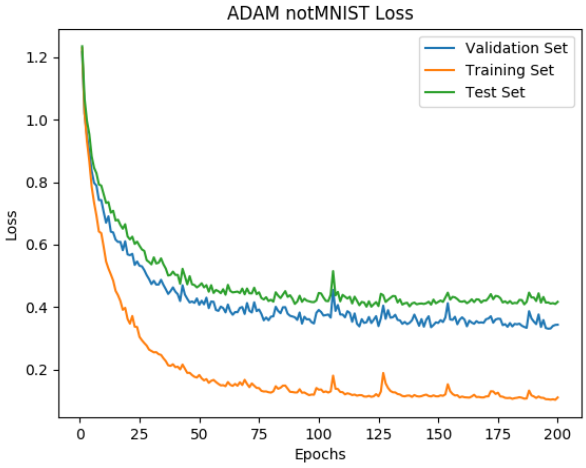


Figure 2.5.1h

**Trial E)** Learning rate: 0.000971 Hidden layers: 2 Hidden units: [426, 365] Weight decay: 0.001675  Dropout: False

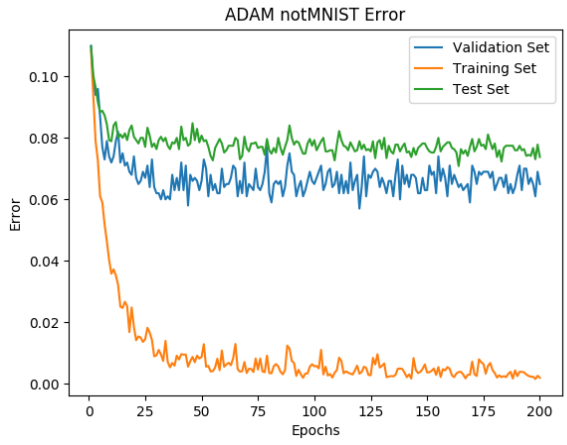


Figure 2.5.1i

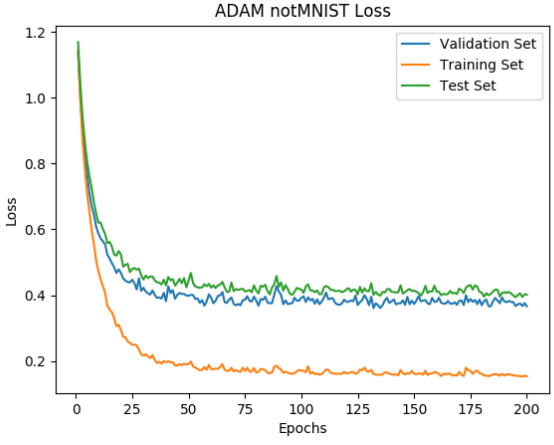


Figure 2.5.1j

Table 2.5.1 - Summary of Five Exhaustive Search Results

| Trial | Validation Error | Test Error |
|---|---|---|
| A | 0.0701 | 0.0794 |
| B | 0.0769 | 0.0882 |
| C | 0.0751 | 0.0780 |
| D | 0.0743 | 0.0803 |
| E | 0.0687 | 0.0754 |

### 2.5.2 Exchange Ideas Among the Groups

From Piazza, the set of hyperparameters producing the best test is:

**Trial F)** Learning rate: 0.0007051  Hidden layers: 5 Hidden units: [236, 438, 326, 298, 496] Weight decay: 0.0017 Dropout: False

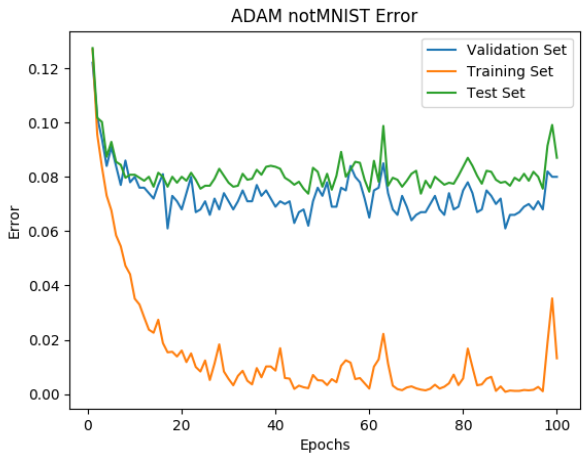


Figure 2.5.2a

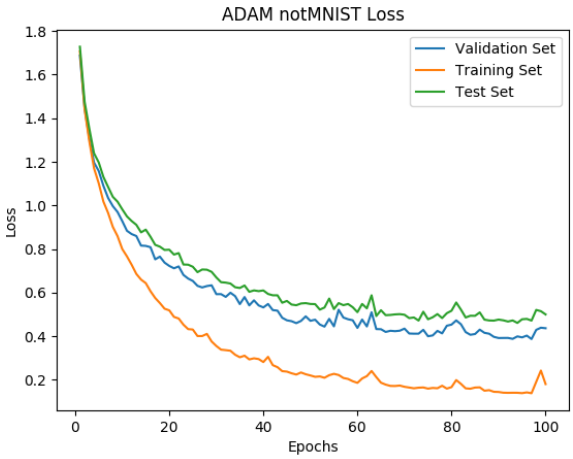


Figure 2.5.2b

Table 2.5.2 - Results of Best Hyperparameters Up-to-Date

| Trial | Validation Error | Test Error |
|---|---|---|
| F | 0.0610 | 0.0738 |

Due to the collaboration among the classmates of ECE521, the following results reported by an anonymous user on Piazza resulted in the best hyperparameters that were available at the time of writing this report. These results illustrate the importance of hyperparameter tuning for general classification machine learning models, as well as highlight the difficulty in finding optimal hyperparameters. However, with teamwork, even this mathematically intractable problem can be solved a little bit easier.