Simple Neural Networks - Architecture and Design Experimentation

Parviz Ali
MSDS 458: Assignment 1
Fall 2021

Abstract

Optical character recognition, such as handwritten digits recognition, is a complex task central to a variety of software applications. It is being widely used by machine learning and computer vision systems in practical applications such as bank check number and postal zip code recognition. We experimented with a multi-layer fully connected neural network with one hidden layer for handwritten digits recognition. The testing has been conducted from publicly available MNIST handwritten database. From the MNIST database, we extracted 60,000 digits images for training and 10,000 digits images for performing the test. All these black and white digits are size normalized, and centered in a fixed-size image where the center of the intensity lies at the center of the image with 28 × 28 pixels. The dimensionality of each image sample vector is 28 * 28 = 784, where each element is binary. The purpose of this paper is to define how neural networks are used to resolve a problem of MNIST handwritten digit classification and to understand how the neurons/nodes in a simple single-hidden layer network have learned to represent features within the input data. We designed a neural network model and then implemented it with various optimization strategies to solve the classification problem. Our multi-layer artificial neural network has an accuracy of 97.20% with test performance.

Keywords— Computer vision, Handwritten digits recognition, Artificial neural network.

Introduction

Handwritten digits recognition continues to be important in our everyday lives. Its practical applications in our daily life are immensely broad. Automatic processing of bank checks and postal address are widely used applications of handwritten digit recognition. It can solve more complex problems which makes our everyday job easier. Reading printed or handwritten

documents and converting them to digital media is very crucial and time-consuming task. Executing a computerized system to do certain kinds of duties is a very complex and challenging matter. As we move on to broader AI applications such as Natural Language Processing (NLP), integration and understanding of digit recognition technology is becoming crucial to using it in various software applications. Office automation, e-document management and many more solutions rely on digit recognition technology. Pattern identification is the fundamental ingredient of a computer vision and artificial intelligence-based system. Understanding the engine behind this technology is the key to developing and optimizing applications. Using various experiments, we intend to derive insight on how the neurons/nodes in a simple single-hidden layer network have learned to represent features within the input data.

Literature review

Optical character recognition is an important problem to understand. Handwritten digit recognition is used as a test case for theories of pattern recognition and machine learning algorithms. Several publicly available standard databases are used to promote research of machine learning and pattern recognition. These repositories contain thousands of preprocessed handwritten digits that have been segmented and normalized for the researchers to be able to compare recognition results of their techniques (Deng 2012).

This paper uses MNIST handwritten digit database on Artificial Neural Network (ANN). MNIST handwritten digit database images were created in 1998 as a combination of two of NIST's databases: Special Database 1 and Special Database 3. Special Database 1 and Special Database 3 consist of digits written by high school students and employees of the United States Census Bureau, respectively. They can be taken from the page of Yann LeCun (Yann.lecun.com,

n.d.). It has become a standard for fast-testing theories of pattern recognition and machine learning algorithms.

Artificial neural networks (ANNs) are inspired by biological central systems in brains and have been utilized in a variety of applications ranging from modeling, classification, pattern recognition, and multivariate data analysis (Basheer and Hajmeer, 2000). Back-propagation (BP) neural network is widely applied in ANN application. Back-propagation algorithm is put forward by D. Rumelhart and J. McCelland in 1986 and they presented the neural networks using BP methods are called backpropagation neural networks (BPNN) (Rumelhart, Hinton, and McClelland, 1986). Back- propagation neural networks are supervised multi-layer feed forward neural networks, commonly consisting of an input layer, an output layer, and one or several hidden layers. Handwriting recognition is one of the most favorite topics among researchers (Memon et al. 2020) because every individual in this world has their style of writing. Similar experiments conducted (Pandey et al. 2020) have resulted in 92% accuracy. Niu and C. Y. Suen (Niu and Suen 2012) introduced a hybrid method for recognition of handwritten digits. Researchers used convolution neural network (CNN) to extract the image features and fed to a hybrid classifier to classification. Their hybrid classifier contained CNN and SVM and achieved 94.40% of classification accuracy

Methods

We implemented a digital neural network (DNN) and trained it to recognize handwritten digits from 0 to 9. A node or a parameter in a neural network can be understood as a neuron in the brain. Each node is connected to other nodes through weights which are adjusted in the machine learning process during training. A value is calculated for each node based on values and ways of previous nodes. This process is called forward propagation (see figure 1). The final

output of the network is associated with the target output, then weights are calibrated to minimize a transgression function describing whether the network guessed correctly. This process is called back propagation. Back-propagation neural networks are supervised multi-layer feed forward neural networks, commonly consisting of an input layer, an output layer, and one or several hidden layers.

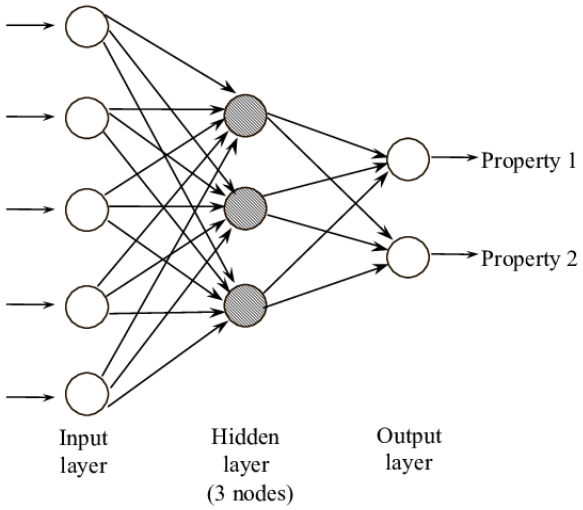


Figure 1. Examples of Input, Hidden and Output Layers with Nodes

Here we build a multilayer perceptron (MLP) neural network with only one hidden layer. Input layer contains 28 * 28 = 784 neurons, representing the features. Hidden layer contains from one to multiple neurons in various experimentations. We used ReLU as activation function for hidden layer and Softmax for output layer. The output layer contains 10 neurons representing the digits from 0 to 9. We used sparse_categorical_crossentropy error function as network loss function. This error function is more suitable for classification problems than mean squared error (MSE) function. We chose rmsprop as our optimizer. After defining the neural network, we train the model by batch. In order to set the appropriate parameters such as batch_size, number_of_epochs and learning rate, we conducted several tests for evaluation.

As mentioned above, in this paper we use MNIST handwritten digit database in which the handwritten digits have been preprocessed including segmentation and normalization. There are 60,000 training images and 10,000 test images, and the dimensionality of each image sample vector is 28 * 28 = 784, where each element is binary.

We used Keras, a part of TensorFlow 2.x, and imported Keras from Tensorflow and used tenserflow.keras.xxx to import all other Keras packages. Google Collaboratory with GPU accelerator was used to take advantage of this publicly available platform and speed.

In order to determine the performance of the neural network and predictions and to report the results produced by our neural network, we choose to calculate the training accuracy and testing accuracy. Using matplotlib.pyplot we plotted historical loss from “all losses” during network learning as a visualized representation. We plotted sample images with their labels to examine our dataset (see figure 2). Data distribution is shown in figure 3.

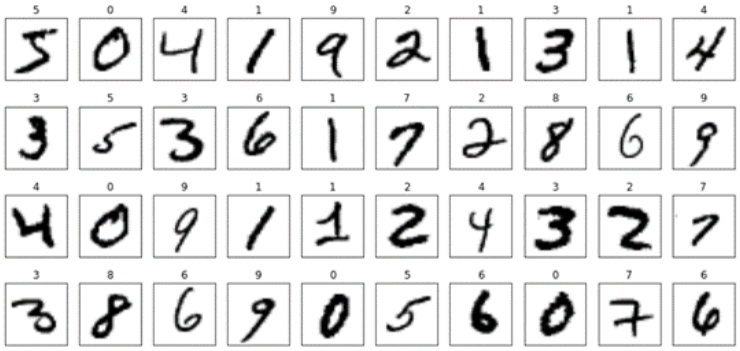


Figure 2. Sample images with label in dataset

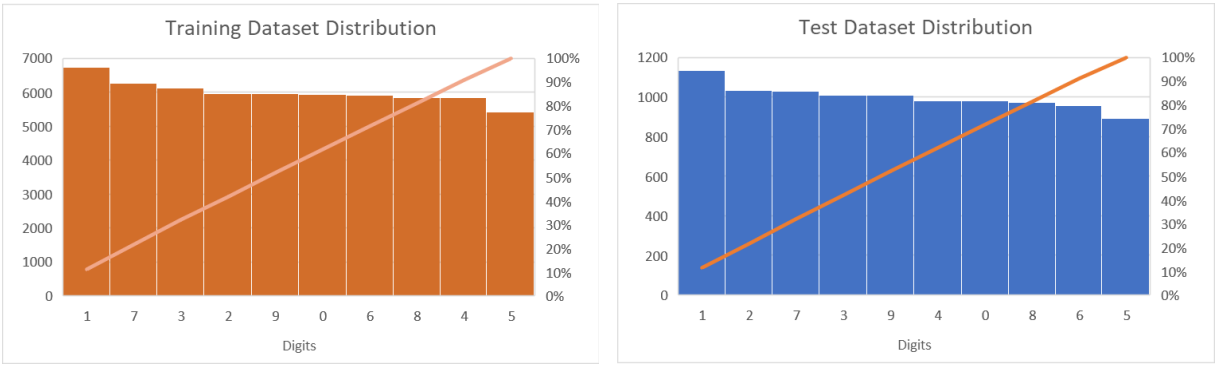


Figure 3. Exploring the training and test dataset distribution

Results

We conducted total 17 experiments (see figure 4). We notice that as the number of neurons is increased from 1 up to 512, test accuracy of the model increased. We found our best

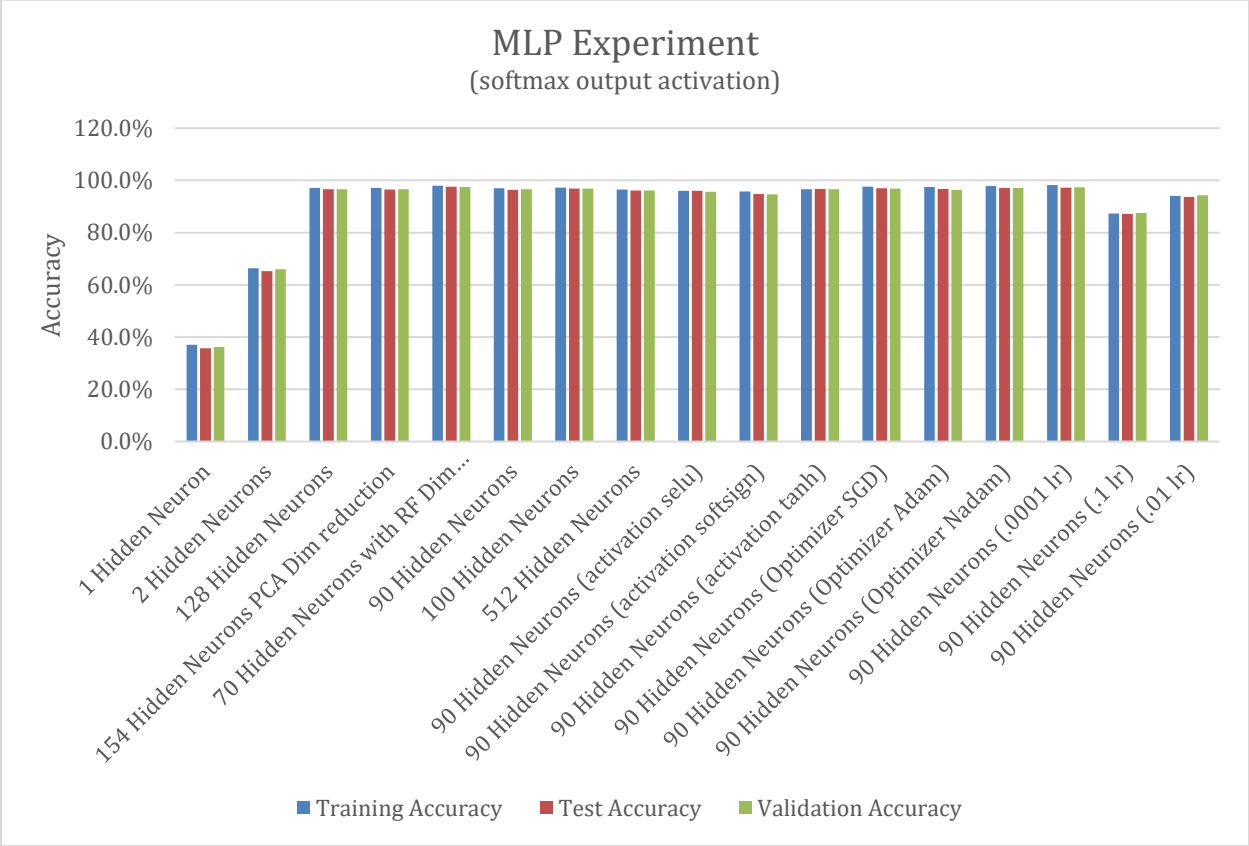


Figure 4. Model performance with various hyperparameter setting

model performance was with 90 neurons with 97.1% test accuracy (see table 1). As expected,

Table 1. MLP Experiment results

MLP Experiment (softmax output)	Training Accuracy	Test Accuracy	Validation Accuracy	Epoch	Train loss	Val loss	Activation (HLayer)	Optimizer	Learning rate
1 Hidden Neuron	37.0%	35.7%	36.2%	6	1.5808	1.562	relu	rmsprop	0.001
2 Hidden Neurons	66.3%	65.3%	66.0%	17	1.0571	1.0562	relu	rmsprop	0.001
128 Hidden Neurons	97.0%	96.6%	96.6%	8	0.1517	0.1631	relu	rmsprop	0.001
154 Hidden Neurons PCA Dim reduction	97.1%	96.5%	96.6%	9	0.1515	0.1706	relu	rmsprop	0.001
70 Hidden Neurons with RF Dim reduction	97.9%	97.5%	97.4%	5	0.0748	0.0924	relu	rmsprop	0.001
90 Hidden Neurons	97.0%	96.4%	96.6%	8	0.1526	0.1698	relu	rmsprop	0.001
100 Hidden Neurons	97.2%	96.9%	96.8%	9	0.1484	0.1587	relu	rmsprop	0.001
512 Hidden Neurons	96.5%	96.2%	96.1%	6	0.1855	0.1942	relu	rmsprop	0.001
90 Hidden Neurons (activation selu)	96.0%	96.0%	95.7%	9	0.1938	0.2014	selu	rmsprop	0.001
90 Hidden Neurons (activation softsign)	95.7%	94.7%	94.6%	9	0.2007	0.2334	softsign	rmsprop	0.001
90 Hidden Neurons (activation tanh)	96.6%	96.7%	96.6%	12	0.1722	0.1786	tanh	rmsprop	0.001
90 Hidden Neurons (Optimizer SGD)	97.6%	97.0%	96.9%	36	0.1567	0.1752	relu	SGD	0.001
90 Hidden Neurons (Optimizer Adam)	97.5%	96.7%	96.3%	8	0.1481	0.1835	relu	Adam	0.001
90 Hidden Neurons (Optimizer Nadam)	97.9%	97.1%	97.1%	6	0.1342	0.1597	relu	Nadam	0.001
90 Hidden Neurons (.0001 lr)	98.3%	97.2%	97.3%	10	0.0802	0.1184	relu	rmsprop	0.0001
90 Hidden Neurons (.1 lr)	87.3%	87.2%	87.5%	4	0.6262	0.6144	relu	rmsprop	0.1
90 Hidden Neurons (.01 lr)	94.0%	93.7%	94.2%	5	0.311	0.3024	relu	rmsprop	0.01

a single neuron model only achieved 35.7% accuracy with high loss. It created a bottle neck. Adding a second neuron almost doubled the accuracy to 65.3%. The model performance plateaued after 90 neurons. In fact, after 100 neurons its performance slightly decreased. Looking at the best performing model's first 25 images (see figure 5) we notice that only one has been incorrectly identified. Training and validation curve starts to diverge after 8 epochs (see figure 6). Confusion matrix chart shows (see figure 7) the model misread certain graphics. Further examination of the digit graphics shows (see figure 8) the handwritten digits 9 and 4 were misclassified. The model could not differentiate

Figure 5. First 25 images identified by the model with 90 neurons

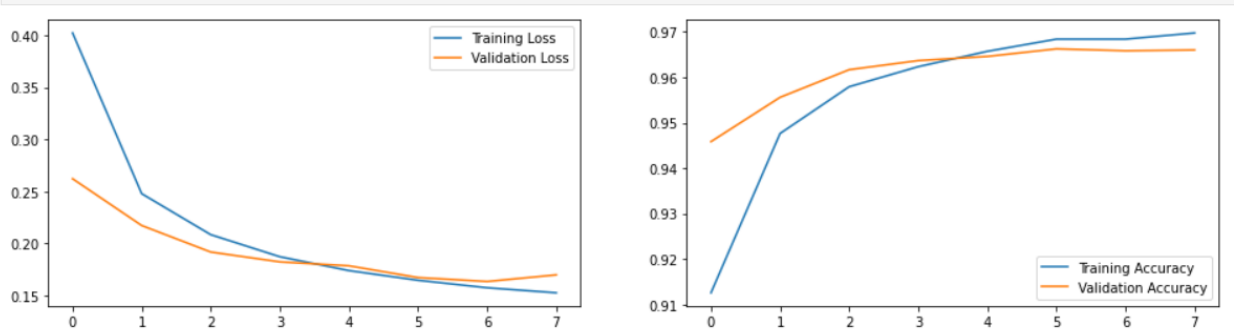


Figure 6. Performance matrix of model with 90 neurons

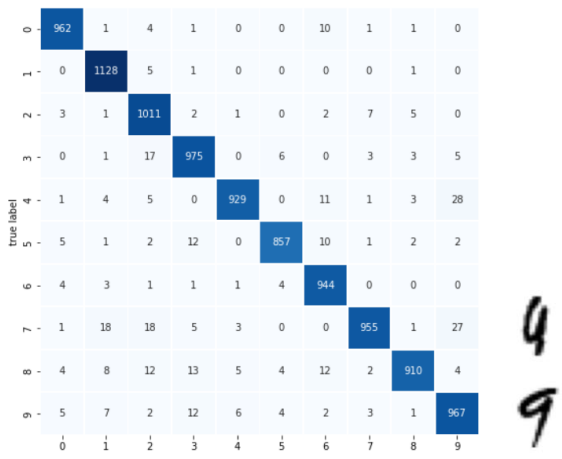


Figure 7. Confusion matrix

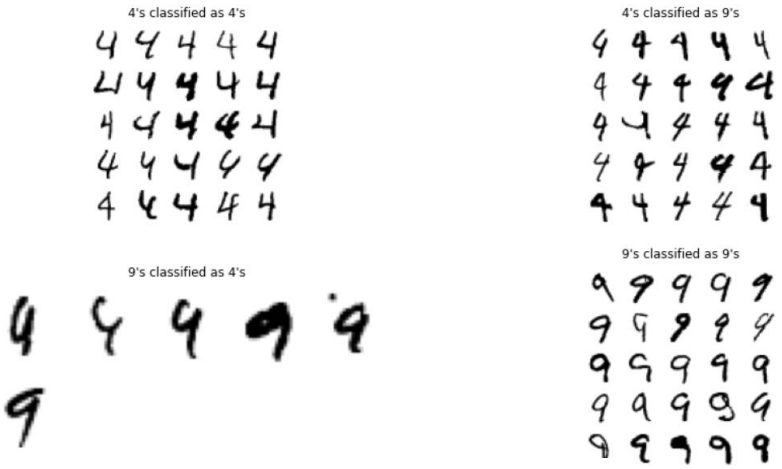


Figure 8. Images of misclassified digits

these images. We used principal component analysis (PCA) decomposition to reduce the (activation) features from 90 hidden nodes to 2. Two dimensional t-Distributed Stochastic Neighbor Embedding (t-SNE) plot (see figure 9) shows the model has been able to distinctly discriminate 10 digits from 90 features using 60,000 data points.

Figure 9. Two-dimensional t-SNE plot

Reducing dimensionality of the data with Random Forests Classifier (100 trees) we found the relative importance of the 784 features (pixels) in the training set. We produced a heat map (see figure 10) visual to investigate the relative importance of the features. Finally, we selectd the 70

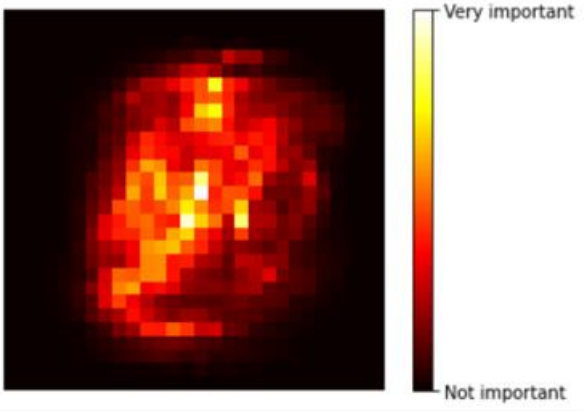


Figure 10. RF classifier heat map

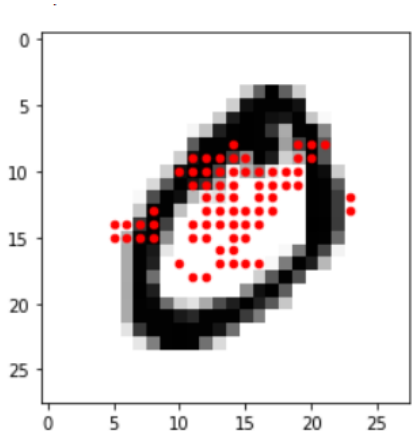


Figure 11. PC 70 features used for training NN

most important feature (pixels) from the training, validation and test images to test our 'best' model on. The ordered pairs were plotted as red circles on the second training image (see figure 11). The neural network was trained using these features (pixels).

With 154 neurons PCA dimension reduction the model accuracy remained stable at 96.5%. With only 70 neurons and Random Forest dimension reduction the model performed extremely well with 97.5% accuracy. Training and validation losses for this model were 0.0748 and 0.0924 respectively, also very low compared to other models (see appendix).

As the learning rate was decreased, we noticed that the best model performance improved by approximately 1%. It resulted in 20% increase in model training epoch. Using the SGD optimizer also improved the model by approximately 1%. However, the cost was reduced speed by 78% increase in epoch.

Conclusion

Overall, a model with Random Forest (RF) dimension reduction performed better at 97.5% accuracy compared with the best model we had selected. Hyper parameter tuning only increased the performance of our best model by the same amount as RF model with 70 features. The other advantage of this RF model is the 40% decrease in training and validation loss. By utilizing optimized principal components, the RF model is able to reduce noise and increase model efficiency and performance.

References

Deng, Li. 2012. “The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web].” *IEEE Signal Processing Magazine* 29 (6): 141–42.

LeCun, Y., B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, and W. E. Hubbard. 1990. “Handwritten Digit Recognition with a Backpropagation Network.” In , 396–404.

Memon, Jamshed, Maira Sami, Rizwan Ahmed Khan, and Mueen Uddin. 2020. “Handwritten Optical Character Recognition (OCR): A Comprehensive Systematic Literature Review (SLR).” *IEEE Access: Practical Innovations, Open Solutions* 8: 142642–68.

Nielsen, Michael A. 2015. “Neural Networks and Deep Learning.” http://neuralnetworksanddeeplearning.com/chap1.html.

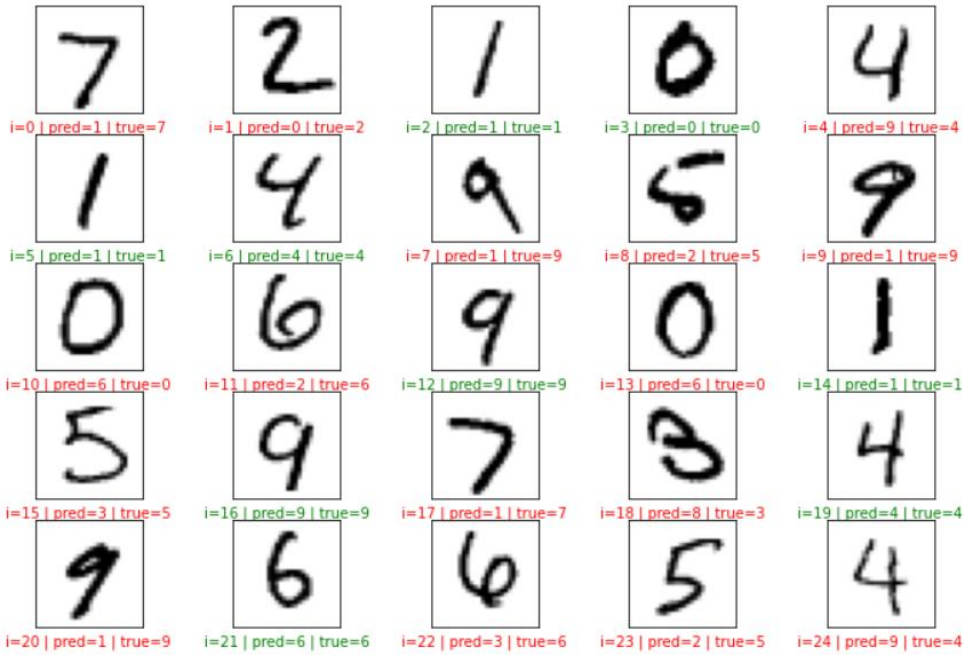
Niu, Xiao-Xiao, and Ching Y. Suen. 2012. “A Novel Hybrid CNN–SVM Classifier for Recognizing Handwritten Digits.” *Pattern Recognition* 45 (4): 1318–25.

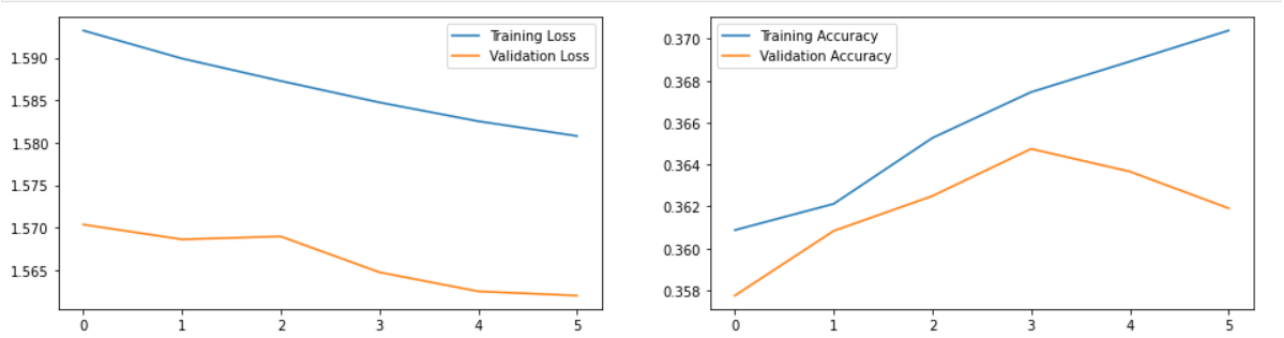
Pandey, Priyansh, Ritu Gupta, Mazhar Khan, Sajid Iqbal, and Amity University, Noida. 2020. “Multi-Digit Number Classification Using MNIST and ANN.” *International Journal of Engineering Research & Technology (Ahmedabad)* V9 (05). https://doi.org/10.17577/ijertv9is050330.

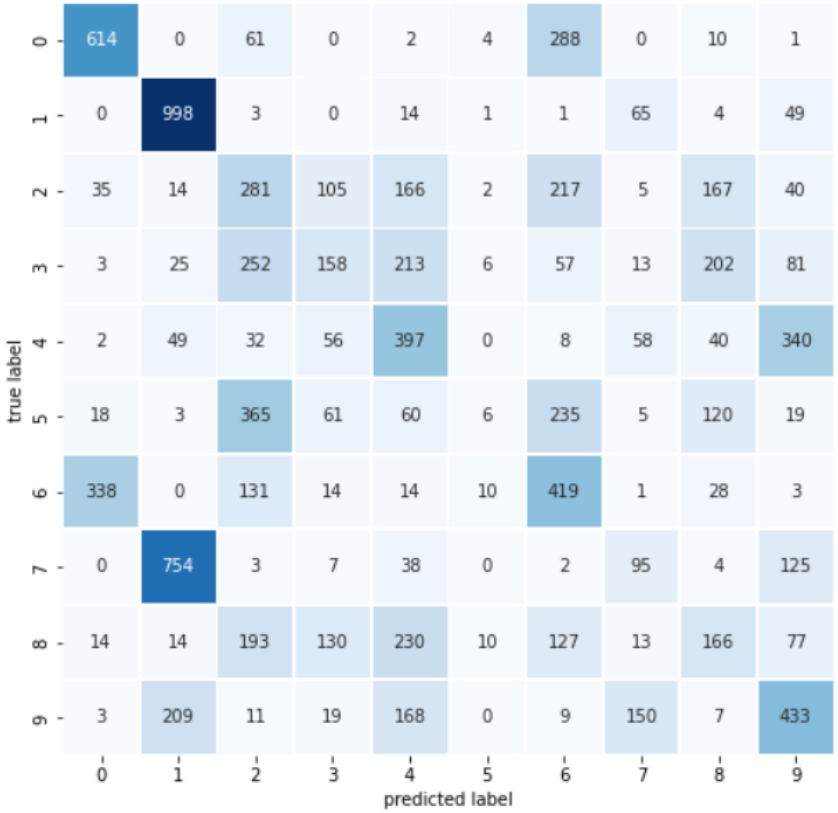
“Yann LeCun’s Home Page.” n.d. Lecun.Com. Accessed October 9, 2021. http://Yann.lecun.com.

Appendix

Experiment 1: MLP with 1 hidden neuron





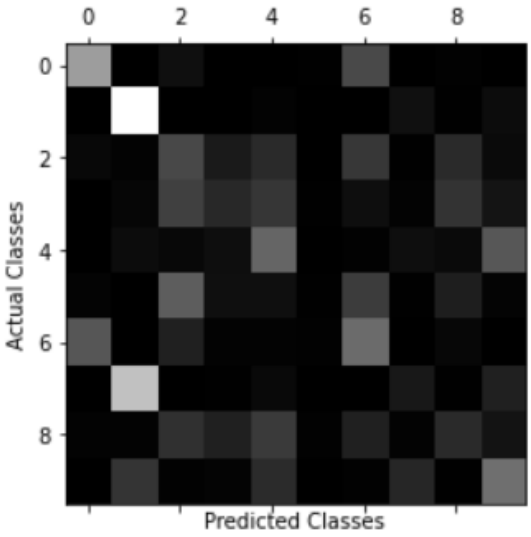


4's classified as 4's

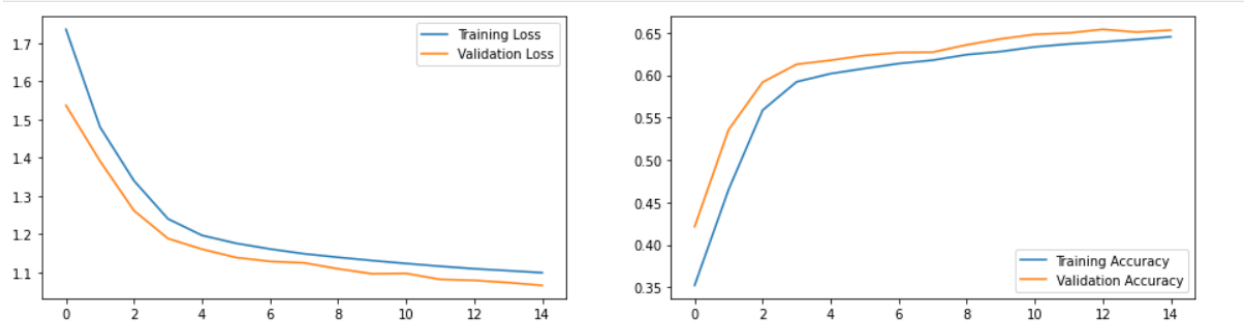
4's classified as 9's

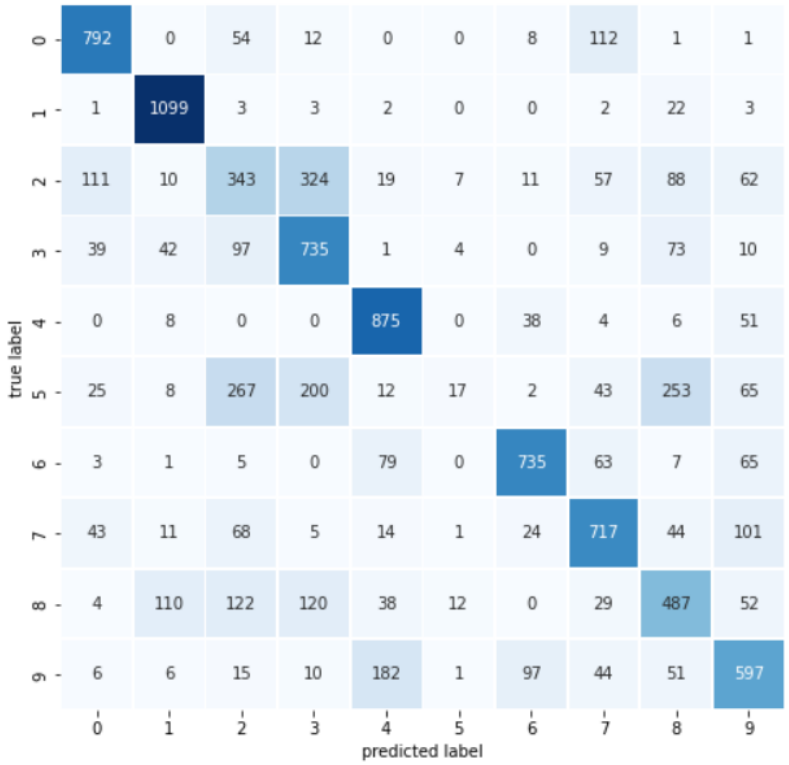
9's classified as 4's

9's classified as 9's



Experiment 2: MLP with 2 hidden neurons



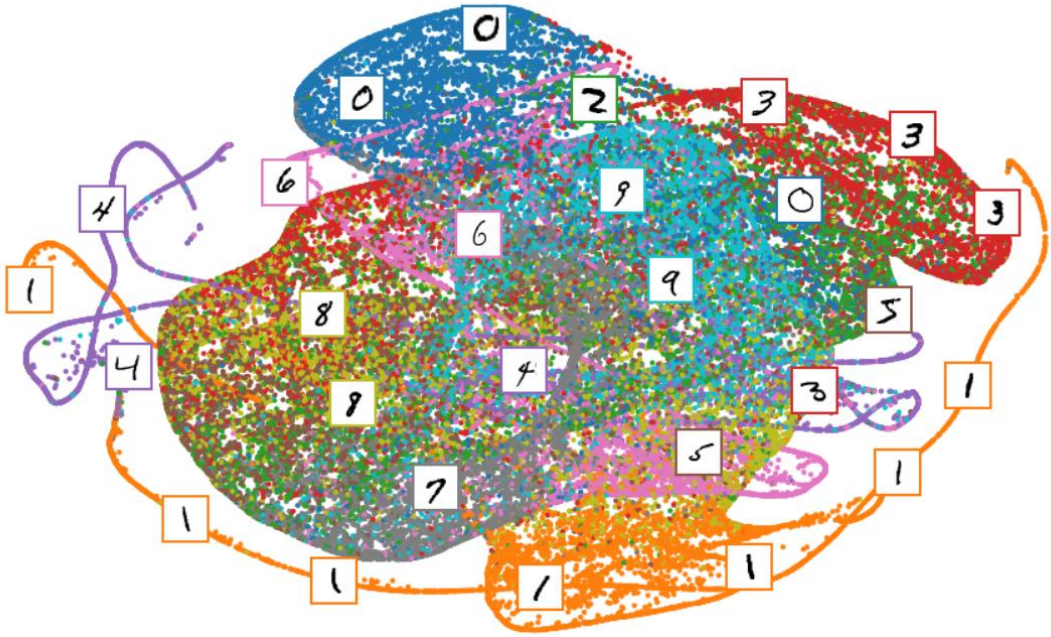


4's classified as 4's

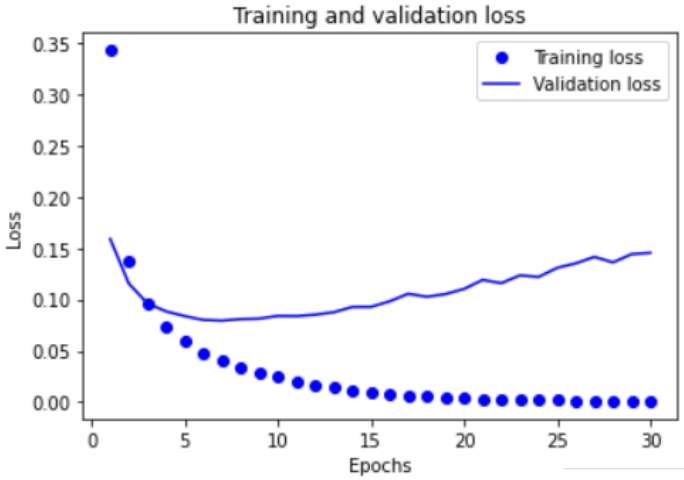
4's classified as 9's

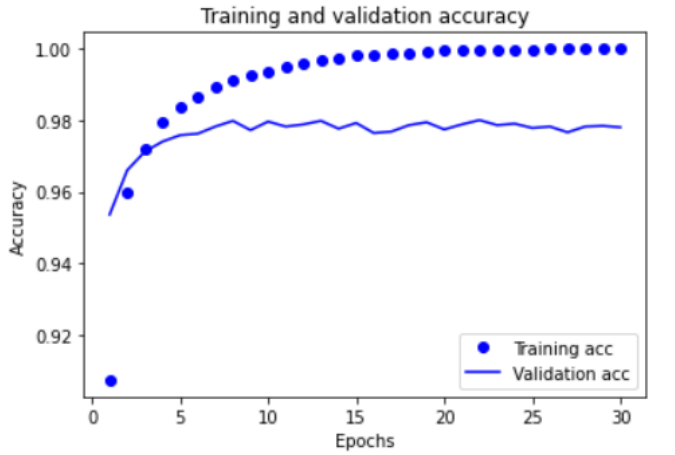
9's classified as 4's

9's classified as 9's



Experiment 4: MLP with 154 hidden Neurons with PCA Dimension reduction





Experiment 5: MLP with 70 hidden Neurons with RF Dimension reduction