# Using Neural Networks To Solve Phone Number Recognition Issue

By: Deemal Patel
Northwestern University
April 18, 2021

## Abstract

Our company is facing the issue of not being able to recognize customer handwriting when they write their phone numbers down to participate in our survey. Missing out on about 15% of customer phone numbers, we are losing potential reviews on improving customer satisfaction. Using a neural network model we are able solve our issue of not being able to recognize numbers with potentially missing out on only 2% of those 15%. The model contains 128 nodes in a hidden layer, testing out various combinations of layers and optimizers in order to determine what the best model for use will be. Based on the results of all the models we have achieved an accuracy of about 99% for training, testing, and validation data. Based on this score there is a model that is ready to be implemented and tested out on the customers phone numbers.

## Introduction

In the 1990s the United States Postal Services was suffering a problem of not being able to decipher zip codes for 10% of the mail that would come through. According to the USPS, they process and deliver about 430 million mail pieces each day (United States Postal Service, 2021). These numbers are for today's day but had the this amount been in the 90s, that would be about 43 million mail pieces in which the zip codes cannot be deciphered, and therefore would not be delivered. The USPS had reached out to a consulting group in order to develop a neural network model that solved their issue of not being able to detect the zip codes. The consulting groups accuracy score came out to around 99% which is a great accuracy score meaning they only missed out on about 1% of the mail which is a lot better than missing out on 10%.

Due to the success of the USPS of getting such a great accuracy score, management has asked me to look in to developing a model that will be able to read the customers call back numbers from what they have written down. We have been receiving many responses in our call back box but about 15% of the customers that wrote call back numbers that are illegible. Since we have been struggling to read the phone numbers customers wrote, we have been unable to give them a call in order to participate in our survey. Missing feedback from our customers could prevent us from improving our company by not knowing how we can increase customer satisfaction.

## Literature Review

Reading the images of handwritten numbers classifies as a computer vision problem. There have been many methods that people have done that tackled this issue as computer vision is a common task that many companies use to solve their business needs. One method that I have decided to include in my research, is the use of Random Forests. Random Forests is a similar method of decision trees but in this case instead of using one decision tree, we are using multiple decision trees. The random forest build multiple decision trees and merges them all together to get a more accurate and stable prediction. Because of random forests capabilities and power that it contains I have decided to include it as one of the classification methods I wanted to research for our digit recognition.

The most commonly used model for deep learning is the use of neural networks. I have made the use of a single layered neural network my main focus and research for developing a digit recognizer. What’s great about neural networks is that by increasing the number of training examples, the network can learn more about the handwriting and therefore the accuracy improves. By having a large sample size creating an accurate neural network will not be an issue showing that it is the method of choice for creating the recognizer.

Usually combined with neural network classifiers is the use of Principal Component Analysis, or PCA. PCA is a dimensionality reduction method to reduce the number of features in order to reduce the computation requirements causing less load on the system and faster training time. Vineet Singh from University of the South Pacific performed a similar test of using PCA on a single layer neural network and noticed and 80% training time reduction on his model (Singh 2014).

## Methods

In order to thoroughly conduct my research on developing the proper model I decided to create multiple models in order to compare and contrast the results so that I can use the best model to recognize the hand written digits. The models used for my research used a multilayer perceptron, or MLP. MLP uses forward feed meaning the data will be passed on to the next layer for training, and models train using backpropagation making neural networks a great choice for creating our digit recognizer. Each model is a multilayer perceptron, with one hidden layer model for seven out of the ten models. Each mode has a variety of neurons in order to determine which number of nodes will provide me with the best accuracy results. For all my single layer models I designed it to use 1, 2, 5, 10, 128, 255, and 512 nodes, respectively. I also developed a 2 layer, 3 layer, and 5 layer model, each consisting of 128 nodes for each of the layers. Each model uses a ReLU activation which is the most commonly used activation function in neural networks (Liu 2017).

One of the main tools I used in this research is TensorFlow, a free open-source software that is used for machine learning. Built in to TensorFlow is the Keras API containing tools to build our model. In order to create our layers for our model Keras comes with a Dense class. The Dense class is how we will be connecting our layers of input, hidden, and output layers. It takes in the input shape, which is the columns that we use to identify our label, or class. It also takes in the activations we are going to use and in our case it is ReLU. Finally it takes in the number of nodes we want to use to train the model. Our output layer is set up to use a SoftMax activation which his essentially the go-to activation when working with multiclass classification problems. I decided to use 128 nodes for each of my layers because that will be well enough to help collect the data and train our model accordingly. Keras also has a datasets package containing our training data. The training data that I used for my models is the MNIST training set which is a widely known and popular data set in order to recognize handwritten digits.

Another package that I used is the seaborn package and matplotlib.pyplot package. These package allow me to generate visuals in order to take a look at how the numbers were properly classified from some of my models. Finally, the last package that I used is the sci-kit learn package. Implementing the sci-kit learn package allows me to use some of the methods mentioned in the literature review like PCA and Random Forest Classifiers. Sci-kit learn also contains a metrics package that allowed me to develop a confusion matrix to check the true positives, false positive, etc. The other function from the metrics package I used is accuracy score which allowed me to see how well my model worked on test data, essentially new data that my model hasn't seen. From that I can tell whether the model is overfitting or underfitting.

In order to use my data properly with my model I need to get some insights as to what my data contains. Each column of the data is a pixel value of the image ranging from 0 to 255. With this wide scale it makes it a little difficult for our model to learn so I needed to prepare my data in such a way that all the values for each column are in a reasonable range such as from 0 to 1. With the use of seaborn, I was able to explore my data through visuals to see how the handwritten images looked like and essentially made sure that data's with the label of 6 actually looked like 6, as shown in Appendix 2.  After complete finish I plan on revising the model every month or so in order to make sure the model stays accurate. The concept of machine learning models becoming less accurate over time is called drift. The accuracy changes over time and causes problems for your model as is becomes less and less accurate as time goes on (Gonfalonieri 2019).

## Results

The first two experiments I conducted were to just see how the neural network will learn and the rate at which it picked up accuracies. The first true research experiment I ran was a single hidden layer model with 128 nodes using the Adam optimizer. The results shown in Appendix 1 show that I got a testing accuracy of about 99.96%, a test accuracy of 97.62%, and a validation of 98.12%. This is a very good performance but you can tell that there is some overfitting going on in the training because the test accuracy is lower by 2.3% meaning we can miss out on some important digits. Taking a look at the confusion matrix in Appendix 1, we can see that there were 24 errors classifying 4 which incorrectly got classified as 9. The second experiment is similar to the first experiment but I used an AdaGrad optimizer. The results gave 95.25% training, 95.09% test, and 94.26% validation. With this experiment there is no overfitting which is good but the accuracy is lower than what we want to aim for. Looking at the confusion matrix in Appendix 2, 3 and 7 contained the most errors with incorrect classifications with about 50 errors in each one. The third experiment is just like the first two, but I am using the rmsprop optimizer. This gave me a 99.90 training, 97.63% test, and 97.80% validation accuracies. This is a stronger model than AdaGrad but just like Adam, there is overfitting occurring in our model. The confusion matrix for this one is a little interesting, the errors were a little spread out with the most number of errors being 10 which we see in 4 and 7. This might make it a little tougher to fix the model because of the variance of errors. Had most of the errors focused on a single digit it would be an easier fix.

In the next experiment I decided to perform dimensionality reduction with PCA by reducing the number of columns to 95% variance coming out to 154 columns to use. What PCA has done essentially is reduce the dimensionality of our data from 784 to 154 so that I can make a simpler predictive model that may perform better when making predictions. This result gave me 100% training, 97.54% test, and 97.88% validation. We were able to get an incredible training score of 100% and the test and validation data are not too far off with being almost 98% for each of them, making this model a contender for the best digit recognition. Then I decided to use a Random Forest Classifier in order to determine which are the top 70 features for classifying the images. Using those top 70 features I ran the model with rmsprop and 128 nodes, the results came out to 96.47% train, 96.34% test, and 94.14% validation. It seems that reducing the model to only it's top 70 features didn’t provide us with the accuracy we need to so maybe if we increase the number of best features we can attain a better accuracy score.

I then decided to run the experiments with multiple hidden layers to see if increasing the layers would play an affect on the accuracy scores. I ran a 128 node 2 layer with rmsprop and the results came to 99.76% train, 97.88% test, and 97.88% validation. This is a good score for both the training and testing but it seems that are is some overfitting occurring in the model. The next experiment had 3 layers of 128 nodes with Adam and the results came to 99.44% train, 97.72% test, and 97.92% validation. There wasn’t much of a change in this model as we can see the accuracy scores are essentially the same. The final multi hidden layer I used 5 layers with a 128 nodes each, using the SGD optimizer setting the learning rate to 0.0001 and momentum to 0.9. The results came out to 99.78% train, 99.04% test, and 98.80% validation. These are great numbers because each of the 3 are within or very close to 99% accuracy, making this model the top choice of use for our digit recognition.

The last two experiments I went back to single layer and used more nodes this time. This time I wanted to see if increasing the number of nodes with a single layer can help improve our accuracy scores. I used 256 nodes with rmsprop which came out to 98.42% train, 97.98% test, and 98.42% validation. My expectations were that this score could come close to 99% but it seems none of the 3 did so, but we can see that are isn't any overfitting going on which is a g. Finally the last experiment was similar to the previous but with 512 nodes with 99.90% train, 98.17% test, and 98.20% validation. This mode came out with some good accuracy scores as well. Looking at the confusion matrix for this model we can see that the errors are minor and spread out across the digits. This could mean that it may prove to be a little difficult on fixing the model since the numbers are so spread out rather than grouped in to a specific digit.

## Conclusion

The best model out of all the experiments came out to be the 5 layer model using 128 nodes and an SGD optimizer with the learning rate and momentum manipulated. I recommend this model to management because the model did not overfit nor underfit and the accuracy scores are roughly around 99%. This means that out of the 15% phone numbers we can't recognize we will only miss out on 1% of that 15%. This will allow us to contact more customers and get their survey responses on how we can improve customer satisfaction which will then allow us to improve the company. I recommend that the model be tweaked every quarter because we will have more data and prevent drift.

## References

Gonfalonieri, Alexandre. “Why Machine Learning Models Degrade In Production.” Medium. Towards Data Science, July 25, 2019. https://towardsdatascience.com/why-machine-learning-models-degrade-in-production-d0f2108e9214.

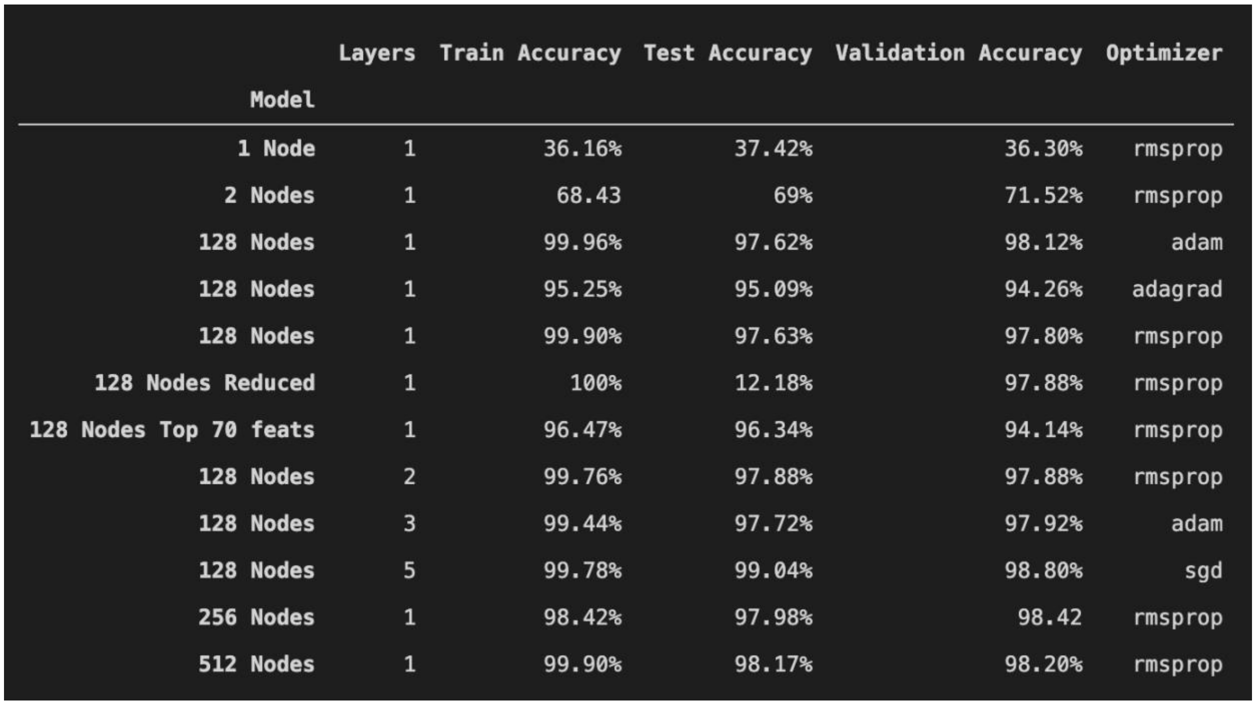
Liu, Danqing. “A Practical Guide to ReLU.” Medium. Medium, November 30, 2017. https://medium.com/@danqing/a-practical-guide-to-relu-b83ca804f1f7.

Singh, Vineet, and Sunil Pranit Lal. “Digit Recognition Using Single Layer Neural Network with Principal Component Analysis.” IEEE Xplore, November 4, 2014. https://ieeexplore.ieee.org/abstract/document/7053842.

United States Postal Service - March 9. Postal Facts - U.S. Postal Service, March 9, 2021. https://facts.usps.com/#:~:text=The%20Postal%20Service%20processes%20and%20delivers%20429.9%20million%20mail%20pieces%20each%20day.

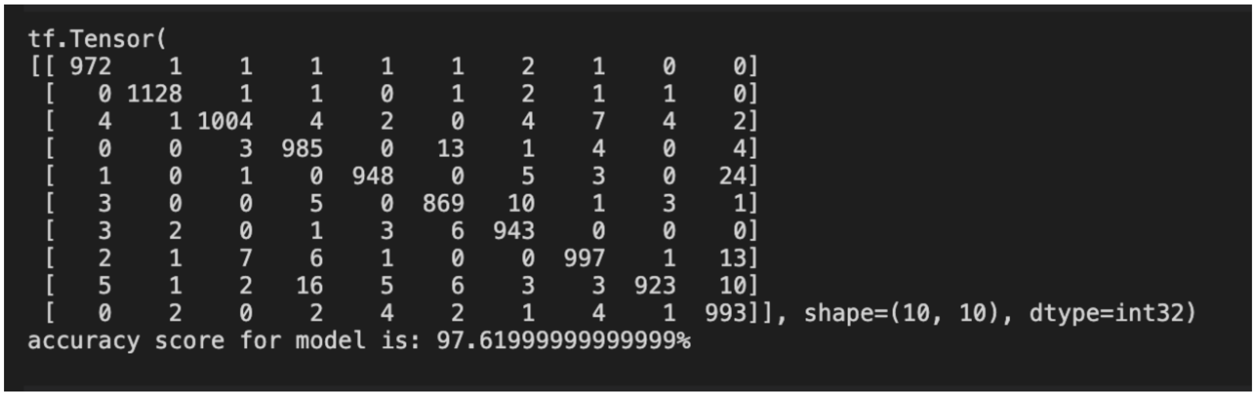
APPENDIX 1:

Summary Table



| Model | Layers | Train Accuracy | Test Accuracy | Validation Accuracy | Optimizer |
|---|---|---|---|---|---|
| 1 Node | 1 | 36.16% | 37.42% | 36.30% | rmsprop |
| 2 Nodes | 1 | 68.43 | 69% | 71.52% | rmsprop |
| 128 Nodes | 1 | 99.96% | 97.62% | 98.12% | adam |
| 128 Nodes | 1 | 95.25% | 95.09% | 94.26% | adagrad |
| 128 Nodes | 1 | 99.90% | 97.63% | 97.80% | rmsprop |
| 128 Nodes Reduced | 1 | 100% | 12.18% | 97.88% | rmsprop |
| 128 Nodes Top 70 feats | 1 | 96.47% | 96.34% | 94.14% | rmsprop |
| 128 Nodes | 2 | 99.76% | 97.88% | 97.88% | rmsprop |
| 128 Nodes | 3 | 99.44% | 97.72% | 97.92% | adam |
| 128 Nodes | 5 | 99.78% | 99.04% | 98.80% | sgd |
| 256 Nodes | 1 | 98.42% | 97.98% | 98.42 | rmsprop |
| 512 Nodes | 1 | 99.90% | 98.17% | 98.20% | rmsprop |

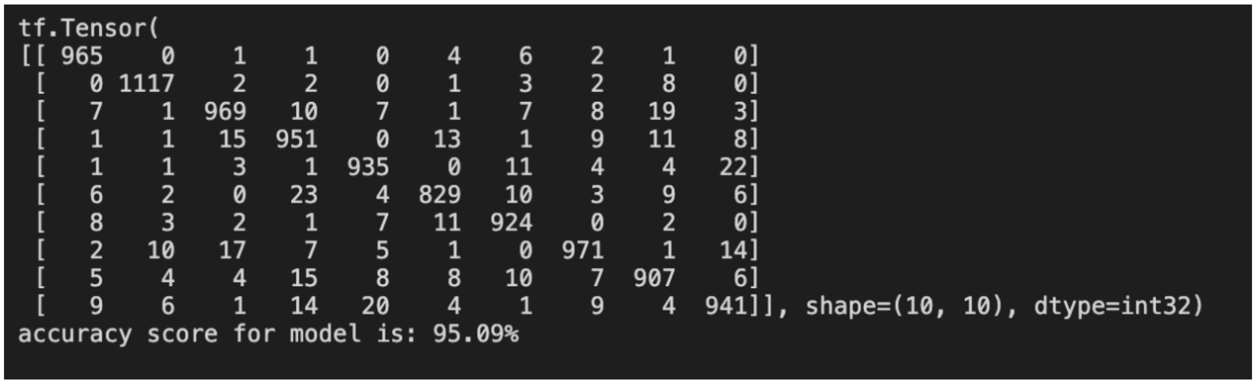
128 Node, single layer, adam optimizer



```
tf.Tensor(
[[ 972    1    1    1    1    1    2    1    0    0]
 [   0 1128    1    1    0    1    2    1    1    0]
 [   4    1 1004    4    2    0    4    7    4    2]
 [   0    0    3  985    0   13    1    4    0    4]
 [   1    0    1    0  948    0    5    3    0   24]
 [   3    0    0    5    0  869   10    1    3    1]
 [   3    2    0    1    3    6  943    0    0    0]
 [   2    1    7    6    1    0    0  997    1   13]
 [   5    1    2   16    5    6    3    3  923   10]
 [   0    2    0    2    4    2    1    4    1  993]], shape=(10, 10), dtype=int32)
accuracy score for model is: 97.61999999999999%
```

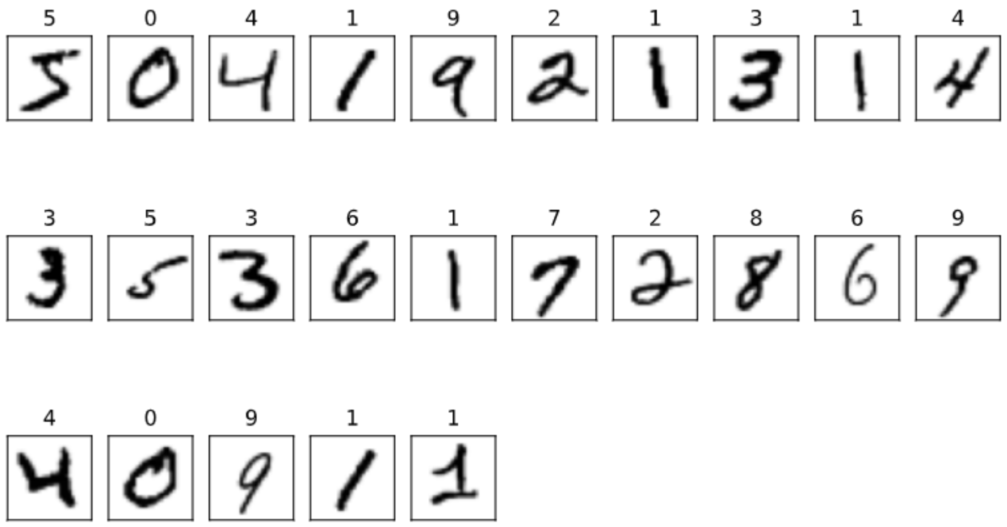
APPENDIX 2

128 Node, single layer, AdaGrad Optimizer



```
tf.Tensor(
[[ 965    0    1    1    0    4    6    2    1    0]
 [   0 1117    2    2    0    1    3    2    8    0]
 [   7    1  969   10    7    1    7    8   19    3]
 [   1    1   15  951    0   13    1    9   11    8]
 [   1    1    3    1  935    0   11    4    4   22]
 [   6    2    0   23    4  829   10    3    9    6]
 [   8    3    2    1    7   11  924    0    2    0]
 [   2   10   17    7    5    1    0  971    1   14]
 [   5    4    4   15    8    8   10    7  907    6]
 [   9    6    1   14   20    4    1    9    4  941]], shape=(10, 10), dtype=int32)
accuracy score for model is: 95.09%
```

Plot of our data, ensuring values match the labels


5
0
4
1
9
2
1
3
1
4
3
5
3
6
1
7
2
8
6
9
4
0
9
1
1