MSDS 458 Fall 2021 Assignment 1

Husein Adenwala

1. Abstract

The objective of this research was to understand how the architecture of a dense neural network, specifically multilayer perceptions impact performance in classifying handwritten images. I will be working with a simple neural network with a single hidden layer and double hidden layer nodes consisting of different numbers of nodes to analyze their impact on performance. For this research, a set of experiments was performed on the MNIST dataset, a collection of 28×28-pixel grayscale images of handwritten single digits between 0 and 9, that is widely used to benchmark and test classification algorithms. Results from my experiment suggest that increasing the width of a hidden layer increases the performance of the model; however, there is diminishing marginal return of performance with additional Nodes. I have also conducted experiments in dimension reduction by using Principal Component Analysis (PCA) and Random Forest classifier, which reduces the number of inputs features. The results show that training a model using PCA to reduce dimensions is computationally efficient and provides higher accuracy compared to models that do not utilize dimension reduction.

2. Introduction

In the 1990s, the United States Postal Services (USPS) had automated sorting based on zip code; however, at that time the algorithms used in sorting the mail were only able to decipher zip codes with 90% accuracy which resulted in 10% mail getting delayed or lost due to incorrect sorting. USPS has more than 100 billion mail deliveries in a year and 10% inaccuracy is a significant number, which led the USPS to initiate a project to develop neural net that could improve zip code sorting accuracy. The neural net model that was developed in this project resulted in 99% accuracy. Looking at the success of the US postal service, my company has asked me to create a neural net that could automate reading handwritten customer contact credit card numbers they send as a payment method. Processing customer payment with manual entry is very time intensive and is prone to human errors. Automating this process would result in significant cost and time savings for the company.

3. Literature review

In broad terms, image recognition is a classification problem and in AI this falls under the field of computer vision. Neural Networks, specifically Convolution Neural Nets (CNNs) based models are the go-to model for computer vision. One of the best performing models for image recognition is Very Deep Convolutional Networks for Large-Scale Image Recognition (VGG16) (Simonyan 2015). It is a convolution neural net (CNN) architecture which was used to win the ILSVR competition in 2014. Recently, attention-based networks, such as the Vision Transformer (ViT) have achieved remarkable results compared to CNN while requiring fewer computational resources for pre-training. However, there is recent MLP-Mixer (Tolstikhin 2021) architecture based exclusively on multi-layer perceptron's (MLPs) which has a simpler architecture, it has a higher throughput and is computationally more efficient.

Machine learning models such as decision trees, Support Vector Machine (SVM) algorithm, K-Nearest Neighbor algorithm (KNN) can also provide good accuracy for image classification problems, but don't do as well as neural nets especially when features are complex and there is a large data set. However, these are still used for dimension reduction, as in most artificial intelligence and machine learning problems there are thousands or even millions of features that are used in training. These features make training slow and may make it harder to find a good solution (curse of dimensionality). Principle Component Analysis (PCA) is one of the most popular dimensionality reduction algorithms (Geron 2019). PCA identifies the axis that accounts for the highest variance in the training set and a second axis that is orthogonal to the first one that accounts for the largest amount of remaining variance. Alternatively, Random Forest classifier can be used to evaluate the feature importance method. The feature importance can be measured as the average impurity decrease computed from all decision trees in the forest (Kumar 2020).

4. Methods

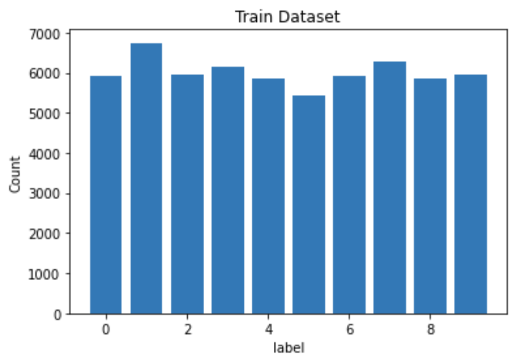
Research design and modeling methods

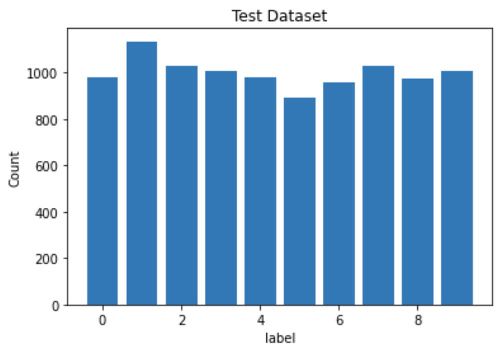
I have decided to use multilayer perceptron (MLP) to classify handwritten digits. A MLP is a class of feedforward artificial neural network. A MLP consists of at least three layers of nodes: an input layer, a hidden layer, and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable. This makes MLP a good candidate for this computer vision problem.

Each MLP Model has a variety of neurons to determine the number of nodes that can provide the best accuracy. The models in my experiments used a single layer design with 1, 2, 128, 256, 512 and 1024 nodes, respectively. I also developed two additional models, which implemented dimension reduction of features from 784 to 154 and 70 by using PCA and Random Forest classification methods respectively and for both experiments, I used 85 nodes for each of the single layers. I also developed a 2-layer model, each consisting of 64 nodes and 128 nodes for each of the layers respectively. All models use the ReLU activation function which is one of the popular activation functions in neural networks because it is computationally efficient and fixes the problem of vanishing gradient. I used Accuracy as the metric to evaluate the performance. Finally, I used categorical cross entropy to measure the loss between labels and predictions and SoftMax activation function for the output layer.

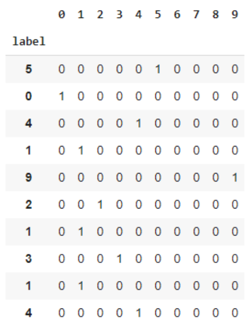
Data preparation, exploration, visualization

The dataset used in this research is the MNIST dataset which was released in 1999 and is a collection of handwritten images that contain 10 classes of digits (0 to 9) that is widely used to benchmark and test classification algorithms. The dataset is loaded from the Keras framework and contains 60,000 training and 10,000 test 28x28 grayscale images of the 10 digits in a NumPy array of target variable (y), which contains label values from 0 to 9 and of features (x) which is in a 28 by 28 array and contains values from 0 to 255. In all experiments, I used 20% of the images from the 60,000 training images as validation dataset that was used as a performance metric in training in the model. The dataset was evenly distributed among all the categories as can be seen in the bar chart below





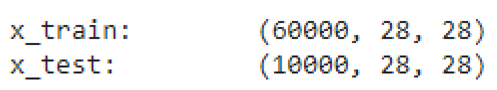
The labels were transformed using One-hot encoding, which involves representing each categorical variable with a binary vector that has one element for each unique label and marking the class label with a 1 and all other elements 0. I created a 10-class label of the target variable as seen in the example below.



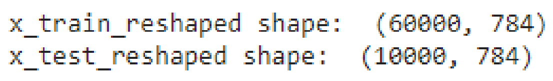
label	0	1	2	3	4	5	6	7	8	9
5	0	0	0	0	0	1	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0
4	0	0	0	0	1	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	1
2	0	0	1	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0
3	0	0	0	1	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0
4	0	0	0	0	1	0	0	0	0	0

Feature Transformation: Reshape and Normalization

I unrolled each 28 x 28 array to create a (784, 1) vector, to make it a series of columns.

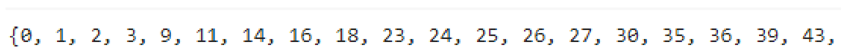


```
x_train:          (60000, 28, 28)
x_test:           (10000, 28, 28)
```



```
x_train_reshaped shape:  (60000, 784)
x_test_reshaped shape:  (10000, 784)
```

I then did min max normalization to scale the data from 0 to 1, as normalizing the data generally speeds up learning and leads to faster convergence.



```
{0, 1, 2, 3, 9, 11, 14, 16, 18, 23, 24, 25, 26, 27, 30, 35, 36, 39, 43,
```

Implementation and programming

Model research was done using Python programming language and TensorFlow. TensorFlow is a free and open-source software library for machine learning and artificial intelligence and TensorFlow can be implemented in python by Keras. Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow which is extremely user-friendly.

I have used the sci-kit learn library which has functions to create a confusion matrix and report on precision, accuracy and F1 score, and it also has packages that can implement PCA, T-SNE and Random Forest Classifiers.

Additionally, I used NumPy and Pandas library for data preprocessing and finally I also used seaborn package and matplotlib library to visualize the data and results.

5. Results

I conducted 10 experiments and all the models in my experiments have the following hyper parameters:

- Optimizer = RMSprop
- L2 regularization
- Early Stopping with a patience of 10

Experiment 1

A dense neural network with 784 input nodes, a hidden layer with 1 node, and 10 output nodes.

Model achieved accuracy of 37% with a loss of 1.6 on the test data. It could not discriminate between different classes as you can see in the confusion matrix and box plot in the Appendix A.

Experiment 2

A dense neural network with 784 input nodes, a hidden layer with 2 nodes, and 10 output nodes.

Model achieved accuracy of 69% with a loss of 0.99 on the test data. The model did better than model 1.

The confusion matrix highlights that the model is confusing the actual versus predicted classes such as the digits 3, 5 and 8. The f1 score for 3, 5 and 8 are also low. The performance is uneven between classes (see Appendix B).

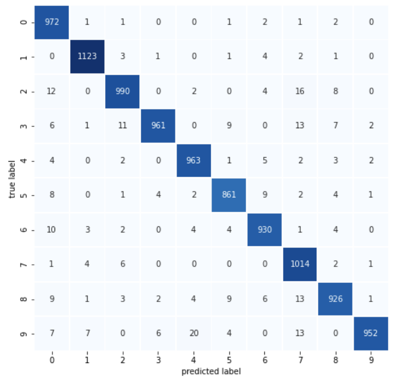
The results show that increasing the width of the hidden layer by adding more nodes improves the performance of the model.

Experiment 3

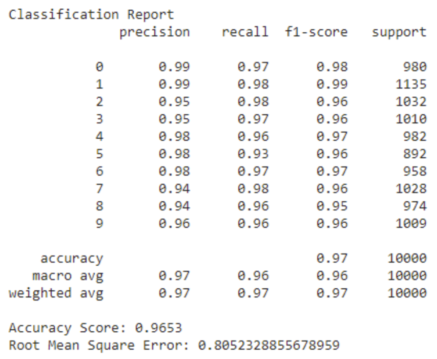
A dense neural network with 784 input nodes, a hidden layer with 128 neurons, and 10 output nodes.

Model achieved accuracy of 97% with a loss of 0.16 on the test data.

The confusion matrix highlights that the model predicted accurately for all the classes. F1, precision and recall scores are also even between different classes.



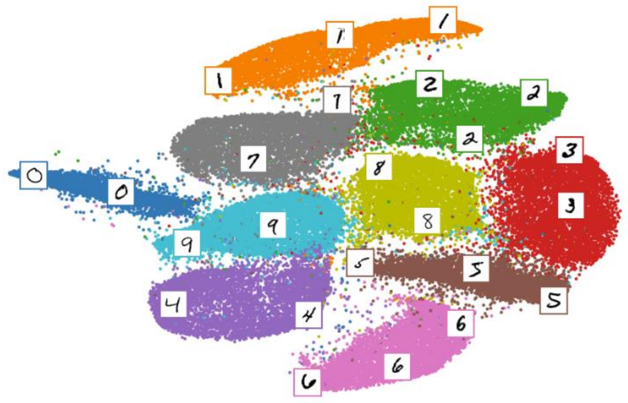
true label \ predicted label	0	1	2	3	4	5	6	7	8	9
0	972	1	1	0	0	1	2	1	2	0
1	0	1123	3	1	0	1	4	2	1	0
2	12	0	990	0	2	0	4	16	8	0
3	6	1	11	961	0	9	0	13	7	2
4	4	0	2	0	963	1	5	2	3	2
5	8	0	1	4	2	861	9	2	4	1
6	10	3	2	0	4	4	930	1	4	0
7	1	4	6	0	0	0	0	1014	2	1
8	9	1	3	2	4	9	6	13	926	1
9	7	7	0	6	20	4	0	13	0	952



```
Classification Report
              precision    recall  f1-score   support

           0       0.99      0.97      0.98       980
           1       0.99      0.98      0.99      1135
           2       0.95      0.98      0.96      1032
           3       0.95      0.97      0.96      1010
           4       0.98      0.96      0.97       982
           5       0.98      0.93      0.96       892
           6       0.98      0.97      0.97       958
           7       0.94      0.98      0.96      1028
           8       0.94      0.96      0.95       974
           9       0.96      0.96      0.96      1009

    accuracy                           0.97     10000
   macro avg       0.97      0.96      0.96     10000
weighted avg       0.97      0.97      0.97     10000

Accuracy Score: 0.9653
Root Mean Square Error: 0.8052328855678959
```

I used t-Distributed Stochastic Neighbor Embedding (t-SNE) to reduce the dimension of 128 nodes to 2.

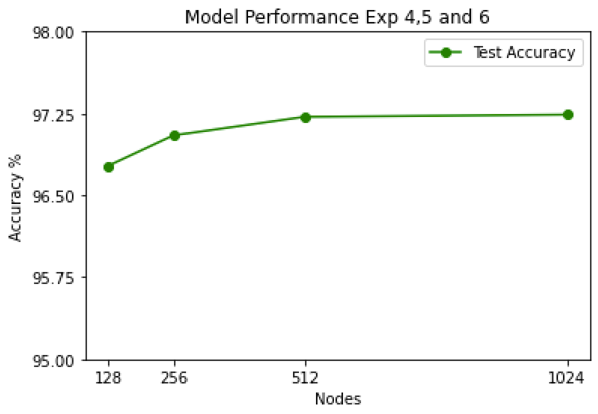
The T-SNE plot shows how well the 128 nodes can discriminate between each label.



Experiments 4, 5 and 6

In experiments 4, 5 and 6, I tested on expanding the nodes by doubling the nodes in a single hidden layer after each experiment. Experiments 4, 5 and 6 each have a hidden layer of 256,512 and 1024 neurons respectively. The input and output nodes were 784 and 10 for each experiment. As the nodes increased the performance of the model did improve slightly; however, it did not change in any significant way, though the training time increased. Therefore, it appears that adding more neurons makes the model computationally expensive.

Models using 128, 256, 512 and 1024 nodes had 96-97% test accuracy (see Appendix C, Appendix D and Appendix E). Confusion matrix and F1 and recall and precision scores were very similar. Adding layers had marginal performance improvements as seen in the plot below.

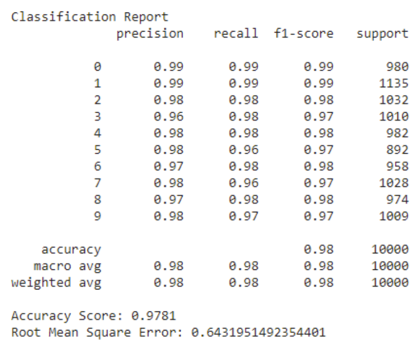


Experiment 7

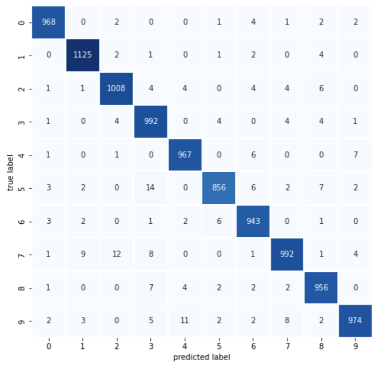
In this Experiment, I tested the impact of using Principal Component Analysis (PCA).

PCA reduced the number of dimensions from 784 to 154 that represented 95% of variability. The dimension reduction reduces computation time and can be a very important tool for a very large dataset.

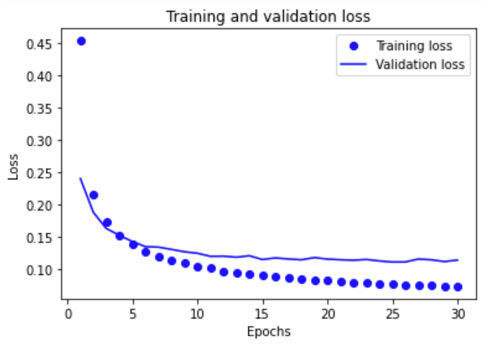
The model comprised a dense neural network with 154 input nodes, a hidden layer with 85 neurons, and 10 output nodes. It achieved an accuracy of 97.32% with a loss of 0.138 on the test data.

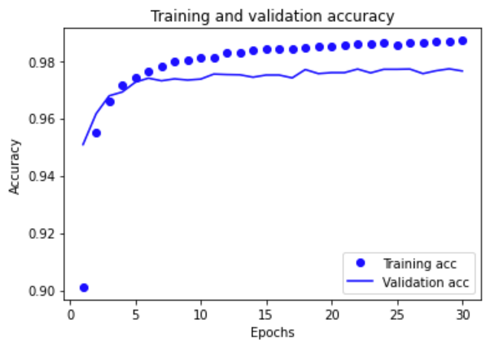


```
Classification Report
              precision    recall  f1-score   support

           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       0.96      0.98      0.97      1010
           4       0.98      0.98      0.98       982
           5       0.98      0.96      0.97       892
           6       0.97      0.98      0.98       958
           7       0.98      0.96      0.97      1028
           8       0.97      0.98      0.98       974
           9       0.98      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: 0.9781
Root Mean Square Error: 0.6431951492354401
```



true label \ predicted label	0	1	2	3	4	5	6	7	8	9
0	968	0	2	0	0	1	4	1	2	2
1	0	1125	2	1	0	1	2	0	4	0
2	1	1	1008	4	4	0	4	4	6	0
3	1	0	4	992	0	4	0	4	4	1
4	1	0	1	0	967	0	6	0	0	7
5	3	2	0	14	0	856	6	2	7	2
6	3	2	0	1	2	6	943	0	1	0
7	1	9	12	8	0	0	1	992	1	4
8	1	0	0	7	4	2	2	2	956	0
9	2	3	0	5	11	2	2	8	2	974





Experiment 8

I tested the impact of Random Forest classifier in reducing the input feature dimensions. Using Random Forest classifier, I found the relative importance of the 784 features and reduced the features to the 70 most important features.

The model comprised a dense neural network with 70 input nodes, a hidden layer with 85 neurons, and 10 output nodes. It achieved a test accuracy of 92.62% and loss of 0.28.

The model performed comparatively badly. Additionally, dimension reduction using random forest is computationally expensive compared to PCA.

Experiment 9 and 10

In Experiments 9 and 10, I increased the width of the neural net to see if there was improvement in performance.

Experiment 9 had 784 input nodes, two hidden layers with 64 neurons each and 10 output nodes.

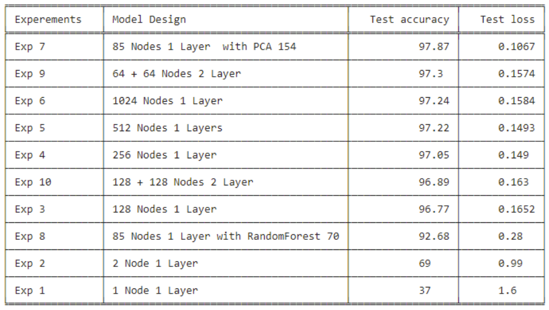
The model achieved an accuracy of 97.3% with a loss of 0.15 on the test data (Appendix G)..

Experiment 10 had 784 input nodes, two hidden layers with 128 neurons each and 10 output nodes.

The model achieved an accuracy of 96.8% with a loss of 0.16 on the test data (Appendix H.

Increasing the width of the dense layer did not have any significant improvement on the model performance but it increased the computation time.

6. Conclusions.



Experements	Model Design	Test accuracy	Test loss
Exp 7	85 Nodes 1 Layer with PCA 154	97.87	0.1067
Exp 9	64 + 64 Nodes 2 Layer	97.3	0.1574
Exp 6	1024 Nodes 1 Layer	97.24	0.1584
Exp 5	512 Nodes 1 Layers	97.22	0.1493
Exp 4	256 Nodes 1 Layer	97.05	0.149
Exp 10	128 + 128 Nodes 2 Layer	96.89	0.163
Exp 3	128 Nodes 1 Layer	96.77	0.1652
Exp 8	85 Nodes 1 Layer with RandomForest 70	92.68	0.28
Exp 2	2 Node 1 Layer	69	0.99
Exp 1	1 Node 1 Layer	37	1.6

The results of the 10 experiments show that increasing the number of neurons improves the performance, but performance improvement stagnates after a certain threshold. The more neurons we have, we can extract more features, but it can also make the model overfit the training data and therefore just increasing the number of layers or adding more dense layers like in experiment 9 and 10 does not improve the performance significantly; experiment 10 with two hidden dense layers of 128 shows signs of overfitting with divergence in test and train accuracy.

I also tested how reducing dimensions with Principal Component Analysis (PCA) and Random Forest impact model performance. Dimension reduction with random forest did not perform well with a 92.68% accuracy and average f1 score of 0.93. On the other hand, PCA perform very well and had a slight edge in performance over the models that did not reduce the dimensions, achieving the best test accuracy of 97.87% and an average f1 score of 0.98.

In conclusion, I would recommend using PCA dimension reduction technique in neural net as demonstrated experiment 7. it is computation efficient, and it was best performing model.

References:

USPS (n.d). *Postal Facts.* https://facts.usps.com/

Boesch, G. (2021). Vision Transformers (ViT) in Image Recognition – 2021 Guide https://viso.ai/deep-learning/vision-transformer-vit/

Frahcois, C. (2018). Deep Learning with Python. Shelter Island, New York: Manning Publications Co.

Geȉron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras and TensorFlow: concepts, tools, and techniques to build intelligent systems* (2nd ed.). O'Reilly.

Kumar, Ajitesh. 2020. "Feature Importance Using Random Forest Classifier - Python." *Data Analytics* (blog). August 2, 2020. https://vitalflux.com/feature-importance-random-forest-classifier-python/.

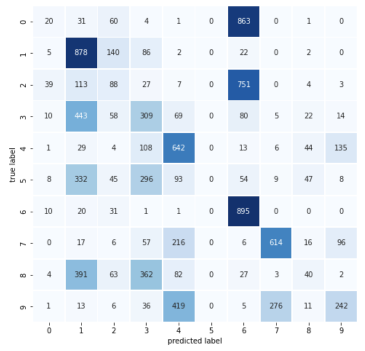
LeCun, L., Corinna, C., & Burges, C. (n.d). THE MNIST DATABASE of handwritten digits http://yann.lecun.com/exdb/mnist/

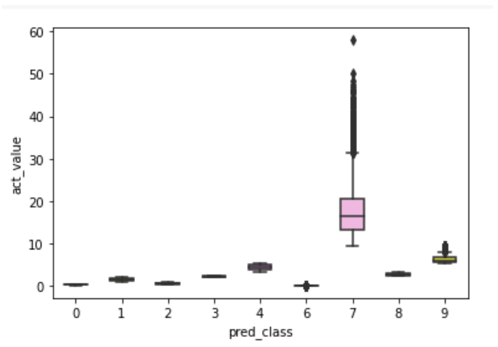
Simonyan, K., & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. CoRR, abs/1409.1556.

Tolstikhin, I.O., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Keysers, D., Uszkoreit, J., Lucic, M., & Dosovitskiy, A. (2021). MLP-Mixer: An all-MLP Architecture for Vision. ArXiv, abs/2105.01601.

Appendix A

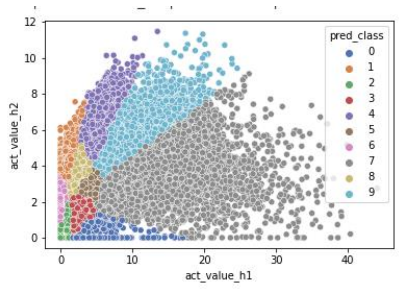
Experiment 1 result

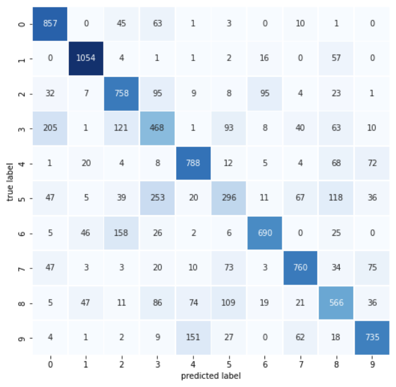




Appendix A

Experiment 2 result





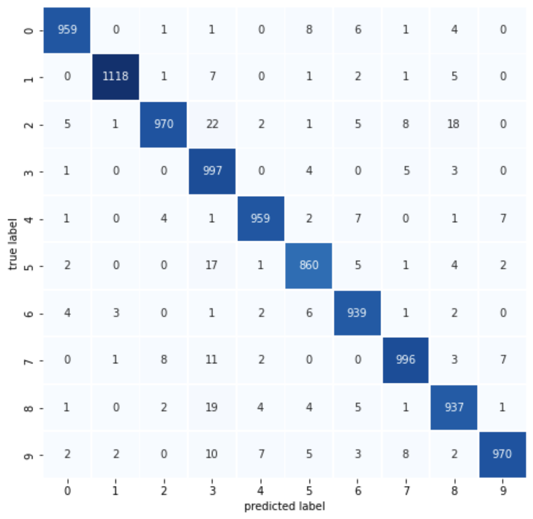
	0	1	2	3	4	5	6	7	8	9
0	857	0	45	63	1	3	0	10	1	0
1	0	1054	4	1	1	2	16	0	57	0
2	32	7	758	95	9	8	95	4	23	1
3	205	1	121	468	1	93	8	40	63	10
4	1	20	4	8	788	12	5	4	68	72
5	47	5	39	253	20	296	11	67	118	36
6	5	46	158	26	2	6	690	0	25	0
7	47	3	3	20	10	73	3	760	34	75
8	5	47	11	86	74	109	19	21	566	36
9	4	1	2	9	151	27	0	62	18	735

```
Classification Report
              precision    recall  f1-score   support

           0       0.71      0.87      0.79       980
           1       0.89      0.93      0.91      1135
           2       0.66      0.73      0.70      1032
           3       0.45      0.46      0.46      1010
           4       0.75      0.80      0.77       982
           5       0.47      0.33      0.39       892
           6       0.81      0.72      0.76       958
           7       0.79      0.74      0.76      1028
           8       0.58      0.58      0.58       974
           9       0.76      0.73      0.74      1009

    accuracy                           0.70     10000
   macro avg       0.69      0.69      0.69     10000
weighted avg       0.69      0.70      0.69     10000

Accuracy Score: 0.6972
Root Mean Square Error: 2.016705233790997
```

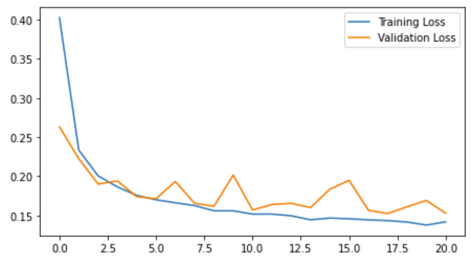
Appendix C

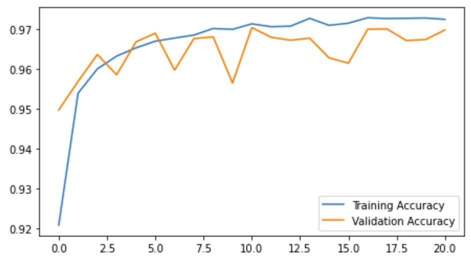
Experiment 4 result

```
Classification Report
              precision    recall  f1-score   support

           0       0.98      0.98      0.98       980
           1       0.99      0.99      0.99      1135
           2       0.98      0.94      0.96      1032
           3       0.92      0.99      0.95      1010
           4       0.98      0.98      0.98       982
           5       0.97      0.96      0.96       892
           6       0.97      0.98      0.97       958
           7       0.97      0.97      0.97      1028
           8       0.96      0.96      0.96       974
           9       0.98      0.96      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: 0.9705
Root Mean Square Error: 0.7168681887209113
```



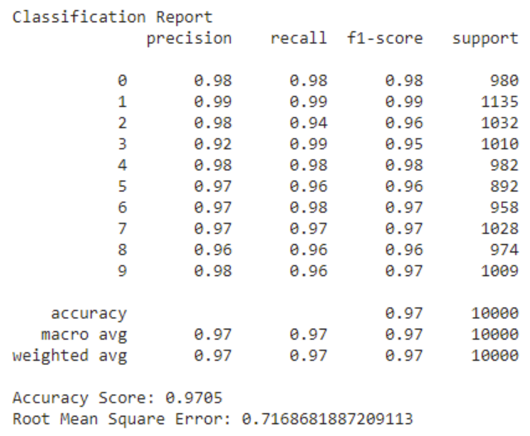
	0	1	2	3	4	5	6	7	8	9
0	959	0	1	1	0	8	6	1	4	0
1	0	1118	1	7	0	1	2	1	5	0
2	5	1	970	22	2	1	5	8	18	0
3	1	0	0	997	0	4	0	5	3	0
4	1	0	4	1	959	2	7	0	1	7
5	2	0	0	17	1	860	5	1	4	2
6	4	3	0	1	2	6	939	1	2	0
7	0	1	8	11	2	0	0	996	3	7
8	1	0	2	19	4	4	5	1	937	1
9	2	2	0	10	7	5	3	8	2	970



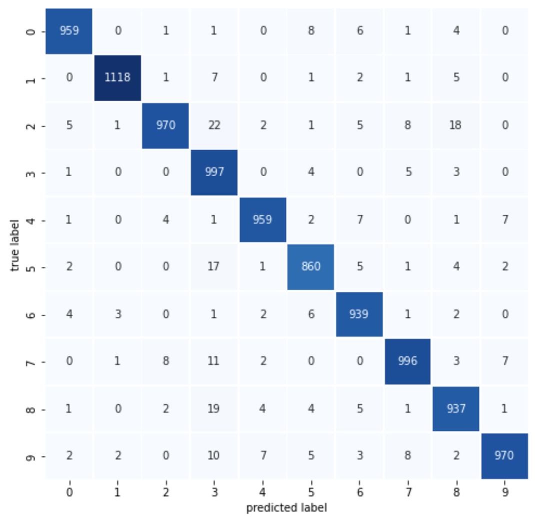


Appendix D

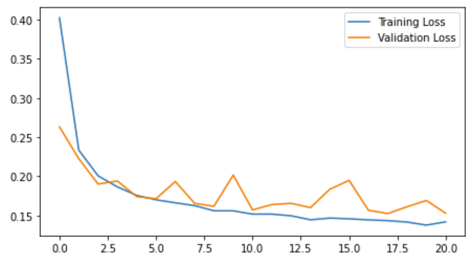
Experiment 5 result

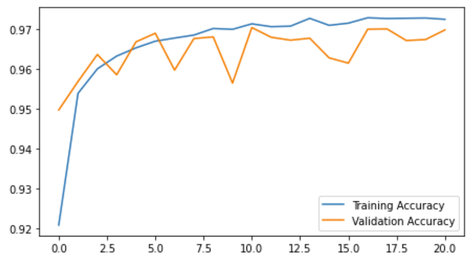


```
Classification Report
              precision    recall  f1-score   support

           0       0.98      0.98      0.98       980
           1       0.99      0.99      0.99      1135
           2       0.98      0.94      0.96      1032
           3       0.92      0.99      0.95      1010
           4       0.98      0.98      0.98       982
           5       0.97      0.96      0.96       892
           6       0.97      0.98      0.97       958
           7       0.97      0.97      0.97      1028
           8       0.96      0.96      0.96       974
           9       0.98      0.96      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: 0.9705
Root Mean Square Error: 0.7168681887209113
```



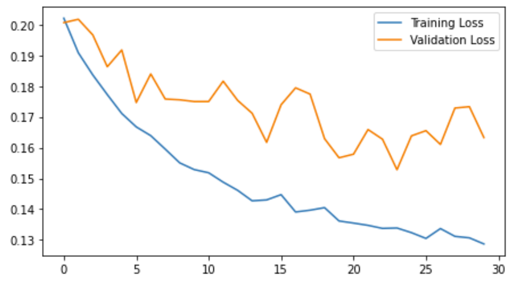
	0	1	2	3	4	5	6	7	8	9
0	959	0	1	1	0	8	6	1	4	0
1	0	1118	1	7	0	1	2	1	5	0
2	5	1	970	22	2	1	5	8	18	0
3	1	0	0	997	0	4	0	5	3	0
4	1	0	4	1	959	2	7	0	1	7
5	2	0	0	17	1	860	5	1	4	2
6	4	3	0	1	2	6	939	1	2	0
7	0	1	8	11	2	0	0	996	3	7
8	1	0	2	19	4	4	5	1	937	1
9	2	2	0	10	7	5	3	8	2	970

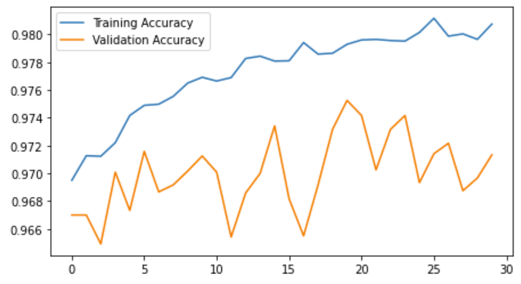


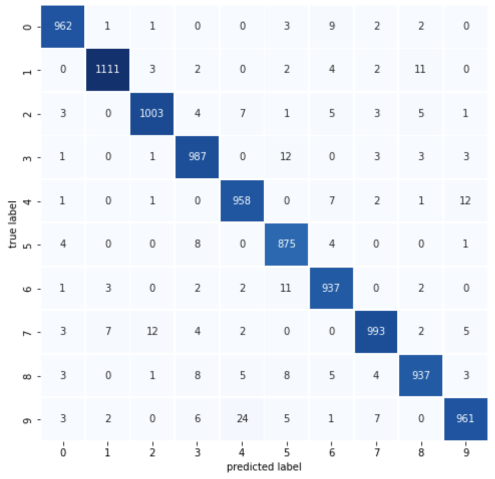


Appendix E

Experiment 6







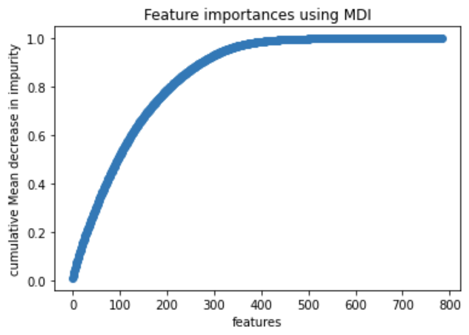
	0	1	2	3	4	5	6	7	8	9
0	962	1	1	0	0	3	9	2	2	0
1	0	1111	3	2	0	2	4	2	11	0
2	3	0	1003	4	7	1	5	3	5	1
3	1	0	1	987	0	12	0	3	3	3
4	1	0	1	0	958	0	7	2	1	12
5	4	0	0	8	0	875	4	0	0	1
6	1	3	0	2	2	11	937	0	2	0
7	3	7	12	4	2	0	0	993	2	5
8	3	0	1	8	5	8	5	4	937	3
9	3	2	0	6	24	5	1	7	0	961

```
Classification Report
              precision    recall  f1-score   support

           0       0.98      0.98      0.98       980
           1       0.99      0.98      0.98      1135
           2       0.98      0.97      0.98      1032
           3       0.97      0.98      0.97      1010
           4       0.96      0.98      0.97       982
           5       0.95      0.98      0.97       892
           6       0.96      0.98      0.97       958
           7       0.98      0.97      0.97      1028
           8       0.97      0.96      0.97       974
           9       0.97      0.95      0.96      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: 0.9724
Root Mean Square Error: 0.7425631286294789
```

Appendix F

Experiment 8

Using 322 features provided aproximate 96% accuracy on the test data.

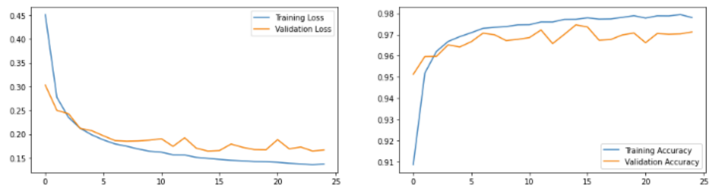


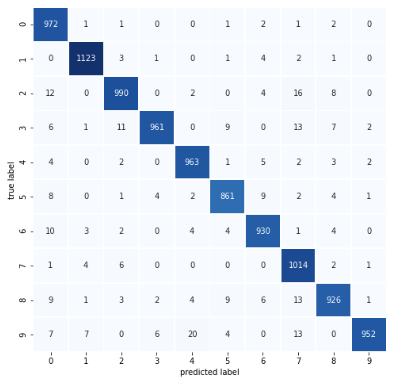
```
Classification Report
              precision    recall  f1-score   support

           0       0.93      0.97      0.95       980
           1       0.97      0.99      0.98      1135
           2       0.91      0.91      0.91      1032
           3       0.96      0.89      0.92      1010
           4       0.90      0.96      0.93       982
           5       0.88      0.90      0.89       892
           6       0.95      0.92      0.94       958
           7       0.95      0.90      0.93      1028
           8       0.89      0.94      0.92       974
           9       0.93      0.88      0.91      1009

    accuracy                           0.93     10000
   macro avg       0.93      0.93      0.93     10000
weighted avg       0.93      0.93      0.93     10000

Accuracy Score: 0.9268
Root Mean Square Error: 1.1251666543228163
```

Appendix G

Experiment 9

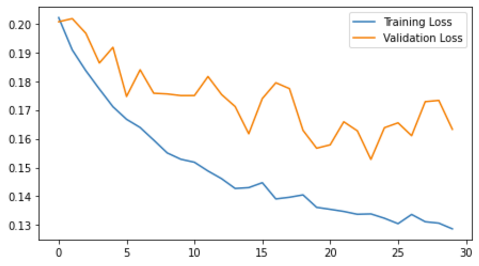
```
Classification Report
              precision    recall  f1-score   support

           0       0.97      0.99      0.98       980
           1       0.99      0.99      0.99      1135
           2       0.94      0.98      0.96      1032
           3       0.95      0.98      0.97      1010
           4       0.98      0.97      0.97       982
           5       0.99      0.96      0.97       892
           6       0.98      0.99      0.98       958
           7       0.98      0.96      0.97      1028
           8       0.99      0.95      0.97       974
           9       0.96      0.97      0.96      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: 0.973
Root Mean Square Error: 0.7370210309075311
```

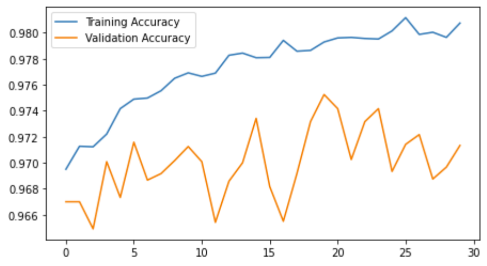


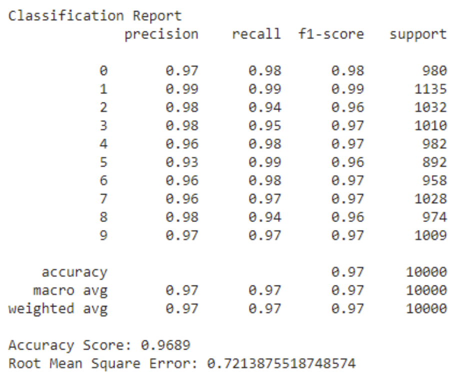


Appendix H

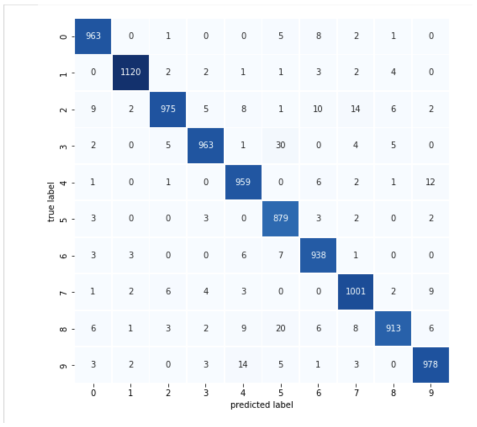
Experiment 8







```
Classification Report
              precision    recall  f1-score   support

           0       0.97      0.98      0.98       980
           1       0.99      0.99      0.99      1135
           2       0.98      0.94      0.96      1032
           3       0.98      0.95      0.97      1010
           4       0.96      0.98      0.97       982
           5       0.93      0.99      0.96       892
           6       0.96      0.98      0.97       958
           7       0.96      0.97      0.97      1028
           8       0.98      0.94      0.96       974
           9       0.97      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: 0.9689
Root Mean Square Error: 0.7213875518748574
```


true label
predicted label
0: 963 0 1 0 0 5 8 2 1 0
1: 0 1120 2 2 1 1 3 2 4 0
2: 9 2 975 5 8 1 10 14 6 2
3: 2 0 5 963 1 30 0 4 5 0
4: 1 0 1 0 959 0 6 2 1 12
5: 3 0 0 3 0 879 3 2 0 2
6: 3 3 0 0 6 7 938 1 0 0
7: 1 2 6 4 3 0 0 1001 2 9
8: 6 1 3 2 9 20 6 8 913 6
9: 3 2 0 3 14 5 1 3 0 978