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

Northwestern University

July 10, 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 providing 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 in a neural network. Results suggest the PCA is an effective and the best solution to reduce the inputs in a neural network. achieved by the classic K-means algorithm and the Rand Index score.

*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. This study explores both PCA and the Random Forest classifier impacts on the results of the neural network. 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 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). A fundamental element of a neural network is the processing node, which 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 in the neural network. Most of the variance is persevered and the dataset is now less than 20% of the original size. Another method to reduce the size of the dataset is to use 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. This is irrespective of the fact whether the data is linear or non-linear (linearly inseparable) (Kumar 2020).

**Data and Methods**

The dataset used in this study is the Modified National Institute of Standards and Technology (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 in the appendix 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 using image classification and how they can impact the results of the neural network, the study conducted three experiments:

1. A dense neural network with 784 input nodes, a hidden layer with 1 node, and 10 output nodes (corresponding to the 10 digits). The hypothesis was to see substantial overlap between the range of values in the boxes 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 (corresponding to the 10 digits). The hypothesis was to see similar results to the first experiment and the activation values fail to discriminate between the classes
3. A dense neural network with 784 input nodes, a hidden layer with more hidden nodes (1, 2, 4, 8, 16, 64, 128, 256, 512, and 1024), and 10 output nodes (corresponding to the 10 digits). The hypothesis was as we increase the width of the neural network with more nodes the activation values 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 number of features in the neural network by performing two experiments:

1. 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 3
2. 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 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 3

All models used 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 by 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]](#footnote-1). The results of first experiment with 1 node in the hidden layer confirmed the hypothesis. As we can see in Figure 2 in the appendix, 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. We can also see in Figure 3 in the appendix 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 also confirmed the hypothesis. As we can see in Figure 4 in the appendix, 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 1. The two nodes are able to cluster the digits 1, 2, 3, and 7. We can also see in Figure 5 in the appendix that the model is performing better than experiment 1 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 as we can see in Figure 6 in the appendix. The confusion matrix in Figure 6 in the appendix of both experiments highlights that both models are confusing the actual versus predicted classes such as the digits 8 and 5. Experiment 1 is correct 12% and experiment 2 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 1 and 0.46 and 0.48 in experiment 2. The results show that increasing the width of the hidden layer by added more nodes improves the performance of the models and in experiment 3 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]. The results were similar and as the nodes increased the performance of the model improved. Figure 7 in the appendix show the results of the trials and the best model has 1024 nodes, however using 128, 256, or 512 nodes would also translate into a model that performs well. Figure 8 in the appendix 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 a 0.98 or 0.99 score for each class 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 in the appendix, 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 in the appendix confirms the hypothesis for 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%, the confusion matrix had good ratios of predicted vs actual classes, and the classification report maintained an f1 score of 0.98 and 0.99 for each class with an average of 0.98. Figure 11 in the appendix 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. The f1 score maintained a 0.98 average across all classes using the same model architecture with and without PCA dimension reduction. 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, the using a Random Forest classifier to reduce features is a good option, 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 in the appendix 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 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.

**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 showed proved the 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 to reduce the features to 70 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. Convolutional neural networks (CNN) 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**

Chart, bar chart

Description automatically generated

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.

Chart, box and whisker chart

Description automatically generated

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.

Chart, line chart

Description automatically generated

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.

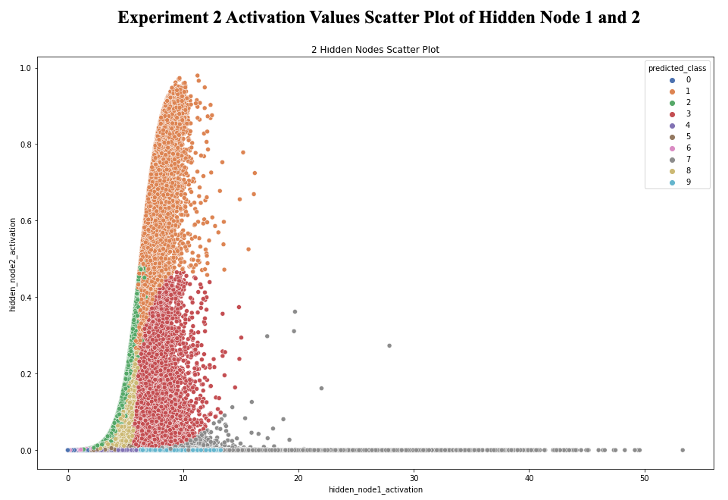


Figure 4: The scatter plot shows substantial overlap between the range of values in the boxes reflecting the fact that the activation values of the hidden node are not able to discriminate between the classes similar to experiment 1. The two nodes are able to cluster the digits 1, 2, 3, and 7.

Graphical user interface, chart, line chart

Description automatically generated

Figure 5: that the model is performing better than experiment 1 but is still underfitting. Using 2 nodes in the hidden layer is able to achieve approximately 69% accuracy with a loss of 0.98.

Calendar

Description automatically generated

Figure 6: Both models are confusing the actual versus predicted classes such as the digits 8 and 5. Experiment 1 is correct 12% and experiment 2 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 1 and 0.46 and 0.48 in experiment 2

Table

Description automatically generated

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.

Calendar

Description automatically generated

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.

A picture containing shape

Description automatically generated

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

Calendar

Description automatically generated

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%, 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.

Calendar

Description automatically generated

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

Table

Description automatically generated

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

(Chollet 2018)

1. GitHub repository for the code that performed the analysis used in this study: <https://github.com/chrisfesta/NWU_MSDS458> [↑](#footnote-ref-1)