Assignment 1: Computer Vision

Vivian Xia

Northwestern University

MSDS458: Artificial Intelligence & Deep Learning

Syamala Srinivasan

January 23, 2022

Abstract

Machine learning modeling is a highly data-driven computer algorithm that analyzes patterns in the data that cannot be easily derived by humans. These networks find those features to help predict the output of similar input data. One such powerful tool is deep neural networks that use at least one hidden layer within the network to transform the inputs into those important features.

An application of deep neural networks is in computer vision. To understand and implement a neural network in this application, this project will experiment using the MNIST data, a dataset of gray-scale handwritten digit images and its digit labels, from TensorFlow. The dataset provides a classification problem where each image should be classified to its corresponding class. A neural network will be built and revised using various techniques to observe and interpret the impact of hyperparameters, which, in turn, will help to evaluate which model best classifies these images.

Introduction

Neural networks in computer vision have been applied in many fields for practical use and research. It has been applied in fields such as weather forecasting, traffic detection, cancer diagnosis, human behavior, score keeping in games, and more (Khan & Al-Habsi, 2020). Because of the increased availability of data and computing power, the use of neural networks has become more prevalent, so it important to understand not only how to build a model but how it works to be able to interpret the workings of the structure and dictate what it is doing and its results.

This project looks to explore the neural nets and how these structures work by building multiple dense neural networks for a classification problem. The focus will be on how the activating values from the hidden nodes produce important features used to discriminate between the classes. This factors into analyzing and visualizing the results to see how well each model did. Dimensional reduction is another useful technique that will be explored to build more efficient networks.

The data used will be a public dataset from TensorFlow that can be imported in from the package. The dataset will contain 70,000 handwritten images of digits and its labels. This project will contain ten experiments that analyze the performance of dense neural networks with one hidden layer. The accuracy scores will be recorded for each experiment to pick the best model among the ten. The performance metric visualizations on loss and accuracy will also help to confirm the model's performance and if it is overfitting or not. Other visualizations will be created to assess the accuracy of the digit classifications.

Literature Review

Computer vision can be approached through supervised, unsupervised, or semi-supervised learning. Among these approaches, computer vision classification problems are commonly modelled with support vector machines SVM or neural networks (Khan & Al-Habsi, 2020). SVM is a supervised learning model that is often used with multivariate data. It is resilient to noisy data samples and resilient towards overfitting (Hoang et al., 2021). Neural networks use nodes connected by layers to discriminate against each class. Convolutional neural network, CNN, is a type of neural network that is often used for classification problems as well. CNN uses neurons with dimensions to predict the labels (Khan & Al-Habsi, 2020). Other

models that have been used for classification problems are Relevance Vector Machine, Random Forest Model, Classification Tree Models (Hoang et al., 2021).

The process of building a model starts with image acquisition then image preprocessing. Feature enhancement and extraction are common techniques used to aid in the detection of objects (Khan & Al-Habsi, 2020). Feature extraction of important features in the pixels have helped with discriminating one class from another. Extraction can include the consideration of pixel intensity, gray-level cooccurrence, texture descriptors, etc. Model classifiers are often integrated with other methods such as segmentation, extraction, and filtering to best deliver accurate predictions of class labels (Hoang et al., 2021).

Methods

The approach to this project is to use multilayer perception model with one hidden layer and varying number of nodes. To process the data and build and analyze the network and its results, the following packages were imported: packaging, numpy, pandas, collections, tabulate, tensorflow, keras, and sklearn.

The package packaging was imported to check and verify the version of the tensorflow and keras package needed is imported. The packages numpy and pandas were imported for formatting data purposes. The collections package was used for data exploration purposes and tabulate was used to make results more readable. With the intent of creating neural network models, tensorflow along with its package keras was used. The keras package includes the MNIST dataset that will be used as the input for the model. Keras is also used to build, compile, and train the network. To visualize the accuracy and classification of the model, various sklearn packages as well as seaborn and matplotlib were imported.

After importing the packages, the data was loaded in and explored. The shape of the images in the training and testing set was observed to be 60,000 28 by 28 images and 10,000 28 by 28 images. The training and testing set labels consisted of digit labels that ranged from 0 to 9. In Figure 1 and 2, the number of samples for each label varied, but the frequency skewness is not too high. If the frequency were highly skewed, the data would need to be balanced out. The lowest frequency was 5,421 for the training and 892 for the testing set labels for the digit 5, which is still sufficient for digit 5 to be trained and tested on. To explore what the data looks like, fifty of the sample data were visualized with the image and its corresponding label. As seen in Figure 3, this visualization showed a range of how varied each digit can be written (Jensen, 2022).

The images in the data are two-dimensional 28 by 28 pixels, but they need to be flattened into one-dimensional arrays of 784 with a float data type. The pixels in each image range from 0, white, to 255, black. The pixels can be normalized by dividing 255 to rescale the values to be between 0 and 1, helping the models train faster (Jensen, 2022).

A validation set was created by setting aside 5,000 of the training set images and labels. The network is then built using a Sequential class with two Dense layers, so that the nodes are connected to the preceding nodes. There is one hidden layer with an activation function of ReLu. The number of nodes within this layer will be experimented with to find the most efficient and accurate model. The output of this layer and the hidden nodes are the activation values, otherwise known as the activated neurons or features, that the model will use to predict. The second Dense layer is the output layer and consists of 10 nodes for the 10 digits. The activation function will be Softmax, which will return a value for each digit between 0 and 1, representing the probability of that image belonging to that class. The model is compiled using the RMSprop

optimizer and Sparse Categorical Cross entropy loss function. The optimizer helps to minimize the loss during model training by deciding how much to change each parameter via stochastic gradient descent. The loss function measures the difference between the actual output and the desired output. The model is trained on the 55,000 training set images and labels and validated on the 5,000 validation set data. An early stopping callback is added to stop training with the validation accuracy no longer improves after 2 epochs to keep the model from overfitting and be more efficient. The model is then evaluated on the 10,000 test set data (Jensen, 2022).

Training and validation loss and accuracy are visualized using matplotlib to analyze the model performance. Other visualizations such as confusion matrices, boxplots, scatter plots, t-Distributed stochastic neighboring embeddings on the nodes to plot a T-distribution curve are also used to visualize the performance of different versions of the model during experimentation. With experimenting, techniques such as dimensionality reduction using Random Forest Classifier and Principal Component Analysis are implemented improve the model's efficiency.

Results

Experiment 1

The first experiment included a model with two Dense layers. The hidden layer consisted of one node with the activation function ReLu and an output layer had 10 nodes with Softmax, resulting in a total of 805 parameters. It used a RMSprop optimizer and Sparse Categorical Cross entropy loss function. It was trained on the training set and validated on the validation set. This experiment ran for 14 epochs. The early stopping callback parameter helped to make the model more efficient so that it only needed to run through the 805 parameters 14 times rather than the

50 times that was set. This callback will be used throughout all the experiments, especially as more nodes are added to the models.

The training, validation, and test accuracies were recorded in Figure 19. The accuracy values for all three sets are around the 0.40, so the model does not seem to be overfitting. The test accuracy score is not very high, so the model does not do well in predicting the test set. However, an accuracy score of 0.40 is still significantly better than predicting randomly at 0.10. With only one node in the hidden layer, the network improves its accuracy by approximately four times that of randomly guessing (Srinivasan, 2022).

The loss and accuracy of the training and validation set is visualized in Figure 4. The training and validation loss and accuracy graphs are moving in the same direction and do not seem to have a big difference, so overfitting is not an issue for this model. The training and validation loss graphs are also going in the down direction, which shows that loss is being minimized in the model. However, the loss is still very high and not near 0. The accuracy of the training and validation are going up but not close to 1 at all. These two graphs show that the model is underfitted.

A confusion matrix, Figure 5, was constructed to evaluate the classification performance of the model on the test set. The matrix compares the predicted labels to the actual or true labels, showing the classes that get classified correctly and misclassified. If all the images are classified correctly, only the diagonal cells would be colored white (Srinivasan, 2022). The error rate can be observed from the confusion matrix. One such notable error rate is that 83% of the 0's is misclassified as 6's. From Figure 6, the images of the 0's that were classified as 6's are visualized as well as 0's classified as 0's, 6's classified as 6's, and 6's classified as 0's. The 0's classified as 6's do not look like the correctly classified 6's visualized in the plot. They look very

similar to the correctly classified 0's. The 6's classified as 0's also do not look like 0's. This visualization supports that the model is underfitted.

Another visualization, a boxplot, is built using the activation values of the hidden node on the y-axis and the corresponding predicted classes on the x-axis. The ideal boxplot should have no overlap. However, there is overlap in this boxplot, Figure 7, since the model is not that accurate and is unable to discriminate between the classes. This model underfit the data, so more nodes should be used.

Experiment 2

Experiment 1's model with one node was not able to discriminate all the classes very accurately and underfitted the data. Experiment 2 will use the same network architecture of Experiment 1 but use 2 hidden nodes instead of 1, resulting in 1,600 parameters. The accuracy of the training, validation, and testing set are all around the same value of 0.66 as seen in Figure 50. The use of two nodes does a better job discriminating between classes than the use of one node. The model is not very good but does better than Experiment 1's model and a random prediction. With two nodes, the model was about to predict six times as well as a random prediction.

The performance metrics of training and validation loss and accuracy are visualized. The difference between training and validation graphs are small and they both go in the same direction, so overfitting is not an issue. The loss graphs are going in the right direction but are still large since the values are not close to 0. The loss values are less than that of Experiment 1's which is an improvement. The accuracy graphs are improving with each additional epoch, but the accuracy is still nowhere close to 1. The accuracy values scores are also better than Experiment 1's. The accuracy scores and plots show that the model is underfitted.

The confusion matrix visual, in Figure 8, shows that this model did a much better job classifying images correctly than Experiment 1's model. The diagonal is colored lighter than the rest of the matrix while Experiment 1's matrix had lighter cells distributed all around the matrix. This matrix had a clear diagonal. This model did a great job classifying 1's correctly. It did still misclassify a few digits including misclassifying 4's as 9's and 8's as 5's. From Figure 9, most of the misclassified 4's does not look like 9's. There are one or two 4's that does look like a 9, so it is understandable that this model cannot easily decipher between the two digits as well. Similarly, the 8's that were misclassified as 5's do not look like 5's except one or two of them as well.

A scatterplot, Figure 10, was created to visualize the activation values of the two hidden nodes, on the axes, on the predicted classes. The plot shows that the two nodes can discriminate the 10 classes from one another. Each color represents a class, and the graph shows 10 separate clusters. The clusters still do overlap, but there is discrimination. The model did a better job than Experiment 1, but it underfit the data.

Experiment 3

Experiment 1 and 2's models underfit the data, resulting in low accuracy and high loss scores. For Experiment 3, the same architecture will be used as Experiment 1 and 2's, but this model will use 100 hidden nodes. This network will have 79,510 parameters. The training, validation, and test accuracy are all around 0.98, which is a very good score.

The performance plots, Figure 11, are visualized to evaluate the loss and accuracy of training and validation sets. The loss plot shows that the training and validation graphs going down, which shows that the loss is decreasing. But the validation graph begins to flatten out while the training graph continues to decrease, so the model may be overfitting the data.

Similarly, the accuracy plot shows that the training graph continues to increase while the validation graph starts to flatten out, indicating overfitting. The direction of the validation and training graphs looks like they are starting to go in the opposite directions in the last two epochs.

The confusion matrix, Figure 12, has a very clear diagonal. The cells along the diagonal are all white and the surrounding cells are black, showing that this model does a very good job in accurately classifying the test images with its corresponding label. The error rates are also very small at 0.01 or 0.02. The correctly classified rates for each class are very high ranging from 0.97 to 0.99. Most digits were classified correctly. The number of misclassified images is less than Experiment 1 and 2's.

Another visualization, Figure 13, that can plot the activation nodes with each other to see the corresponding predicted classes uses t-score stochastic neighboring embedding to reduce the 100 features to 2 dimensions. This will allow the distance between each node to be plotted on a one-dimensional line on a t-distribution curve. The resulting plot shows 10 clear clusters representing the 10 classes. The 100 features do a good job discriminating the 10 classes from one another. There is a little overlap of some misclassified images as seen by, for example, green dots in the orange cluster, but the model, overall, did a good job. This model was more accurate than the models of Experiment 1 and 2.

Experiment 4

Because the model from Experiment 3 may have overfit the data, this experiment will reduce the number of nodes from 100 to 50. Otherwise, it will have the same network architecture as that of Experiment 3. Despite half the number of nodes than Experiment 3, the train, validation, and test accuracy scores are about 0.97. The accuracy scores are very high.

The loss plot shows that the training and validation loss is very small. The validation graph does seem to flatten out while the training graph continues to fall. The accuracy for both training and validation is very high. Like the loss plot, the validation graph flattens out while the training graph continues to increase. However, the scale for the plots shows that the difference between the validation and training is very small, so there may not be overfitting.

The confusion matrix, Figure 14, shows a white diagonal down through the matrix. This visualizes that most of the images were classified correctly. The correct classification rate is a little lower than that of the rates seen in Experiment 3's confusion matrix, but this model still did a good job classifying accurately. The error rates are also minimal at 0.01. Looking at the images of the numbers in Figure 15, the 9's misclassified as 7's have a few of the same images as that of Experiment 3's misclassified 9's. Some of the 9's does look like 7's, so it is understandable that it may be hard for the network to classify these images correctly.

The dimensions of the 50 nodes were reduced to 2 by t-SNE to be able to be plotted on a t-distribution curve with the predicted labels. In Figure 16, there are clear clusters representing the different classes, showing that the 50 features can discriminate the 10 classes from one another. There is some overlap of some dots that are in a cluster that is not its corresponding color. Experiment 3's corresponding plot had more distinct and less overlapping clusters, but overall, this model did a good job. This model had half of the nodes that of Experiment 3 but managed to still have around the same accuracy for its test set and could discriminate between the 10 classes well.

Experiment 5

Principal component analysis decomposition is one way to reduce the dimensions in the model. Principal components are a linear combination of all 784 pixels. Instead of using all 784

inputs, PCA can use 154 dimensions that contain 95% of the training images variance or information in those components instead (Srinivasan, 2022).

Before reducing the dimensions, a network with 55 nodes is built, trained, and evaluated. Because Experiment 3's model seemed to do better than Experiment 4's, this model, uses 55 nodes. The accuracy for the training, validation, and testing set are approximately 0.97, which is a high accuracy. The performance metrics plot shows that the loss and accuracy for training are consistent in the same direction as the validation and the difference between the two are minimal.

The network using the reduced dimensions will include 55 hidden nodes as well. The training accuracy is 0.99 and the validation and testing accuracy is 0.97. This is also high accuracy score, which indicates that these 154 components do not negatively impact the model. The accuracy scores of this model and the preceding model are very similar even though this model runs though 9,085 parameters each epoch and the preceding model runs through 43,735 parameters.

The performance plot, Figure 17, shows the loss and accuracy of the training and validation set. The validation loss and accuracy graphs flatten out as the training set continues to decline and incline, respectively. The difference between the accuracy scores is 0.02, which is still small but can also be an indication that there is overfitting. The number of nodes or the amount of information in the principal component should be reduced.

Experiment 6

Experiment 5's model overfit the data with principal components that had 95% of the information and 60 nodes. This experiment will use principal components that has 75% of the information and 60 nodes. The input of the network will consist of 34 dimensions.

The training, validation, and test set accuracy is approximately 0.97. The high accuracy scores are reflected in the accuracy plots as well. The loss plot shows small loss values for both training and validation. The difference between the training and validation graphs are very small, less than 0.03 for accuracy. This model does not seem to overfit.

Experiment 7

This experiment will increase the amount of information from the training images from 75% to 80% in exchange for less hidden nodes. This network will consist of 50 nodes instead of the 55 used in Experiment 5 and 6.

The training, validation, and test accuracy are all around 0.97, which is a high accuracy score. The performance plots show that the training and validation loss and accuracy scores are very close and that there is no overfitting. This model does very similarly to Experiment 6's model, which shows that the amount of information that is inputted can offset the number of nodes in the hidden layer. The method of dimensional reduction can make the model run faster and require less parameters to run through each epoch.

Experiment 8

Another way to reduce dimensions is using a Random Forest Classifier to get the most important features or pixels instead of using all 784 pixels. This model will use only the top 70 pixels. The important 70 pixels can also be visualized as seen in Figure 18.

Similar to Experiment 5, the first network that is evaluated is the model without any dimensional reduction to compare it to the accuracy of the model with the dimensional reduction. This first network has 55 hidden nodes. The training accuracy is 0.99 and validation and test accuracy are 0.97.

The model with the input of the 70 most important features and 55 nodes is trained. The training, validation, and testing accuracy score is approximately 0.92. The accuracy score is good, especially since this model is using about one-eleventh of the number of inputs as the first model. The use of less inputs minimizes the parameters from 43,735 from the first model to 4,465 from this model. The performance loss plot shows that the training and validation graph are minimizing the loss with each epoch but are not exactly close to 0 yet. The accuracy plot shows there the training and validation are going in the right direction but are not close to 1 yet. The number of nodes can be increased or more input pixels can be added to improve the model.

Experiment 9

This experiment increases the number of input pixels from 70 to 100. The network will continue to consist of 55 hidden nodes. The training, validation, and test accuracy is 0.94. The model did better than that of Experiment 8's with the addition of more input pixels.

The performance plots also reflect the accuracy scores. The loss and accuracy plots are going in the right direction but not close enough to 0 and 1 yet, respectively. The number of pixels should be increased to improve accuracy.

Experiment 10

The Random Forest Classifier will be used to take the top 200 pixels. The network will have an input of 200 pixels and 55 hidden nodes. The training and validation accuracy is 0.97. The test accuracy is 0.96. These accuracy scores are better than Experiment 9's. The performance plots are also better than Experiment 9's. The plots approach 0 and 1 more closely for the loss and accuracy than the preceding model's, respectively.

Summary

From the experiments, the "best" model based on test accuracy is Experiment 3's model. The test accuracy is 0.98. But based on the performance of Experiment 4, which used half the number of nodes of Experiment 3's, and had a test accuracy of 0.97, the better model for implementation may be Experiment 4's. There is also a chance that Experiment 3 is overfitting the data since the performance plots show that the validation and training graph start going in opposite directions. Experiment 3 had 79,510 parameters while Experiment 4 had 39,760 parameters, so using Experiment 4 is also the more efficient model.

In comparison to Experiment 4, the models from Experiment 6 and 7 are better. The test accuracy for both models are 0.97, so they had the same performance as Experiment 4 and did a very good job classifying the images correctly. However, the model from Experiment 6 used 2,485 parameters and Experiment 7 used 2,760 parameters. These two experiments used more than ten times less parameters than Experiment 4 while achieving the same accuracy score. These two models also did not overfit.

The experiments using Random Forest Classifier to reduce input pixels overall required more dimensions and, in turn, parameters than the experiments using principal component analysis to reduce dimensions. Experiment 10 used the top 200 pixels and 55 nodes to achieve a 0.96 in test accuracy while Experiment 6 used 34 principal components containing 75% of the training image's variance and 55 nodes achieved a 0.97 in test accuracy. Experiment 10 used 11,615 parameters, which is more than five times that of Experiment 6's. In terms of best model, Experiment 6 is the best option of the ten experiments from this project.

Conclusions

As seen in this classification problem, the use of neural networks is a powerful tool in prediction. The handwritten images were accurately classified at a rate over 0.90 in most experiments. Even the experiments using one hidden node was able to accurately classify the images at 0.40, ten times the accuracy rate of randomly predicting the class. The nodes activate the dimensions to create activated values or features that are able to discriminate the ten classes. The experiments also provided examples of models that underfit and overfit the data, so analyzing the performance of each model is an important step in picking the best model.

This project used methods of dimension reduction using principal component analysis and random forest classifier. These methods proved to be effective in reducing the dimensions yet still achieving a high accuracy score. The efficiency of each model compared to one another should also be considered. Overall, the best model from these ten experiments was Experiment 6 which used principal component analysis to reduce the input dimensions from 784 pixels to 34 components and 55 hidden nodes. The performance test accuracy was high at 0.97 while not overfitting the data as well as used less parameters than other models similar in performance.

References

Hoang, N.-D., Huynh, T.-C., & Tran, V.-D. (2021). Computer Vision-Based Patched and Unpatched Pothole Classification Using Machine Learning Approach Optimized by Forensic-Based Investigation Metaheuristic. *Frontiers in Data-Driven Methods for Understanding, Prediction, and Control of Complex Systems 2021*, *2021*. https://doi.org/https://doi.org/10.1155/2021/3511375

Jensen, D. (2022). *[MSDS 458] Code Review Sync Session* 1 [Zoom Cloud Recordings].

Khan, A. I., & Al-Habsi, S. (2020). Machine Learning in Computer Vision. *Procedia Computer Science*, *167*, 1444–1451. https://doi.org/https://doi.org/10.1016/j.procs.2020.03.355

Srinivasan, S. (2022). *MSDS 458 AI/Deep Learning: Sync Session #2* [Zoom Cloud Recordings].

Appendix

Figure 1. Frequency of each label in training data.

Training Labels	Frequency
1	6742
7	6265
3	6131
2	5958
9	5949
0	5923
6	5918
8	5851
4	5842
5	5421

Figure 2. Frequency of each label in testing set.

Testing Labels	Frequency
1	1135
2	1032
7	1028
3	1010
9	1009
4	982
0	980
8	974
6	958
5	892

Figure 3. Visualization of sample training data and its label.

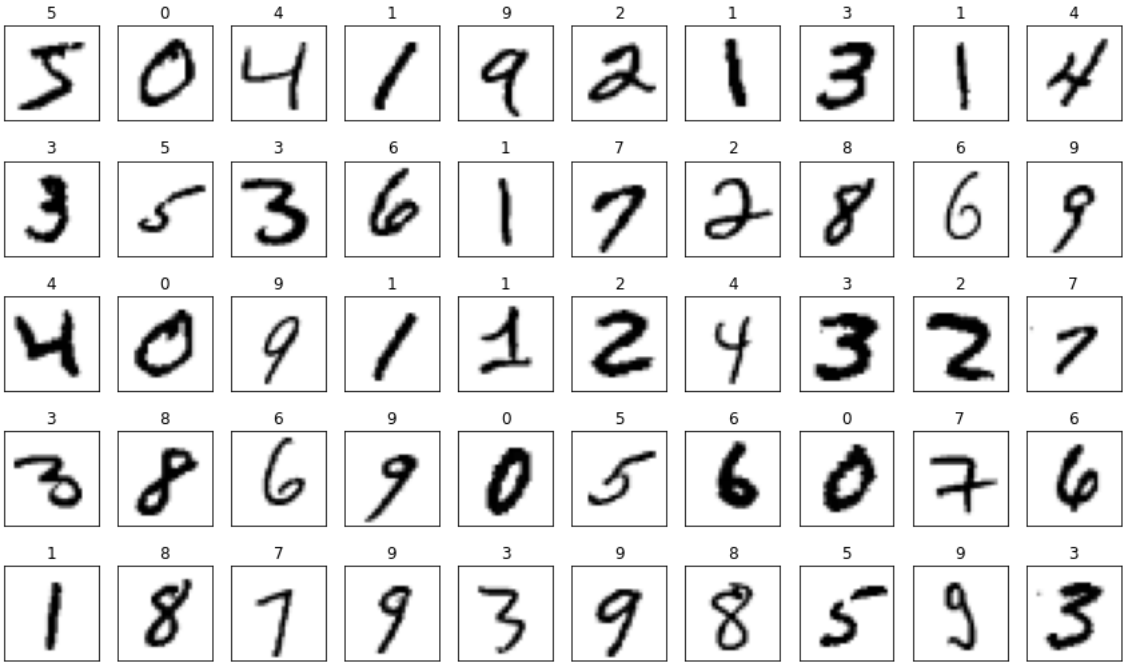

Figure 4. Experiment 1 Performance Metric plots.

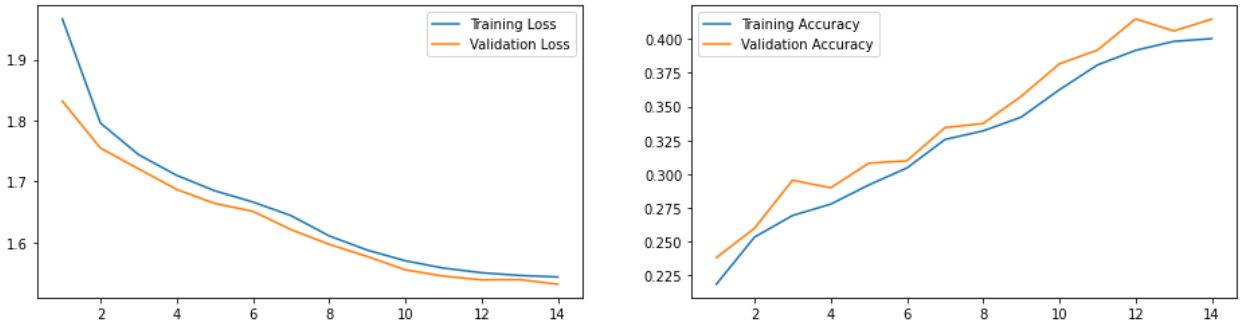

Figure 5. Experiment 1 Confusion Matrix with classification rates.

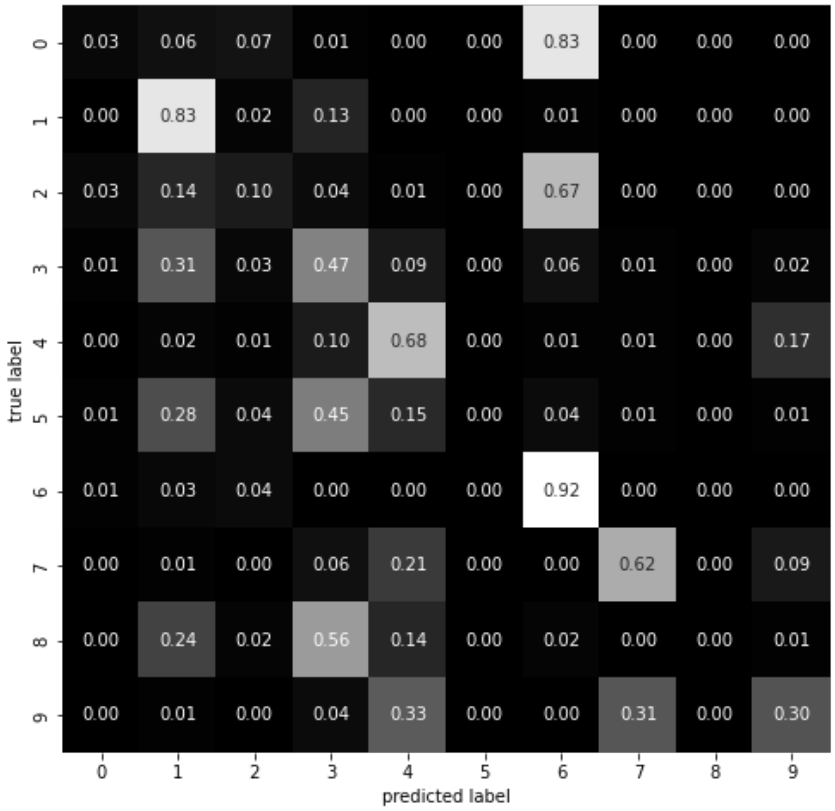

Figure 6. Experiment 1 misclassified images instance.

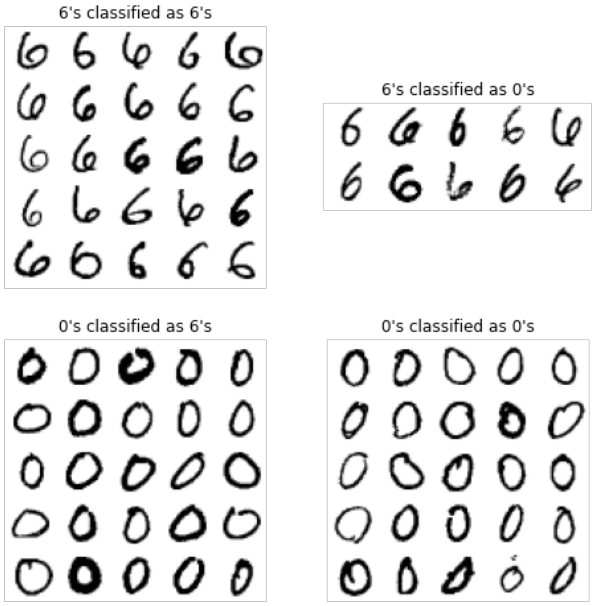

Figure 7. Experiment 1 predicted class vs. activated values boxplot.

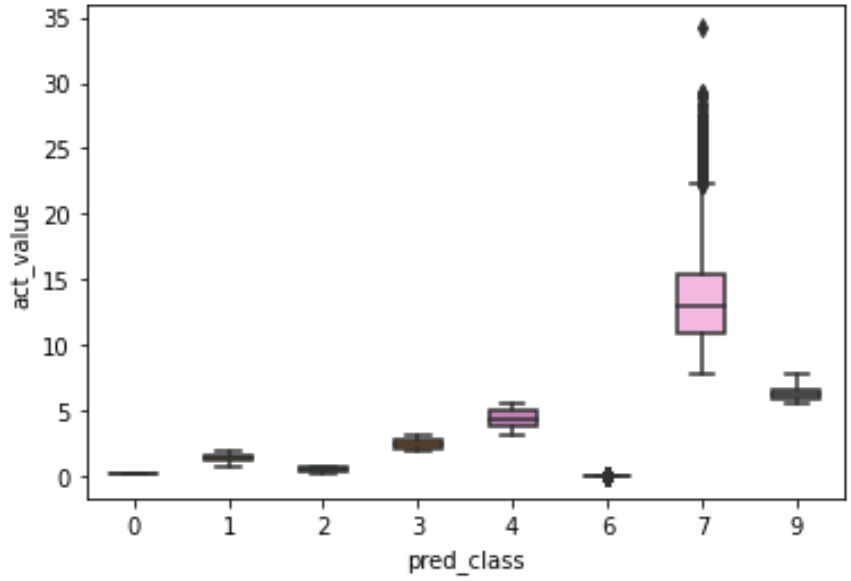

Figure 8. Experiment 2 Confusion Matrix with classification rates.

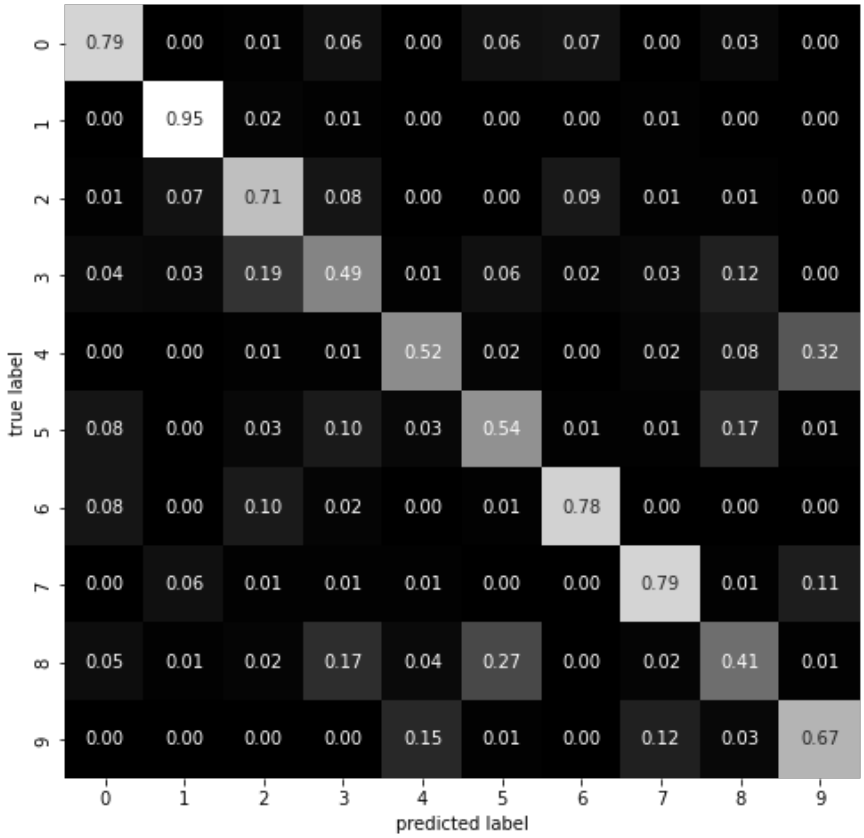

Figure 9. Experiment 2 misclassified images instance.

Figure 10. Experiment 2 scatter plot of activated values.

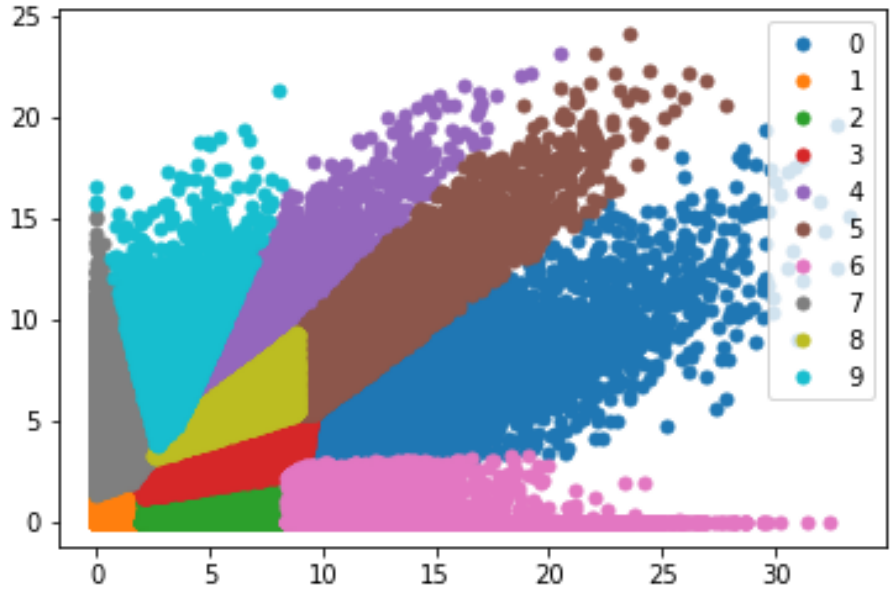

Figure 11. Experiment 3 Performance Metric plots.

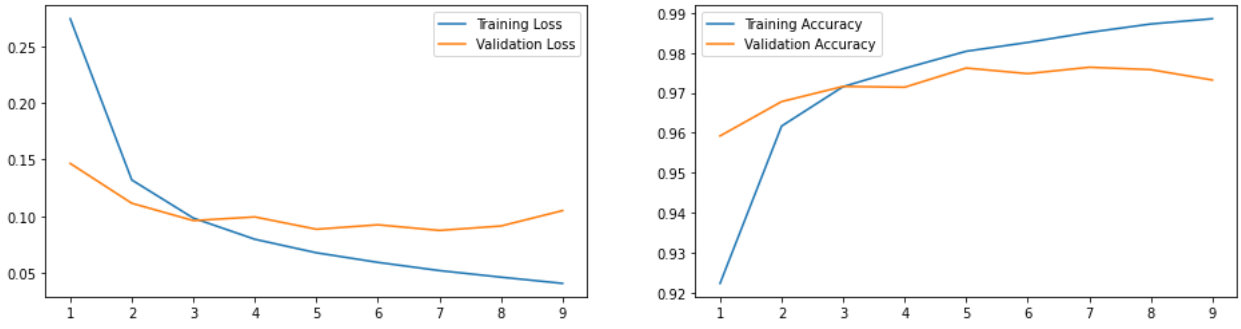

Figure 12. Experiment 3 Confusion Matrix with classification rates.

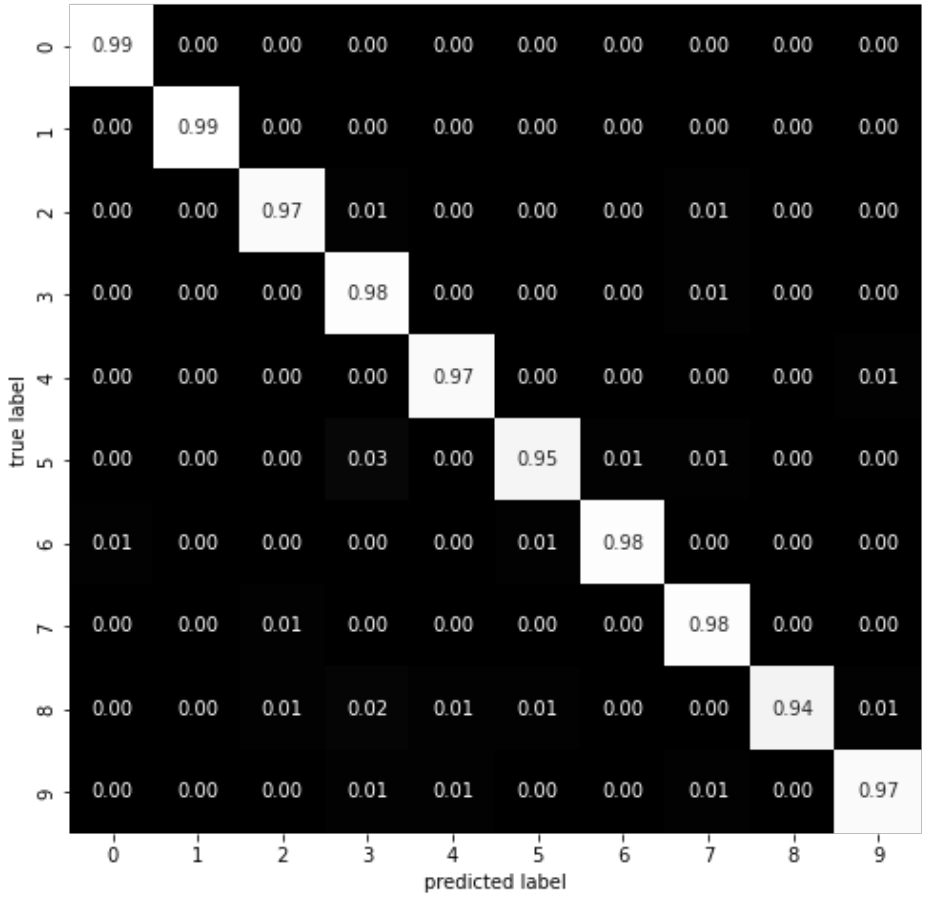

Figure 13. Activated values and predicted labels plotted on a t-distribution curve

Figure 14. Experiment 4 Confusion Matrix with classification rates.

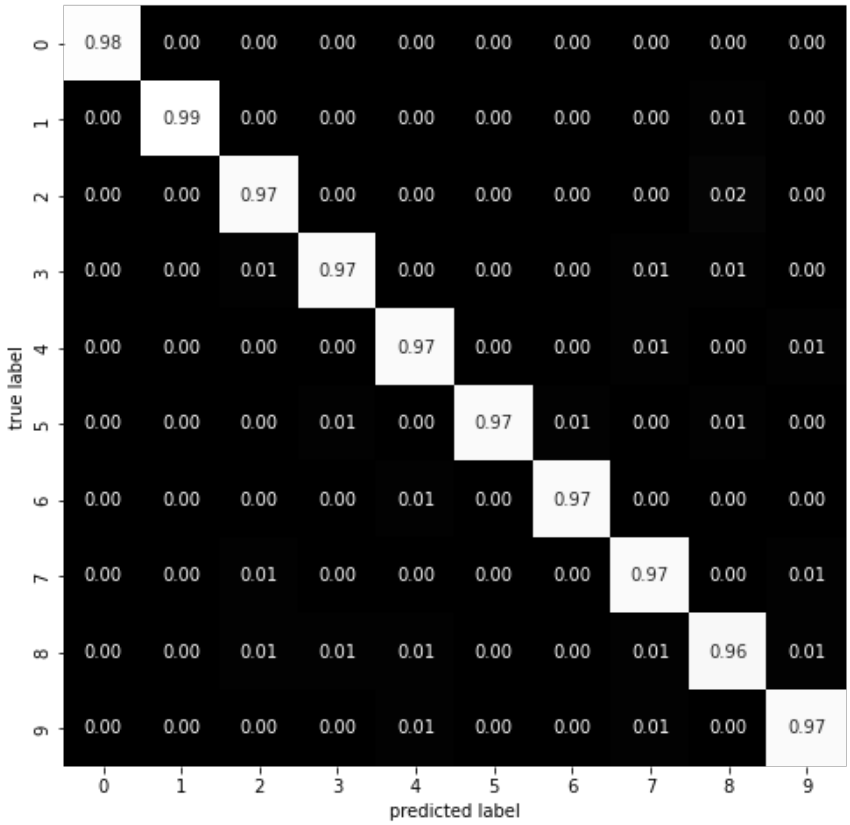

Figure 15. Experiment 3 misclassified images instance.

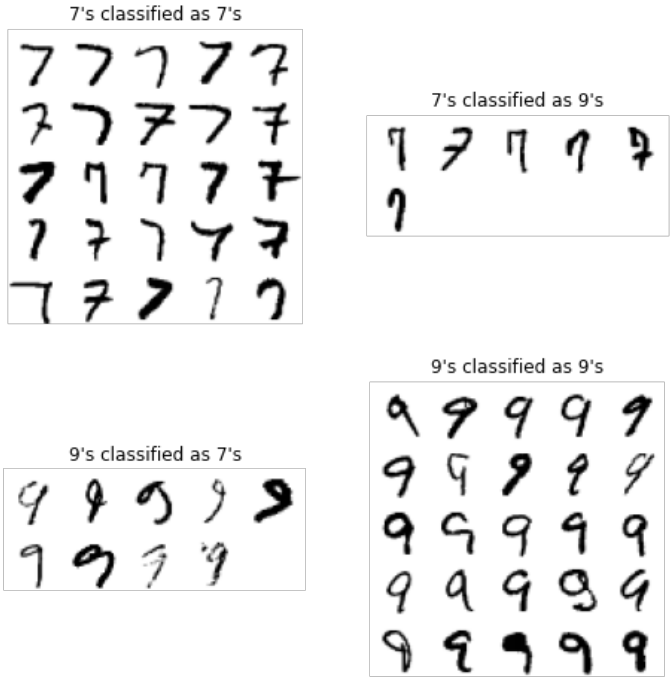

Figure 16. Activated values and predicted labels plotted on a t-distribution curve.

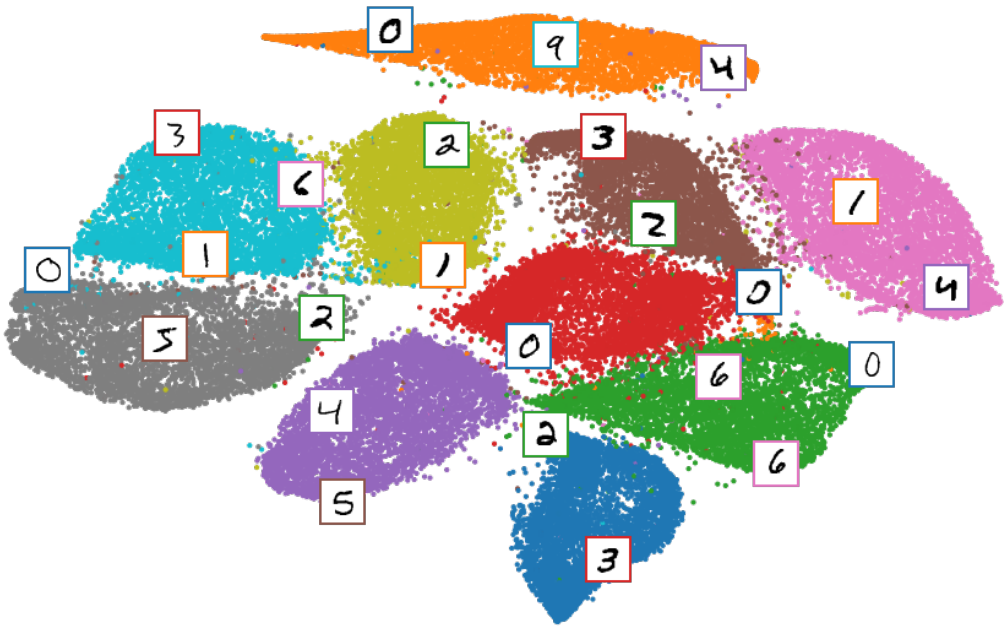

Figure 17. Experiment 5 with principal components Performance Metric plots.

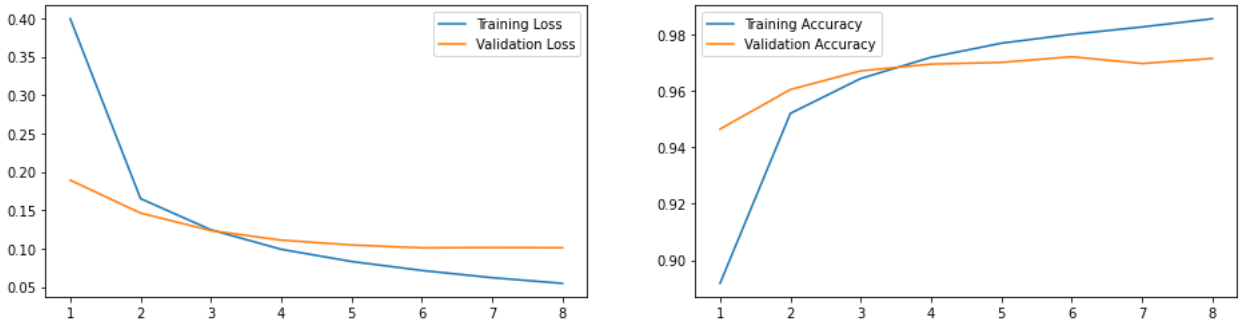

Figure 18. Experiment 8 visualization of the top 70 pixels.

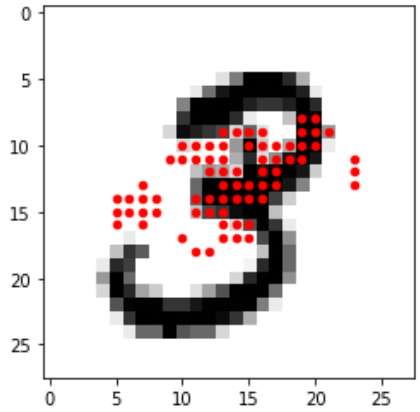

Figure 19. Training, validation, test accuracy scores of all experiments.

Experiment	Description	Training	Validation	Testing
1	1 hidden layer with 1 node	0.40	0.41	0.40
2	1 hidden layer with 2 nodes	0.66	0.67	0.67
3	1 hidden layer with 100 nodes	0.99	0.97	0.98
4	1 hidden layer with 50 nodes	0.98	0.97	0.97
5	PCA with 95% of the information and use 55 nodes	0.99	0.97	0.97
6	PCA with 75% of the information and use 55 nodes	0.97	0.97	0.97
7	PCA with 80% of the information and use 50 nodes	0.98	0.97	0.97
8	Random Forest Classifier reduce input pixels to 70 and use 55 nodes	0.93	0.93	0.93
9	Random Forest Classifier reduce input pixels to 100 and use 55 nodes	0.94	0.94	0.94
10	Random Forest Classifier reduce input pixels to 200 and use 55 nodes	0.97	0.97	0.96