Increasing the Width of a Neural Network Can Provide Significant Improvement on Image Classification and Principal Component Analysis Is an Effective Solution to Reduce Dimensions

Christopher Festa

cfesta@gmail.com
https://www.linkedin.com/in/chris-festa/
+1 (516) 287-7496

Northwestern University

July 9, 2021

Abstract

Neural networks have widely been used by data scientist because they are prominent and perform well on imagine classification datasets. The aim of this study is to show how making nodes in a hidden layer wider can significantly improve the performance of classifying images. A set of experiments was performed on the MNIST dataset, a collection of handwritten images that contain 10 classes of digits (0 to 9) that is widely used to benchmark and test classification algorithms. Results suggest that increasing the width of a hidden layer will perform better than shallow neural networks (e.g., 1 or 2 nodes). Sample comparisons are provided using the results of neural networks that contain [1, 2, 4, 8, 16, 32, 128, 256, 512, 1024] nodes. The study also evaluated the efficacy of using Principal Component Analysis (PCA) to reduce the dimensions of the images and the Random Forest classifier to get the relative importance of features to reduce the number of inputs. Results suggest that PCA is effective and the best solution to reduce the inputs in a neural network.

Keywords: Neural Networks, Hidden Layers, Principal Component Analysis, PCA, Random Forest Classifier, Image Classification

Introduction and Problem Statement

Many factors can impact the results of image classification. One of the most important factors is the architecture of a neural network and the number of nodes in a hidden layer. The number of hidden nodes is defined as the width of a neural network (Rojas 2003). The hidden layer of the neural network must be wide enough so that the result of a classification can be accurate and be able to discriminate between the classes. Too wide can result in overfitting and too little can result in underfitting ("How Many Hidden Units Should I Use?" n.d.). In this study we aim to conduct an investigation into the effects of the number of hidden nodes used in a single-hidden layer neural network using image classification and how they can impact the results of the neural network. This study also looks at principal component analysis (PCA) and the Random Forest classifier to reduce the features in the neural network. PCA reduces the dimensions of the variables, and the Random Forest classifier identifies the most important features. By reducing the dimensions and removing features that are not required to attain an effective model, we can reduce the runtime making it cheaper to train the model and reduce the size of the dataset making it cheaper to store. In much larger image classification datasets, using dimension or feature reduction may be required to train the model due to the high costs of the compute and time required to train the model.

Literature Review

According to Chollet (2018), images typically have three dimensions: height, width, and color depth. This study we use the MNSIT dataset, which is 28x28 and greyscale, therefore the color depth is only a single-color channel. The simplest variant of neural network algorithms is the three-layer backpropagation neural network (Paola and Schowengerdt 1995). The processing node performs two functions. First, the processing node sums the values of its inputs. The sum is

then passed through an activation function to produce the node's output value. There are several activation functions such as sigmoid, rectified linear unit (ReLU), hyperbolic tangent (tanh), scaled exponential linear unit (SeLU), exponential linear unit (ELU), and softmax. The activation functions used in this study was the ReLU function for the input layer and softmax for the output layer. The ReLU function is a piecewise linear function that will output the input directly if it positive, otherwise, it will output zero. The ReLU function overcomes the vanishing gradient problem, allowing models to learn and perform better (Brownlee 2019). The softmax function converts a vector of numbers into a vector of probabilities, where the probabilities of each value are proportional to the relative scale of each value in the vector (Brownlee 2020).

As a neural network trains, it consists of input and output data vectors. The input data vector is the pattern to be learned and the output vector is the desired set of outputs to be produced by the network. The goal of the training is to minimize the overall error between the desired and actual outputs of the network. In order to decrease the error, incremental adjustments in the weights after each iteration is performed based on a gradient descent training procedure called backpropagation. There are several gradient descent optimizers that can be used such as stochastic gradient descent (SGD), root mean square propagation (RMSprop), and adaptive moment estimation (ADAM). The optimizer used in this study was RMSprop. Geoffrey Hinton proposed RMSprop because gradients of very complex functions like neural networks have a tendency to either vanish or explode as the data propagates through the function (Sanghvirajit 2021). RMSprop deals with this issue by using a moving average of squared gradients to normalize the gradient. The normalization balances the step size (momentum), decreasing the step for large gradients to avoid exploding and increasing the step for small gradients to avoid vanishing.

In most artificial intelligence and machine learning problems we have thousands or even millions of features that are used in training. These features make training extremely slow and may make it harder to find a good solution. We call this the curse of dimensionality. Principle Component Analysis (PCA) is the most popular dimensionality reduction algorithm (Géron 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. In this study we preserve 95% of the variance of the MNIST dataset which reduces the 784 features to 154, while preserving the most important information to perform the classification. Alternatively, the study evaluates the feature importance method in the Random Forest classifier. The feature importance can be measured as the average impurity decrease computed from all decision trees in the forest (Kumar 2020).

Data and Methods

The dataset used in this study 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 28x28 grayscale images of the 10 digits, along with a test set of 10,000 images. The study held back 5,000 images from the 60,000 training images for the validation dataset to be used to evaluate the model during training. Figure 1 depicts what an image looks like and the distribution of classes across the training, test, and validation datasets.

To evaluate the effects of the number of hidden nodes used in a single-hidden layer neural network and how they can impact the results, the study conducted three experiments:

1. A dense neural network with 784 input nodes, a hidden layer with 1 node, and 10 output nodes. The hypothesis was to see substantial overlap between the range of values in a boxplot reflecting that the activation values of the hidden node are not able to discriminate between classes.
2. A dense neural network with 784 input nodes, a hidden layer with 2 nodes, and 10 output nodes. The hypothesis was that the activation values also fail to discriminate between the classes.
3. A dense neural network with 784 input nodes, a hidden layer with various widths [1, 2, 4, 8, 16, 64, 128, 256, 512, 1024], and 10 output nodes. The hypothesis was as we increase the width with more nodes the model will do better at discriminating between the classes and improve the accuracy and loss of the model.

The study also evaluated the efficacy of using Principal Component Analysis (PCA) and the Random Forest classifier to reduce the dimensions and features by performing two experiments:

4. Executing PCA decomposition on the training set of 28x28 dimensional MNIST images, generating principal components that represent 95 percent of the variability in the explanatory variables. Using PCA will reduce the number of dimensions from 784 to 154 and the hypothesis was that the dimension reduction was viable to use and did not impact the performance of the model using the best model from experiment three.
5. Using the Random Forest classifier to get the relative importance of the 784 features (pixels) of the 28x28 dimensional images in the training set of MINST images. The study selected the top 70 features and hypothesized that feature reduction is also viable.

All models in the study used the following hyperparameters:

- Hidden layer activation function = ReLU
- Output layer activation function = softmax
- Gradient descent optimizer = RMSprop
- Measurement of loss between labels and predictions = categorical crossentropy
- Metrics to evaluate the performance of the model = accuracy
- Early Stopping with a patience of 25; if the model does not improve after 25 epochs the model will stop training

A final experiment was performed that evaluated the performance of increasing the number of hidden layers and the number of nodes in each layer. The following architectures were tested:

- Two hidden layers with 10 nodes in each layer
- Two hidden layers with 20 nodes in each layer
- Two hidden layers with 150 nodes in the first and 50 in the second layer
- Five hidden layers with 10 nodes in each layer
- Five hidden layers with 20 nodes in each layer
- Five hidden layers with 300 in the first, 200 in the second, 100 in the third, 50 in the fourth, and 25 in the fifth layer

Results and Analysis

The code used in this study can be found on GitHub[1]. The results of first experiment with 1 node in the hidden layer confirmed the hypothesis. As we can see in Figure 2, the boxplot shows substantial overlap between the range of values in the boxes reflecting the fact that the

[1] GitHub repository for the code that performed the analysis used in this study: https://github.com/chrisfesta/NWU_MSDS458

activation values of the hidden node are not able to discriminate between the classes. We can also see in Figure 3 that the model is underfitting and is not complex enough to accurately capture relationships between the features and the target classes. Using 1 node in the hidden layer is only able to achieve approximately 40% accuracy with a loss of 1.6.

The second experiment with 2 nodes in the hidden layer had similar results. In Figure 4, the scatter plot shows substantial overlap between the range of values in the boxes reflecting the fact that the activation values of the hidden node are not able to discriminate between the classes similar to experiment one. We can also see in Figure 5 that the model is performing better than experiment one but is still underfitting. Using 2 nodes in the hidden layer is able to achieve approximately 69% accuracy with a loss of 0.98. While the second experiment performs better than the first experiment there is still room for improvement. The confusion matrix in Figure 6 of both experiments highlights that both models are confusing the actual versus predicted classes such as the digits 8 and 5. Experiment one is correct 12% and experiment two is correct 48% for digit 8 and 11% and 43% for digit 5. The f1 score for both of these digits are also low with 0.17 for both digits 5 and 8 in experiment one and 0.46 and 0.48 in experiment two. The results show that increasing the width of the hidden layer by adding more nodes improves the performance of the model and in experiment three we see that the performance improves even more.

The third experiment tested expanding the nodes by doubling the nodes in a single hidden layer after each trial for 11 trials [1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024]. As the nodes increased the performance of the model improved. Figure 7 shows the results of the trials and the best model had 1024 nodes. However, using 128, 256, or 512 nodes would also be a model that is effective. Figure 8 confirms that the confusion matrix using a neural network with 1024 nodes

in a single hidden layer is able to correctly predict the classes and the f1 score for each class was almost perfect with an average of 0.98.

The MNIST dataset is a small dataset where we can conduct experiments with relative ease and not pay the costs of long training times and expensive GPUs. This is not the case for most image classification problems, therefore dimension and feature reduction become an important step to avoid the costs. In experiment four and five we tested the impacts of using Principal Component Analysis (PCA) to reduce the dimensions and the Random Forest classifier to get the relative importance of the 784 features and reduce the features to the 70 most important features. As we can see in Figure 9, the PCA explained ratio peaks at about 155 features and the Random Forest classifier identified that the most important features are more toward the center of the image. Therefore, we can assume we can reduce the inputs into the model and the performance will be similar.

Figure 10 confirms the hypothesis for PCA dimension reduction; the best model from experiment three with 1024 nodes in a single hidden layer using PCA was able to achieve an accuracy of 98.43%, the confusion matrix had good ratios of predicted vs actual classes, and the classification report maintained an average f1 score of 0.98. Figure 11 mostly confirms the hypothesis using the Random Forest classifier to identify the most important features. Using the same model architecture as the PCA test, the Random Forest classifier's 70 most important features was able to achieve an accuracy of 94.97%, the confusion matrix had good ratios of predicted vs actual classes, and the classification report had f1 scores as low as 0.92 for the digit 5 and as high as 0.99 for the digit 1 with an average of 0.95 across all digits. Based on the test we can conclude the PCA dimension reduction is a viable method to reduce the cost of training while maintaining similar performance. While the Random Forest classifier performed well, the

average f1 score was 0.95 versus 0.98 using the same model architecture and it did not preserve the same performance as PCA. Therefore, PCA should be the preferred method.

Finally, experiment six evaluated the performance of using model architectures with more than one hidden layer and various widths of each layer. Figure 12 shows the results and the best model used 5 hidden layers with [300, 200, 100, 50, 25] nodes in each layer. The model was able to achieve a 98.34% accuracy and a loss of 0.3087. The model was comparable to the best model from experiment three and experiment four with PCA dimension reduction. Therefore, increasing the layers with various widths does not provide tangible improvement to the performance.

Conclusion

In this study, six experiments of image data using the MNIST dataset were examined and compared the results of different model architectures at various widths. The width of models must be wide enough so the result of the classification will be accurate and be able to discriminate between the classes. Experiment one and two showed the width was not wide enough and the models were not able to discriminate between classes causing the model to underfit. Experiment three tested different model widths and the best model had 1024 nodes versus 1 and 2 in the first two experiments. The wider model performed well with a 98.45% accuracy and an average f1 score of 0.98. The study also explored the efficacy of reducing the dimensions with Principal Component Analysis (PCA) and the number of features using the Random Forest classifier. PCA proved effective in preserving model performance compared to the model that did not reduce the dimensions achieving the same accuracy of 98% and an average f1 score of 0.98. Using the Random Forest classifier performed well with a 95% accuracy and average f1 score of 0.95 but did not perform as well as PCA. Finally, adding more

hidden layers did not improve the performance compared to the best model with a single hidden layer with 1024 nodes. Therefore, we can conclude that increasing the width of a single hidden layer neural network improves the performance and reducing the dimensions with PCA is the preferred method to reduce the cost of training neural networks.

Future Work

Several open questions remain. First, the study was limited to the MNIST dataset. How would the performance of different datasets be impacted if we had pictures such as dogs and cats? If the original images were not greyscale and had more color depths would the findings of the study remain valid? In general, adding more hidden layers and increasing the nodes in the architecture will improve model performance and performing similar tests in this study should be completed on different datasets to confirm the findings of this study. Finally, the study did not consider convolutional neural networks (CNN). CNNs are considered the best model architectures to use in image recognition (Maladkar 2018). Will evaluating CNN change the findings of this study? A future study would require evaluating different datasets and moving this study to the cloud where graphics processing units (GPUs) to train different models and datasets could be leveraged.

Appendix

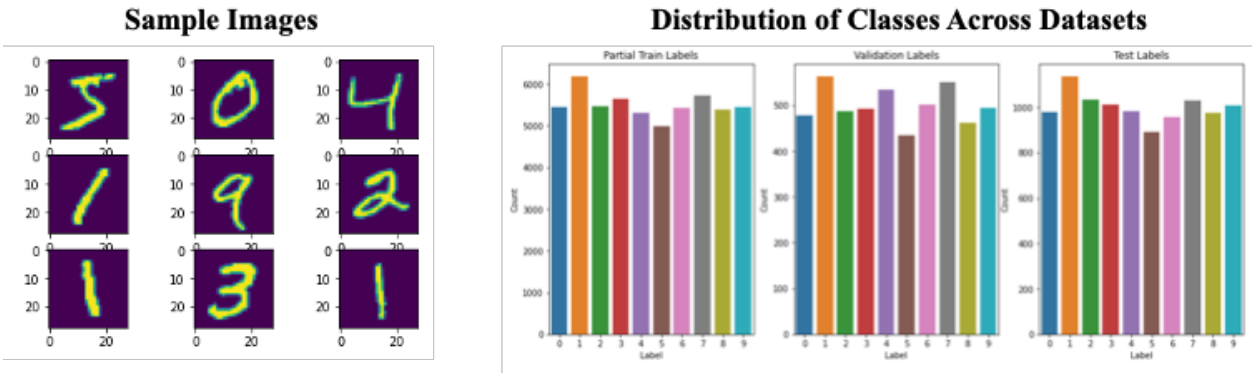


Figure 1: The sample images are examples of what the neural network will be tasked to classify. The distribution of classes across the train, validation, and test datasets are evenly spread across datasets.

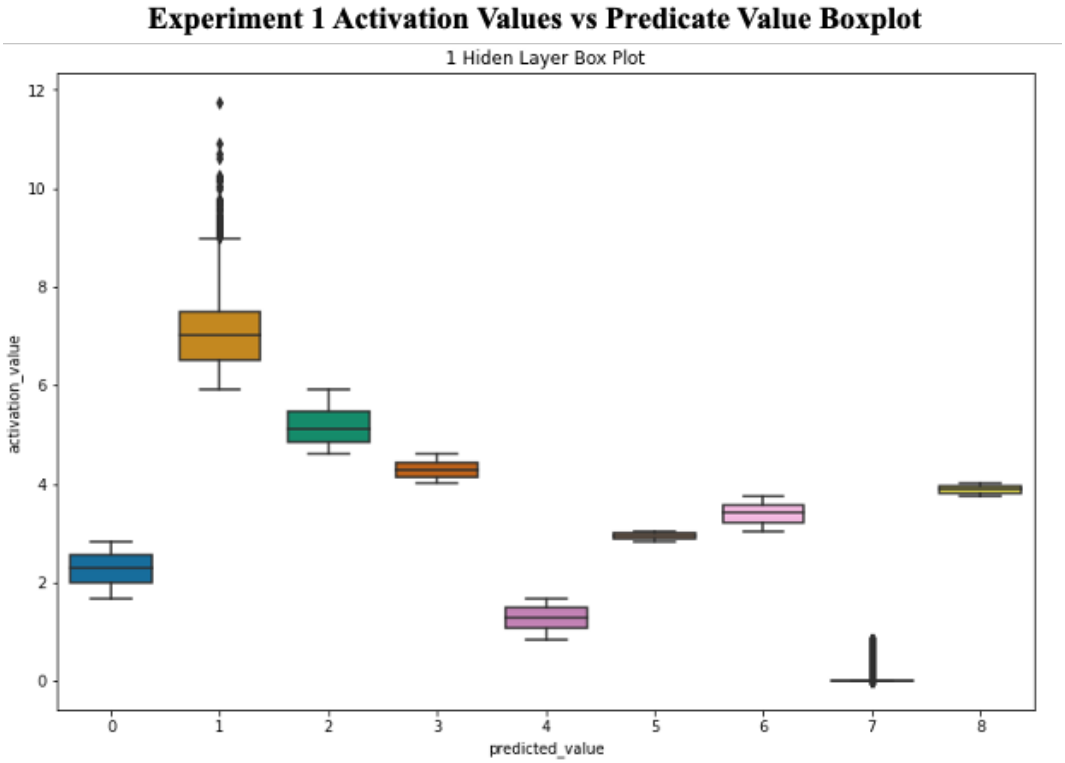


Figure 2: The boxplot shows substantial overlap between the range of values in the boxes reflecting the fact that the activation values of the hidden node are not able to discriminate between the classes.

Experiment 1 Results (Accuracy and Loss of Training and Validation Datasets)

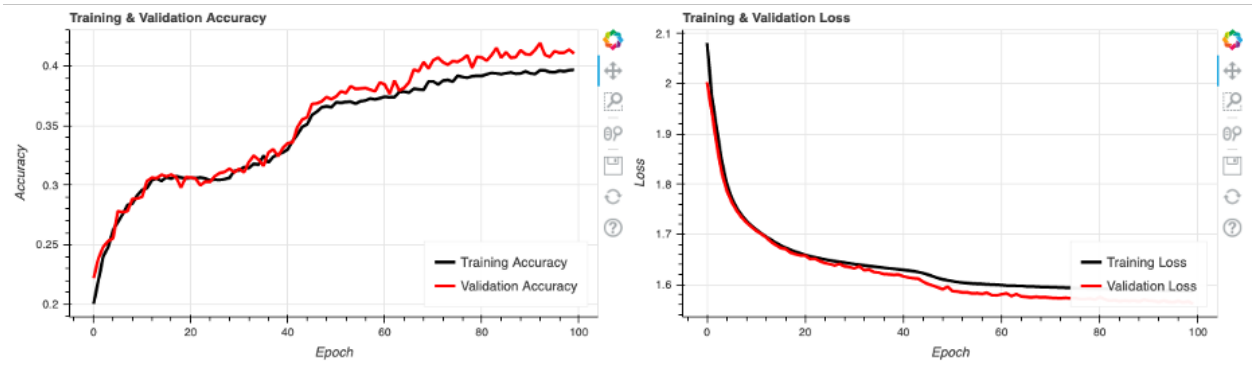


Figure 3: The model is underfitting and is not complex enough to accurately capture relationships between the features and the target classes. Using 1 node in the hidden layer is only able to achieve approximately 40% accuracy with a loss of 1.6.

Experiment 2 Activation Values Scatter Plot of Hidden Node 1 and 2

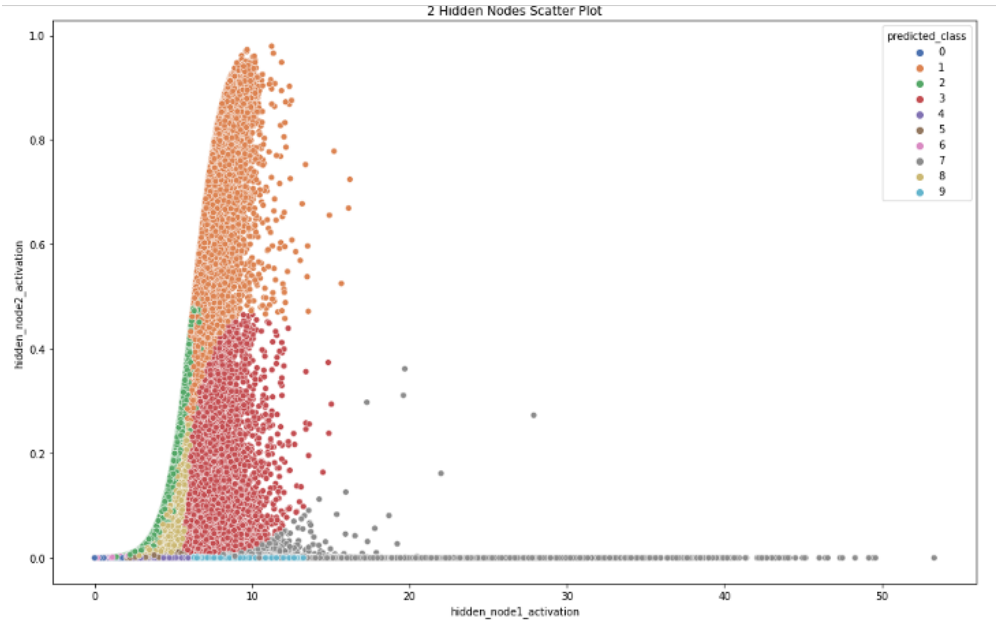


Figure 4: The scatter plot shows substantial overlap between the range of values in the scatter plot reflecting the fact that the activation values of the hidden node are not able to discriminate between the classes similar to experiment one.

Experiment 2 Results (Accuracy and Loss of Training and Validation Datasets)

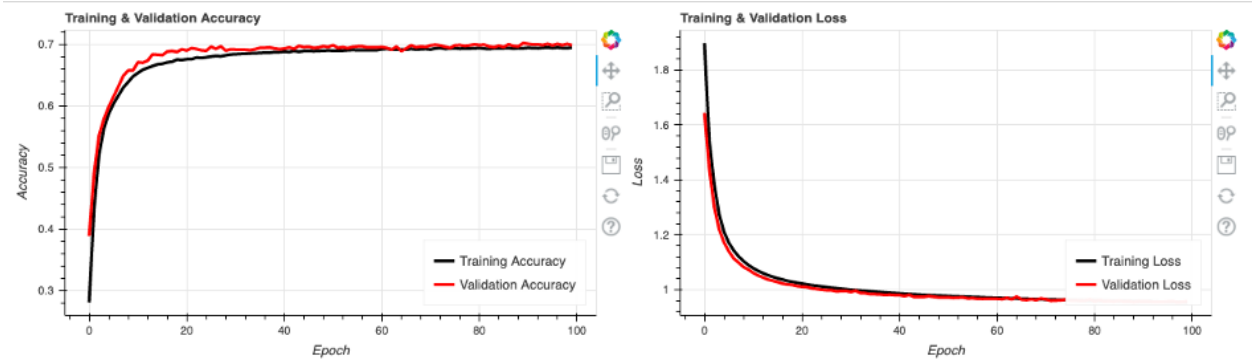


Figure 5: The model is performing better than experiment one but is still underfitting. Using 2 nodes in the hidden layer is able to achieve approximately 69% accuracy with a loss of 0.98.

Experiment 1 – Classification Report

	precision	recall	f1-score	support
0	0.65	0.40	0.50	1601.0
1	0.90	0.71	0.79	1439.0
2	0.38	0.34	0.36	1137.0
3	0.23	0.26	0.24	893.0
4	0.35	0.45	0.39	762.0
5	0.12	0.29	0.17	358.0
6	0.27	0.23	0.25	1122.0
7	0.84	0.39	0.53	2214.0
8	0.12	0.25	0.17	474.0
9	0.00	0.00	0.00	0.0
accuracy	0.40	0.40	0.40	0.4
macro avg	0.39	0.33	0.34	10000.0
weighted avg	0.55	0.40	0.45	10000.0

Experiment 1 – Confusion Matrix

	0	1	2	3	4	5	6	7	8	9
0	641	2	23	58	230	281	153	49	104	60
1	0	1018	345	23	0	9	13	5	26	0
2	1	106	391	285	6	40	138	19	151	0
3	4	6	148	232	18	55	208	10	209	3
4	160	0	3	17	340	49	14	47	28	104
5	59	1	13	37	12	105	55	8	58	10
6	77	1	65	206	28	210	258	17	245	15
7	37	0	3	39	347	73	3	866	33	813
8	1	1	41	113	1	70	116	7	120	4
9	0	0	0	0	0	0	0	0	0	0

Experiment 2 – Classification Report

	precision	recall	f1-score	support
0	0.87	0.81	0.84	1054.00
1	0.92	0.83	0.87	1271.00
2	0.72	0.78	0.75	962.00
3	0.60	0.57	0.58	1055.00
4	0.70	0.69	0.70	990.00
5	0.43	0.50	0.46	780.00
6	0.85	0.78	0.81	1044.00
7	0.64	0.75	0.69	886.00
8	0.48	0.48	0.48	967.00
9	0.63	0.64	0.64	991.00
accuracy	0.69	0.69	0.69	0.69
macro avg	0.68	0.68	0.68	10000.00
weighted avg	0.70	0.69	0.69	10000.00

Experiment 2 – Confusion Matrix

	0	1	2	3	4	5	6	7	8	9
0	851	0	4	0	43	52	72	2	22	8
1	0	1049	54	99	0	9	0	22	37	1
2	0	23	748	47	3	23	40	4	74	0
3	2	41	30	604	36	49	0	128	135	30
4	32	0	3	3	686	98	3	18	10	137
5	47	1	17	46	63	387	14	7	156	42
6	44	3	71	5	9	43	811	0	56	2
7	0	0	1	38	13	10	0	662	10	152
8	3	18	100	152	13	185	18	8	468	2
9	1	0	4	16	116	36	0	177	6	635

Figure 6: Both models are confusing the actual versus predicted classes such as the digits 8 and 5. Experiment one is correct 12% and experiment two is correct 48% for digit 8 and 11% and 43% for digit 5. The f1 score for both of these digits are also low with 0.17 for both digits 5 and 8 in experiment one and 0.46 and 0.48 in experiment two.

Experiment 3 Trial Results

	# Nodes	Test Accuracy (%)	Test Loss	Epochs Before Stop	Process Time (seconds)
0	1	40.62	1.5356	100	47.70
1	2	43.19	1.5138	100	82.56
2	4	86.46	0.4949	69	39.93
3	8	92.77	0.2707	48	29.73
4	16	95.45	0.1769	72	44.95
5	32	96.48	0.1556	57	42.24
6	64	97.46	0.1307	38	31.84
7	128	98.01	0.1902	74	89.17
8	256	98.14	0.1659	61	107.25
9	512	98.34	0.1471	52	148.43
10	1024	98.45	0.1444	55	285.26

Figure 7: The best model has 1024 nodes in a single hidden layer, however using 128, 256, or 512 nodes would also translate into a model that performs well.

1024 Nodes Confusion Matrix

	0	1	2	3	4	5	6	7	8	9
0	973	0	3	0	0	2	5	1	1	1
1	1	1126	2	0	0	0	2	2	1	2
2	1	2	1013	2	3	0	0	5	2	1
3	0	2	0	995	0	8	1	1	2	1
4	1	0	1	0	968	1	2	1	3	7
5	0	1	0	3	0	873	5	0	3	2
6	2	2	2	0	2	5	943	0	1	0
7	1	1	6	2	2	0	0	1010	2	3
8	1	1	4	4	0	2	0	5	954	2
9	0	0	1	4	7	1	0	3	5	990

1024 Nodes Classification Report

	precision	recall	f1-score	support
0	0.99	0.99	0.99	986.00
1	0.99	0.99	0.99	1136.00
2	0.98	0.98	0.98	1029.00
3	0.99	0.99	0.99	1010.00
4	0.99	0.98	0.98	984.00
5	0.98	0.98	0.98	887.00
6	0.98	0.99	0.98	957.00
7	0.98	0.98	0.98	1027.00
8	0.98	0.98	0.98	973.00
9	0.98	0.98	0.98	1011.00
accuracy	0.98	0.98	0.98	0.98
macro avg	0.98	0.98	0.98	10000.00
weighted avg	0.98	0.98	0.98	10000.00

Figure 8: The confusion matrix using a neural network with 1024 nodes in a single hidden layer is able to correctly predict the classes and the f1 score for each class was almost perfect with a 0.98 or 0.99 score for each class.

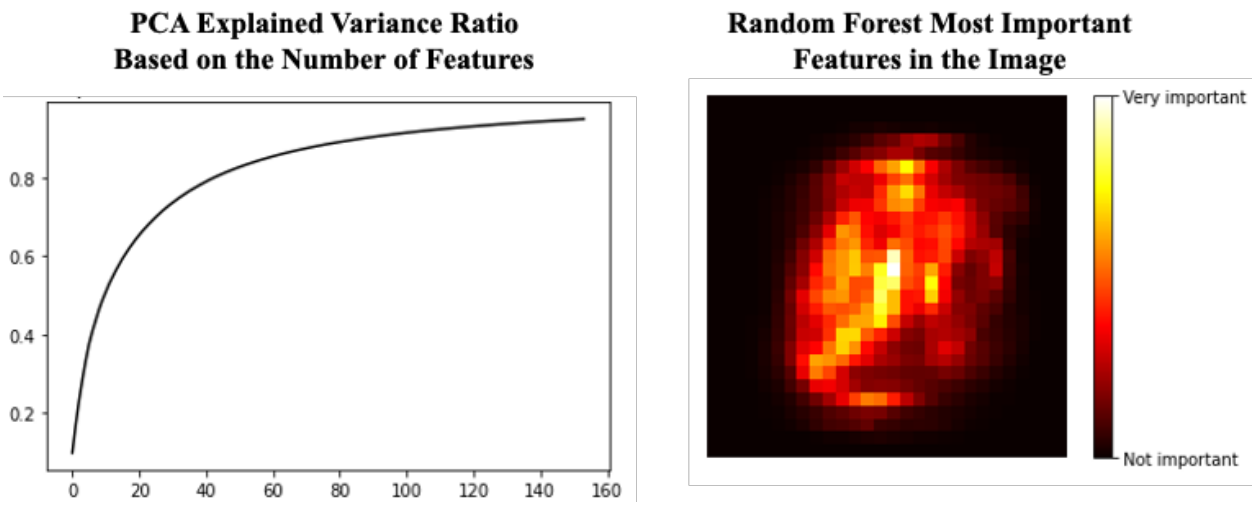
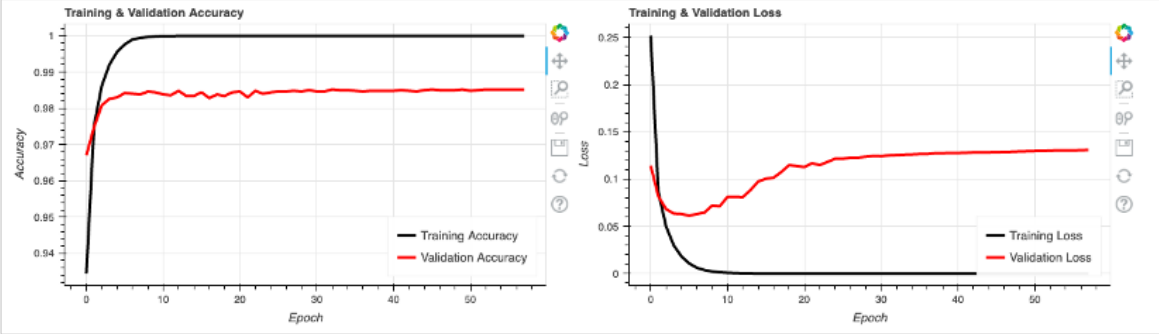


Figure 9: The PCA explained ratio peaks at about 155 features and the Random Forest classifier identified that the most important features are more toward the center of the image

1024 Nodes – Single Layer – PCA Reduction with 95% Variance – Performance



1024 Nodes – Single Layer – PCA Reduction with 95% Variance – Confusion Matrix

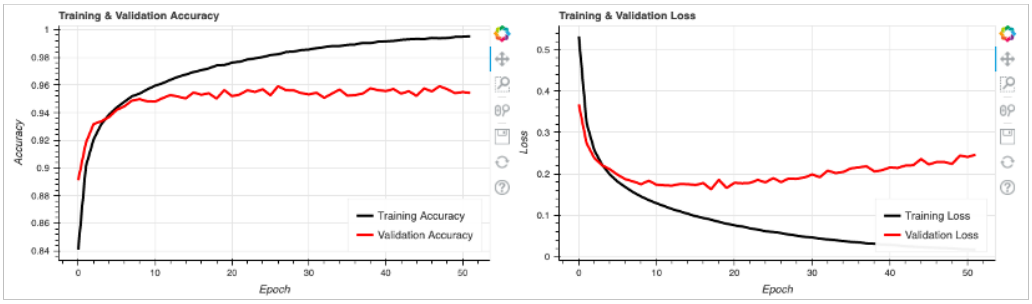
	0	1	2	3	4	5	6	7	8	9
0	975	0	1	1	0	3	5	1	2	2
1	1	1125	1	0	0	0	2	3	1	3
2	1	3	1015	5	4	0	0	8	3	0
3	0	1	1	993	0	8	1	3	2	2
4	1	0	3	0	969	1	1	0	3	6
5	0	1	0	3	0	875	4	0	1	2
6	0	2	2	0	2	2	944	0	2	0
7	1	1	5	2	0	1	0	1007	3	5
8	1	2	4	3	0	1	1	2	952	1
9	0	0	0	3	7	1	0	4	5	988

1024 Nodes – Single Layer – PCA Reduction with 95% Variance – Classification Report

	precision	recall	f1-score	support
0	0.99	0.98	0.99	990.00
1	0.99	0.99	0.99	1136.00
2	0.98	0.98	0.98	1039.00
3	0.98	0.98	0.98	1011.00
4	0.99	0.98	0.99	984.00
5	0.98	0.99	0.98	886.00
6	0.99	0.99	0.99	954.00
7	0.98	0.98	0.98	1025.00
8	0.98	0.98	0.98	967.00
9	0.98	0.98	0.98	1008.00
accuracy	0.98	0.98	0.98	0.98
macro avg	0.98	0.98	0.98	10000.00
weighted avg	0.98	0.98	0.98	10000.00

Figure 10: Using PCA dimension reduction, the best model from experiment three with 1024 nodes in a single hidden layer was able to achieve an accuracy of 98.43% using PCA, the confusion matrix has good ratios of predicted vs actual classes, and the classification report maintained an f1 score of 0.98 and 0.99 for each class.

1024 Nodes – Single Layer – Random Forest Feature Reduction – Performance



1024 Nodes – Single Layer – Random Forest Feature Reduction – Confusion Matrix

	0	1	2	3	4	5	6	7	8	9
0	953	1	2	5	1	8	20	1	7	7
1	1	1123	4	0	1	0	5	3	1	2
2	8	1	964	10	4	3	3	31	5	4
3	0	4	8	948	2	21	1	4	11	13
4	0	0	8	0	940	2	14	4	6	12
5	12	3	8	37	5	845	8	1	12	17
6	3	2	4	0	4	2	900	0	2	3
7	3	0	17	4	1	1	0	969	1	3
8	0	1	13	5	4	4	7	7	918	11
9	0	0	4	1	20	6	0	8	11	937

1024 Nodes – Single Layer – Random Forest Feature Reduction – Classification Report

	precision	recall	f1-score	support
0	0.97	0.95	0.96	1005.00
1	0.99	0.99	0.99	1140.00
2	0.93	0.93	0.93	1033.00
3	0.94	0.94	0.94	1012.00
4	0.96	0.95	0.96	986.00
5	0.95	0.89	0.92	948.00
6	0.94	0.98	0.96	920.00
7	0.94	0.97	0.96	999.00
8	0.94	0.95	0.94	970.00
9	0.93	0.95	0.94	987.00
accuracy	0.95	0.95	0.95	0.95
macro avg	0.95	0.95	0.95	10000.00
weighted avg	0.95	0.95	0.95	10000.00

Figure 11: The Random Forest classifier's 70 most important features was able to achieve an accuracy of 94.97%, the confusion matrix had good ratios of predicted vs actual classes, and the classification report had f1 scores as low as 0.92 for the digit 5 and as high as 0.99 for the digit 1 with an average of 0.95 across all digits.

Various Model Architecture Performance on MNIST dataset

	Architecture	# Nodes	Test Accuracy (%)	Test Loss	Epochs Before Stop	Process Time (seconds)
0	2x10	[10, 10]	94.09	0.2236	58	38.06
1	2x20	[20, 20]	96.08	0.1654	50	35.03
2	2xMix	[150, 50]	97.81	0.1942	35	50.05
3	5x10	[10, 10, 10, 10, 10]	93.15	0.2627	89	64.57
4	5x20	[20, 20, 20, 20, 20]	95.40	0.2574	68	53.87
5	5xMix	[300, 200, 100, 50, 25]	98.34	0.3087	62	185.89

Figure 12: The best model used 5 hidden layers with [300, 200, 100, 50, 25] nodes in each layer. The model was able to achieve a 98.34% accuracy and a loss of 0.3087. The model was comparable to the best model from experiment three and experiment four with PCA dimension reduction. Therefore, increasing the layers and making the nodes wider does not provide tangible improvement to the performance. The results of experiment six may not be true for other datasets.

References

Brownlee, Jason. 2019. “A Gentle Introduction to the Rectified Linear Unit (ReLU).” *Machine Learning Mastery* (blog). January 8, 2019. https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/.

———. 2020. “Softmax Activation Function with Python.” *Machine Learning Mastery* (blog). October 18, 2020. https://machinelearningmastery.com/softmax-activation-function-with-python/.

Chollet, François. 2018. *Deep Learning with Python*. Shelter Island, New York: Manning Publications Co.

Géron, Aurélien. 2019. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. Second edition. Beijing [China] ; Sebastopol, CA: O’Reilly Media, Inc.

“How Many Hidden Units Should I Use?” n.d. Accessed July 6, 2021. http://www.faqs.org/faqs/ai-faq/neural-nets/part3/section-10.html.

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/.

Maladkar, Kishan. 2018. “Overview Of Convolutional Neural Network In Image Classification.” *Analytics India Magazine* (blog). January 25, 2018. https://analyticsindiamag.com/convolutional-neural-network-image-classification-overview/.

Novotný, Jaromír, and Pavel Ircing. 2018. “The Benefit of Document Embedding in Unsupervised Document Classification.” In *Speech and Computer*, 470–78. Lecture Notes in Computer Science. Cham: Springer International Publishing.

Paola, Justin D., and Robert A. Schowengerdt. 1995. “A Detailed Comparison of Backpropagation Neural Network and Maximum-Likelihood Classifiers for Urban Land Use Classification.” *IEEE Transactions on Geoscience and Remote Sensing* 33 (4): 981–96.

Rojas, Raul. 2003. “Networks of Width One Are Universal Classifiers.” In *Proceedings of the International Joint Conference on Neural Networks, 2003.*, 4:3124–27 vol.4. https://doi.org/10.1109/IJCNN.2003.1224071.

Sanghvirajit. 2021. “A Complete Guide to Adam and RMSprop Optimizer.” Medium. May 14, 2021. https://medium.com/analytics-vidhya/a-complete-guide-to-adam-and-rmsprop-optimizer-75f4502d83be.