**A.1 First Research/Programming Assignment**

**MNIST Classification**

MSDS 458 - Artificial Intelligence and Deep Learning

Grayson Matera

October 12, 2022

## Abstract

Utilizing the MNIST dataset, Deep Neural Network (DNN) architecture was explored in a series of five (5) experiments. The purpose of this research is to examine neural network structures consisting of a single hidden layer. Throughout the experimentation phase, small adjustments are made to the MNIST dataset and DNN structure. The results of the experiments are then compared to determine the impact of each DNN design. The methods of this research are non-comprehensive and open-ended. However, the conclusions drawn from these experiments can provide a broad look at the interactions between DNN structural changes and their relative performance outcomes. Phase 1 of experimentation, Experiments 1-3, is designed to explore the impact of the number of nodes in the DNNs single hidden layer. Our best performing model from this phase, consisting of 16 nodes, will then be deployed in Phase 2 where PCA decomposition and Random Forest classifier will be utilized to reduce the dimensionality of the dataset. Through the comparisons made across Phase 1, it is safe to conclude that increasing the number of nodes in our single hidden layer can have a substantial impact on network performance. However, it should be noted that a threshold of optimal node numbers exists where the benefit of increasing further is drastically reduced. Phase two comparison yielded less clear results than Phase 1, with dimensionality reduction have minimal impact on performance.


## Introduction

The problem of classifying handwritten digits has earned its place as a fundamental stepping-stone for machine learning (ML) and neural network research. Solving the problem serves as an optimal learning experience for many young data scientists looking to explore neural network architecture. Perhaps, this is because the methods involved in solving such a problem are capable of easily communicating the practical nature of classification. Classification through neural networks have a vast array of practical use cases—some much more complex than others. However, understanding the methods

of classification and the nuances of neural network structures can seem like a daunting task for beginners. By encouraging new learners to experiment with these methods in a simplified process, they can begin to formulate their own internal representation of how these methods function. The purpose of this research is to do just that—act as simple guide towards dismantling neural networks structures and diagnosing their performance and utility. The experiments that follow will utilize the MNIST dataset consisting of handwritten digits 0-9 to perform multiclass classification using DNN architecture.

## Literature Review

There is a vast and ever-growing pool of research dedicated to designing and testing DNNs. It's important to understand that the methods utilized in the following experiments only cover a small portion of DNN architecture and classification methods as a whole. DNN classification can be useful in many applications ranging from identifying and removing watermarks on photos to predicting the likelihood that someone will default on a loan (Shichang et al, 2021; Bayraci et al, 2019). In this literature review, we will stick to reviewing an applications that utilized the MNIST dataset in their research. The use case stated above where researchers identifying and removed watermarks from photos has done just that. Their methods were tested using the MNIST dataset and a standard generated watermark place on each image. The researchers utilized a two-phase watermark removal method using generative adversarial networks (GAN). Ultimately, the researchers we capable of training the network to classify the watermark, simulate it, and remove that portion generated portion of the image. Surprisingly, they were able to accomplish this with minimal training data.

## Methods

The following experiments were run using python via the Google CoLab environment. This was strategically chosen to help reduce runtimes and GPU strain. Additionally, the environment comes preinstalled with all the packages necessary for the experiments. These packages include numpy and

pandas for data manipulation, matplotlib and seaborn for visualization, an array of module from sklearn and keras for modelling. The MNIST dataset is also preinstalled. Upon loading in the MNIST dataset consists of 70,000, it was split into 60,000 training images and 10,000 test images. During experimentation, 5,000 of the 60,000 training images were held back for validation. Before the dataset is put to use, it undergoes preprocessing to convert the labels to a 1D array and the images from a 2D array to a 1D array. We then applied one-hot encoding to our labels.

The experimental portion of our research consists of five (5) individual experiments separated into two (2) phases. In Phase 1, consisting of experiments 1-3, a DNN with a single hidden layer is implemented, trained, and tested on our preprocessed data. The number of nodes in the single hidden layer is increased in each subsequent experiment and the results are compared. Each DNN model utilizes ReLu for its activation function and keras's Sequential model.

In Experiment 1, the model's input shape is set to fit the (60000, 784) shape of our data with 128 nodes on our input layer. The hidden layer in Experiment 1 is set to only include one node. Below is a summary of this model.

```
model1.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 128)               100480

 hidden_layer (Dense)        (None, 1)                 129

 output_layer (Dense)        (None, 10)                20

=================================================================
Total params: 100,629
Trainable params: 100,629
Non-trainable params: 0
_________________________________________________________________
```

In Experiment 2, the model is adjusted to include two nodes in its hidden layer. Experiment 3 contains multiple models adjusting the hidden layers number of nodes to 5, 16, 32. After comparing all five models, the best performing model is chosen for further experimentation in Phase 2.

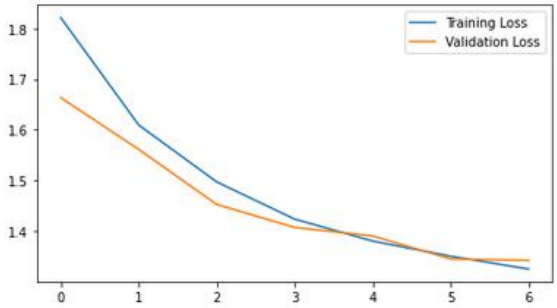
Phase 2, consisting of Experiments 4 & 5, will implement PCA decomposition and Random Forest classifier on the MNIST dataset. This dimensionally reduced data will then be applied to the best performing model from Phase 1. The results of both Phase 2 experiments will be compared to the original results of chosen model from Phase 1.

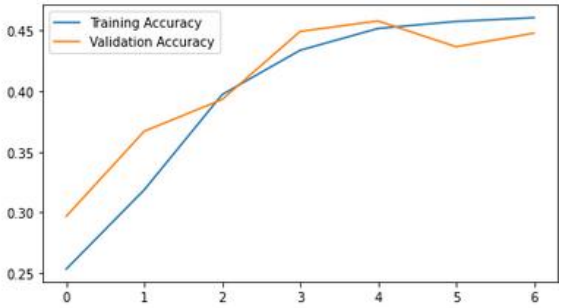
## Results

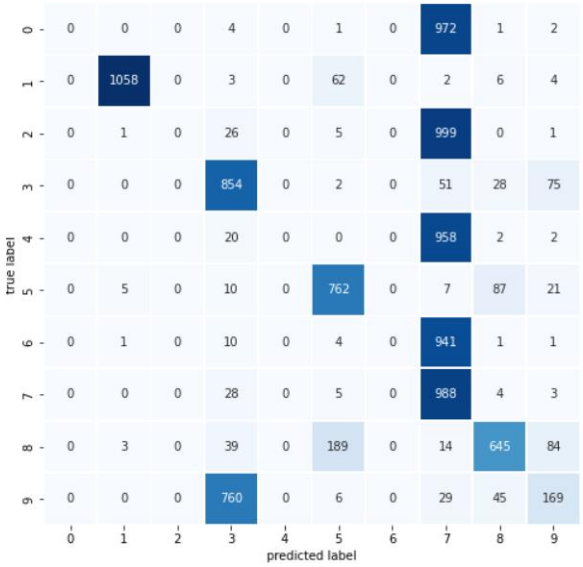
This section will cover the results of each of our experiments in order. Comparisons will then be made when relevant to our methodology.

**Experiment 1:**

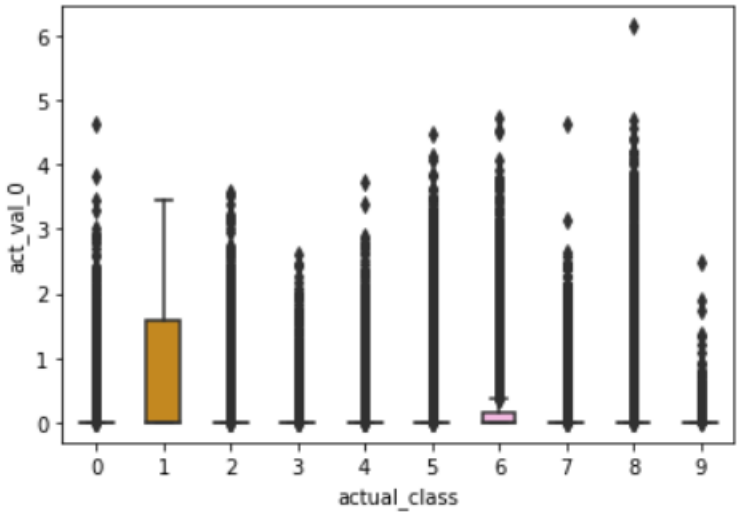
Model1 consisted of 1 node in the model's single hidden layer. It performed better than expected with a final test accuracy of 48.8. It appears to have trained fairly well with a loss of only 1.3649. However, its accuracy was likely capped due to the limited number of nodes.





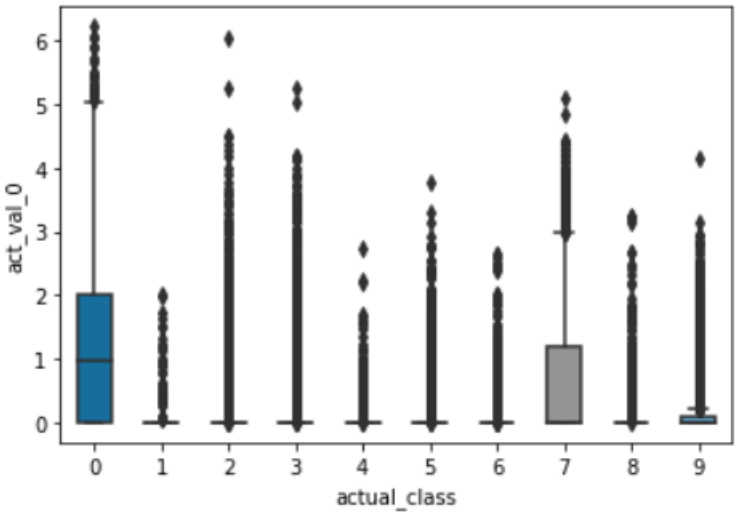

Its clear from the confusion matrix that the model had a difficult time classifying the data despite its low loss in training. The boxplots show that there is quite a bit of overlap between the classes.

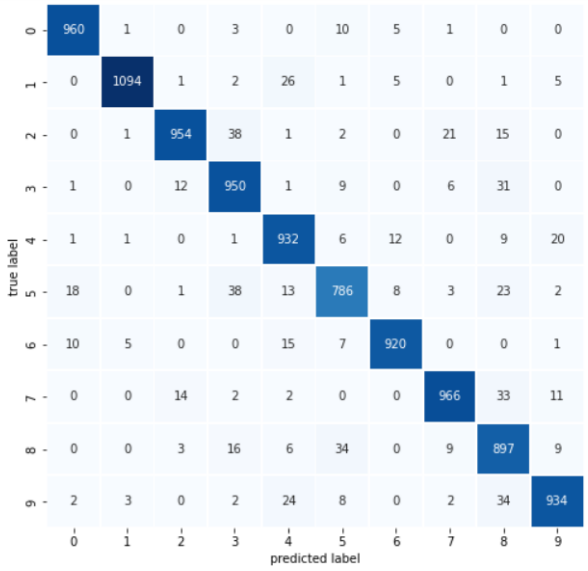


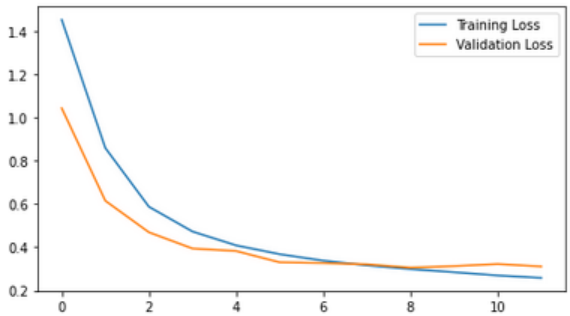

**Experiment 2:**

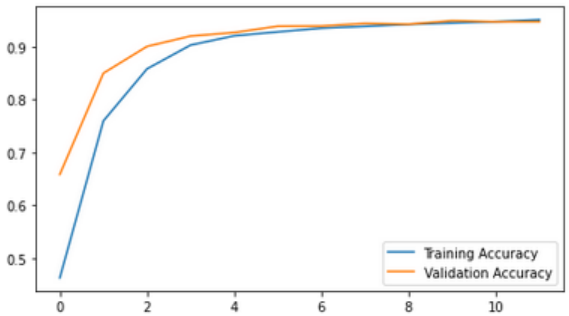
Model2, consisting of 2 nodes, performed much better then model1. The model performed with a test accuracy of 93.93 and a loss of only 0.36. The confusion matrix puts displays just how well model2

does compared to model1 when making accurate predictions. Model2's boxplot also displays less overlap between the classes which speaks to how the nodes are correlating class label with activation values.



**Experiment 3:**

In Experiment 3 we compare three more models with increasing nodes. Our first model in this experiment contains 5 nodes in its hidden layer. The second model contains 16 nodes and the third model contains 32 nodes. All models in this experiment performed better than the previous experiments. The first model (5 nodes) performed with a test accuracy score of 96.49 and a loss of 0.1589. This model was outperformed by our second model (16 nodes) with a test accuracy of 97.89 and loss of 0.1008. The third model (32 nodes) saw a decrease in test accuracy 97.33 and loss of 0.0994. The performance between the second and third model were similar. However, there was markedly more activation in the 16 node model and less error. The 16 node model will be chosen as our best model and used in the next phase of research.

Classification Report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.99 | 0.99 | 0.99 | 980 |
| 1 | 0.99 | 0.99 | 0.99 | 1135 |
| 2 | 0.98 | 0.98 | 0.98 | 1032 |
| 3 | 0.97 | 0.98 | 0.98 | 1010 |
| 4 | 0.98 | 0.98 | 0.98 | 982 |
| 5 | 0.98 | 0.97 | 0.97 | 892 |
| 6 | 0.98 | 0.99 | 0.98 | 958 |
| 7 | 0.98 | 0.96 | 0.97 | 1028 |
| 8 | 0.97 | 0.97 | 0.97 | 974 |
| 9 | 0.97 | 0.97 | 0.97 | 1009 |
| accuracy | | | 0.98 | 10000 |
| macro avg | 0.98 | 0.98 | 0.98 | 10000 |
| weighted avg | 0.98 | 0.98 | 0.98 | 10000 |

Accuracy Score: 0.9789
Root Mean Square Error: 0.6293647591023825

```
Classification Report
              precision    recall  f1-score   support

           0       0.98      0.99      0.98       980
           1       1.00      0.99      0.99      1135
           2       0.99      0.97      0.98      1032
           3       0.95      0.99      0.97      1010
           4       0.96      0.99      0.97       982
           5       0.99      0.94      0.96       892
           6       0.98      0.97      0.98       958
           7       0.98      0.96      0.97      1028
           8       0.95      0.97      0.96       974
           9       0.96      0.97      0.97      1009

    accuracy                           0.97     10000
   macro avg       0.97      0.97      0.97     10000
weighted avg       0.97      0.97      0.97     10000

Accuracy Score: 0.9733
Root Mean Square Error: 0.666783323126786
```

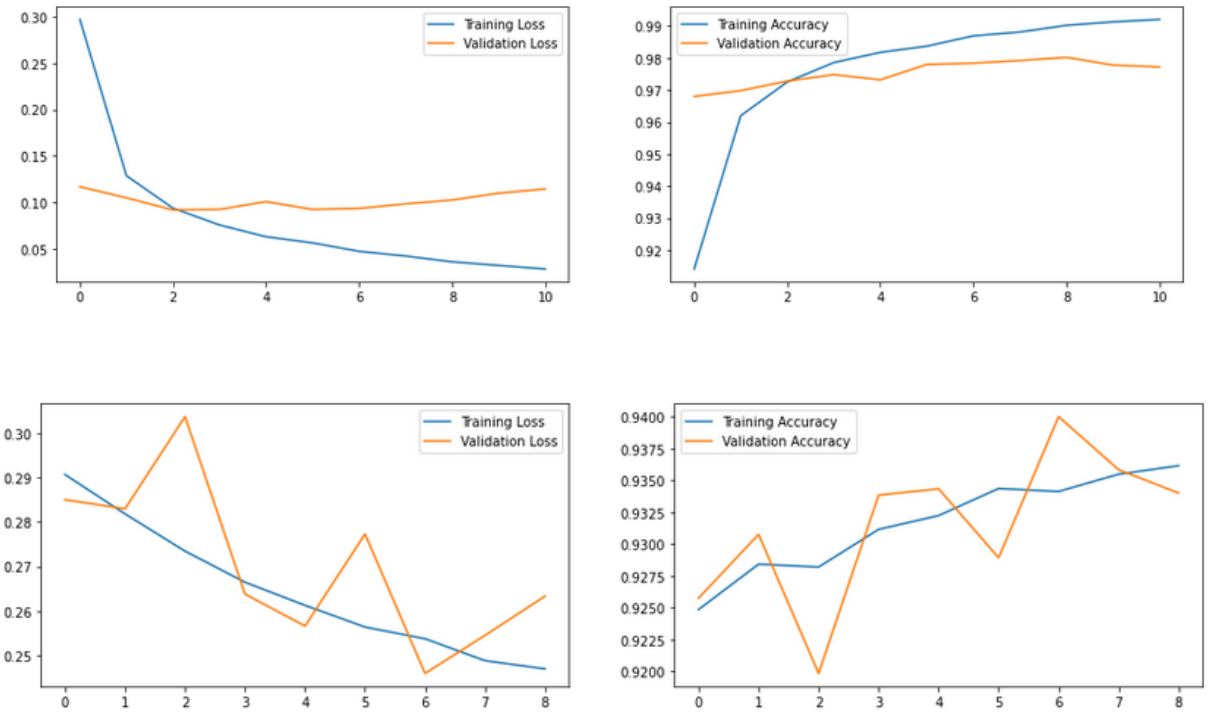
**Experiment 4:**

In Experiment 4, PCA decomposition is used to reduce the dimensionality of the training set. This experiment trained with an val_accuracy score of 0.98 and seemed to have the potential to outperform our best model from Phase 1. However, there was significant difficulties in getting this to run properly against the test data. Despite much effort, it seems there was a grave error in implementation that resulted in the test set accuracy scoring 13.73 with a loss of 2234.27. This is likely do to a technical error, but shows promise that dimensionality reduction using PCA can be an effective tool if performed properly.

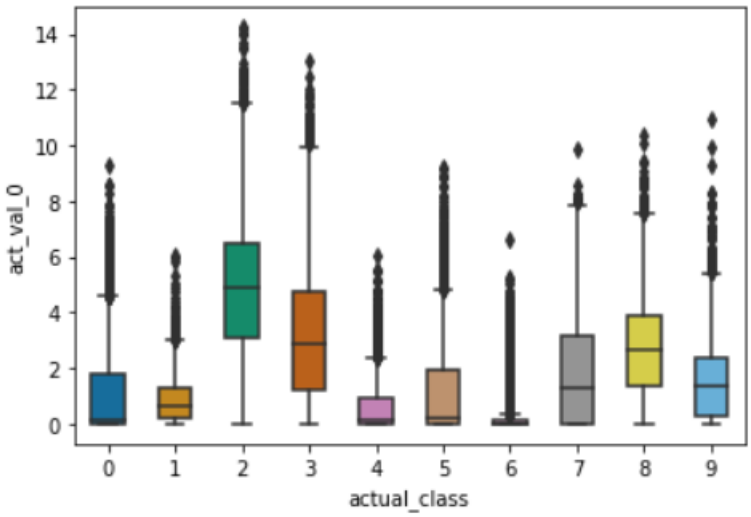
**Experiment 5:**

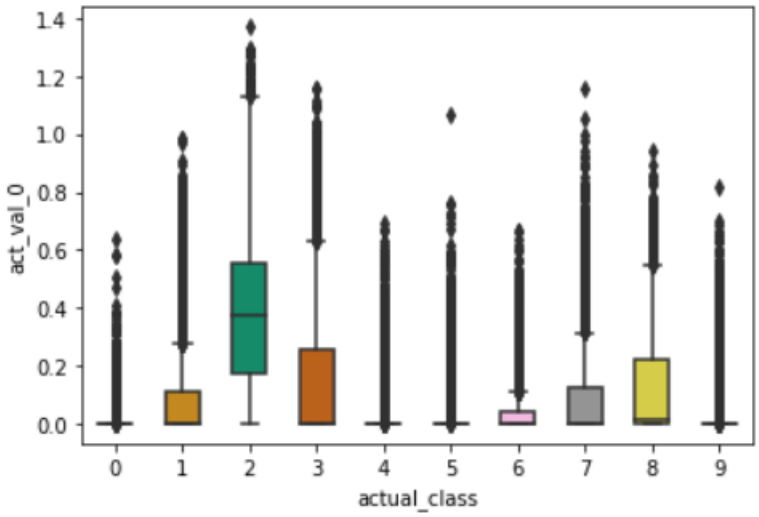
In Experiment 5, we utilize Random Forest classifier to reduce the dimensions of the image (28x28) to the most useful 70 features (pixels) of the image. We then deploy of best model from Phase 1 (16 nodes) to train on this new data. The 16 node model with Random Forest classifier did not outperform the original model. Its test accuracy score was 93.26 with a loss of 0.2736. The loss indicates that it did not train as well on this data set. We can better understand this through visualization. The top chart

displays the loss and accuracy of the original model while the bottom displays the model from Experiment 5.




The activation of the nodes in our experiment 5 model was also significantly lower than the original best model. We can view this more easily through boxplot visualization. The top image is from the original best model, while the bottom is our newly trained version.




## Conclusions

Through experimentation with the number of nodes in a hidden single hidden layer DNN, we can determine several important mechanisms at play. DNN models with very few nodes can still train and perform surprisingly well, however, they are unlikely to perform well enough for practical application. The simple process of increasing the number of nodes will benefit the model's performance drastically. That being said, there is likely a threshold for the optimal number of nodes to use in a DNN model. It is

not entirely clear from these experiments how this is determined, however, DNN performance appears to decrease once this threshold is passed.

Furthermore, PCA decomposition for dimensionality reduction show promise in improving the performance of a model. This conclusion is made despite the technical errors in implementation. Contrary to this, the effectiveness of Random Forest classifier remains in question. Although its application would theoretically appear to benefit the models performance, our experimentation on the subject are inconclusive. Nevertheless, the benefit of experimentation in this area is apparent. Just like DNNs, humans learn from training, adjusting methods, and trying again. There remains no better way to understand the nuances of DNN models than trial and error.

## References

Selçuk BAYRACI, & Orkun SUSUZ. (2019). A Deep Neural Network (DNN) based classification model in application to loan default prediction. *Theoretical and Applied Economics*, *XXVI*(4), 75–84.

Sun, Wang, H., Xue, M., Zhang, Y., Wang, J., & Liu, W. (2021). Detect and Remove Watermark in Deep Neural Networks via Generative Adversarial Networks. In *Information Security* (pp. 341–357). Springer International Publishing. https://doi.org/10.1007/978-3-030-91356-4_18

Tolba, Tesfai, H. T., Saleh, H., Mohammad, B., & Al-Qutayri, M. (2022). Deep Neural Networks-Based Weight Approximation and Computation Reuse for 2-D Image Classification. *IEEE Access*, *10*, 41551–41563. https://doi.org/10.1109/ACCESS.2022.3161738