# Evaluating Basic Direct Neural Network Structures

Coursework for Data Science 458: Research Programming Assignment I

**Researcher: Michael Rocchio**

*Graduate Student at Northwestern University*

*Researcher Contact Email: MichaelRocchio2023@u.northwestern.edu*

## Abstract

Applications for DNNs (Deep Neural Networks) are countless, from image classification to the prediction of real estate prices. Due to easy to use open source programming packages such as Keras, Tensorflow, and Pytorch., which are the primary packages utilized for building these models, as well as an ever-increasing availability of data they have in many ways replaced traditional machine learning methods as the go to solution within many industries. In the area of image recognition and classification, creating neural networks based on the MNIST data set have been the go-to way to test new types of networks and their efficacy.

## Research, Dataset Introduction, & Literature Review

The purpose of the research below will be to, in 5 separate experiments, build and evaluate several different network architectures for the purpose of image classification on the MNIST data set. Examples from this set are shown in **Figure 1**. Each image consists of 784 pixels laid out on a 28x28 plane in grayscale. These images are then converted into 1 dimensional NumPy arrays consisting of int values between 0 – 255 representing the intensity of the various pixels within greyscale image. The image set contains 70,000 images labeled

between 0 and 9 which will be split into a train and test set consisting of 60,000 & 10,000 images, respectively. For the input into our neural network we will be breaking down the 28x28 images into a single row of 784 pixels. As there is no data leakage between the models, we will be using the test MNIST set as the validation data and evaluate our models based on that. We will also be min max scaling the pixels of the images by subtracting the min, 0, and dividing by the max, 255. For each of the experiments to ensure that we control variables, we will be using the following parameters to compile the models; for an optimizer we will be using RMSprop, a standard use adaptive learning rate optimizer. To compute loss, we will be using sparse categorical cross-entropy as the y labels are not hot coded. The metrics observed will be categorical accuracy of the training and validation set. We will also be fitting the model with 150 epochs as the standard without early cutoff to achieve the aforementioned goal of controlling the variables of the experiments. There have been a number of various pieces of literature on this topic as classification of the MNIST data set is quit an old problem within the field. However, all most all the research recently has been based on CNNs and transformers. This is true in the trained model IJCNN 2015. However, Yann LeCun, Corinna Cortes, & Christopher J.C. Burges were able to achieve a test error rate of 0.35 using a 6 layer DNN with 6 layers consisting of 784, 2500, 2000, 1500, 1000, 500 & 10 nodes, respectively. A known source within the data science community with various attempts at the MNIST dataset and their accuracy rates can be found at http://yann.lecun.com/exdb/mnist/**.**

**Experiment I**

In the first experiment we will be creating a 3-layer Dense Neural Network using Keras. The purpose of this will be to demonstrate in inefficacy of dense layers within networks containing only one node. The architecture of the network is as follows and can be visualized

in **Figure 2**. The input layer will be incorporated into the hidden layer and contain 784 input notes which is tailored to restructured MNIST data. The one hidden layer will contain 1 node activated by the Rectified Linear Unit function (RELU). This is the most utilized function within the hidden layers of neural networks as it is a simple and efficient linear function that doesn't overuse computing resources. Finally, the output layer will contain 10 nodes with SoftMax activation functions as we are doing a multi-classification output. After training the first model, the final validation accuracy was approx. 40% which was to be expected, a graph of the history of model1's training metrics is shown in **Figure 3**. This means that the model is unable to accurately predict the digits which is demonstrated in **Figure 4** & **Figure 5,** the latter of the two figures shows a boxplot that demonstrates the severe overlap of the range of the values output by the singular note within the hidden layer.

**Experiment II**

In the second experiment we will be recreating all the same scenarios of the first experiment with the exception that there will be two nodes in the hidden layer. This will very slightly increase the accuracy of the network but will not be sufficient in accurate classification of the images. A visual architecture of the network created in this experiment is demonstrated in **Figure 6**. Upon training the model we were able to achieve a validation accuracy of approx.. 68%, which is to be expected as the second model, despite doubling the number of hidden nodes, did not make it large enough to accurately predict the images. Selected outputs for the model predictions are in **Figure 7.** This can be show in **Figure 8** & **Figure 9,** the latter shows a scatter of output values for the nodes and demonstrates that some clustering is occurring, but the two nodes are still unable to differentiate between the classes.

**Experiment III**

In the final DNN experiment we will be recreating all the same scenarios as the first two experiments, but we will be creating a network that is accurate by scaling up the number of nodes significantly, to 256. A visualization of the network architecture is shown in **Figure 10** and **Figure 11** shows selected results and model prediction for 16 MNIST images**.** We will also be conducting a PCA analysis of each of the hidden nodes based on their significance in predicting each one of the output classes. The choice in the number of hidden nodes was selected after several experiments were conducted testing the validation accuracy at different densities of the layer. After training the model was able to achieve 98% accuracy on the validation set thus making it a production level model. Additionally, a principal component analysis was done to demonstrate the necessity of a larger hidden dense network. This PCA analysis identified 3 principal clusters necessary for each class's correct output prediction. Each graph in **Figure 12** shows scatterplots of the principal nodes and demonstrates uniquely identifiable clustering for the classification variable it is done on. Finally, **Figure 13** shows a 3d model and the clustering of the principal 3 nodes for all classes.

**Experiment IV**

During this next experiment we will be doing a PCA analysis to reduce the dimensionality of the MNIST data and create a more efficient model. We will reduce the number of input nodes from 784 to 154 as it makes up 95% of the variability of the MNIST image data. After doing the dimensionality reduction on the training data we will be recreating the model built in Experiment 3 with the exception of reducing the number of inputs to 154 to account for the reduced data. . A visualization of the network architecture is shown in **Figure 14.** Additionally, when preparing the dataset care was taken to transform the test data with the PCA model created with the training data in order to eliminate leakage between the two sets. After running the experiment the trained model was able to achieve a 98.2% validation

accuracy, which is on par with the performance of our 'best' model created in experiment 3, while also reducing the dimensionality of the input data and thus saving computing resources.

**Experiment V**

During our final experiment we will be using a random forest classifier to determine and visualize the importance of differing inputs within the pixels of a MNIST pixel and their importance on predicting output classes. This exemplifies the feasibility of conducting the PCA analysis in Experiment one as it determines if further dimensionality reduction on the data is a prudent option. **Figure 15** shows this and demonstrates the necessity of dimensionality reductions as many of the outer pixels have next to no importance on identifying classes and thus should be eliminated after identification through a PCA analysis.

**Management Analysis & Conclusion**

Regarding the management analysis, if asked to develop a DNN for image classification, I would follow a lot of the steps demonstrated in Experiments 3,4 & 5. However, I would do them backwards with some differences. First, I would split the data between a training and test set and be careful not to create any data leakage. Second, I would test to see if the data could undergo meaningful dimensionality reduction by identifying the variability in specific input pixels. If I determined this to be true, I would then conduct a PCA analysis to reduce the data and eliminate all components under a certain threshold of variability. Finally, I would train various models and deploy the one that had the best efficiency while maintain appropriate accuracy based on the results of the validation set.

To sum up, these five experiments show the beginning steps of building a DNN and specific ways of improving them using dimensionality reduction and by choosing the right number of nodes in the hidden network.

**Figure Index**

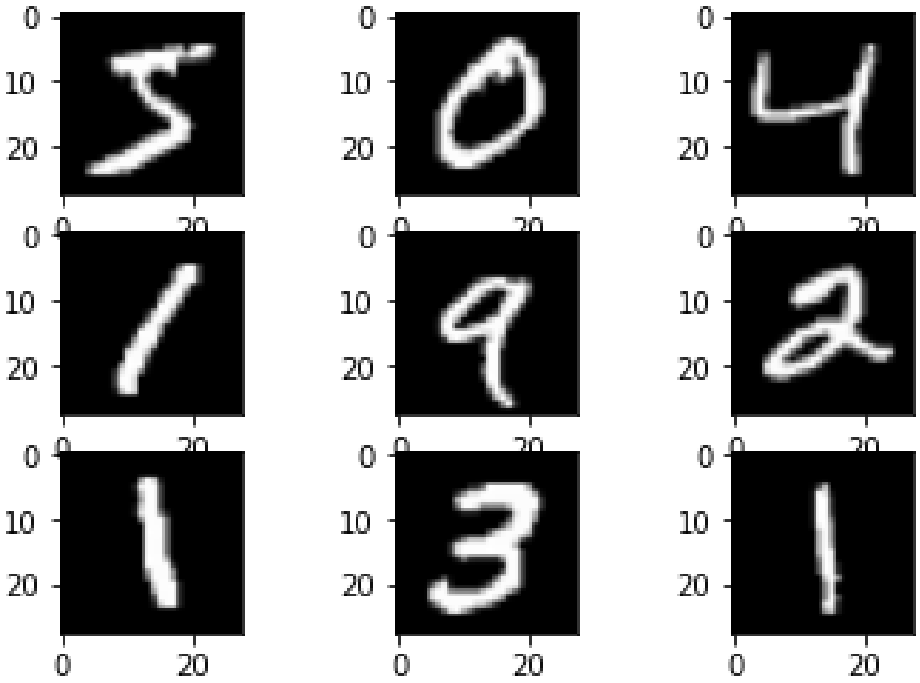

**Figure 1.** A Selection of 9 images from the MNIST dataset on 28 by 28 graphs.

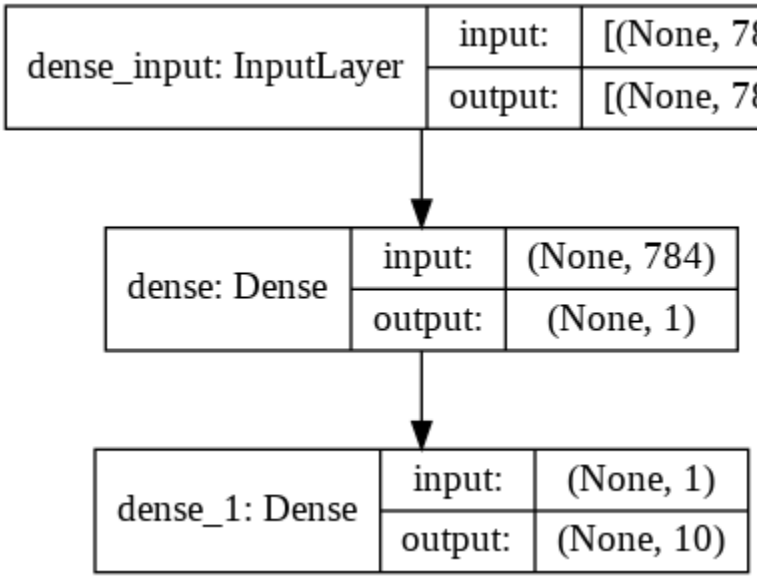


**Figure 2.** A visualization of the model created in Experiment 1.

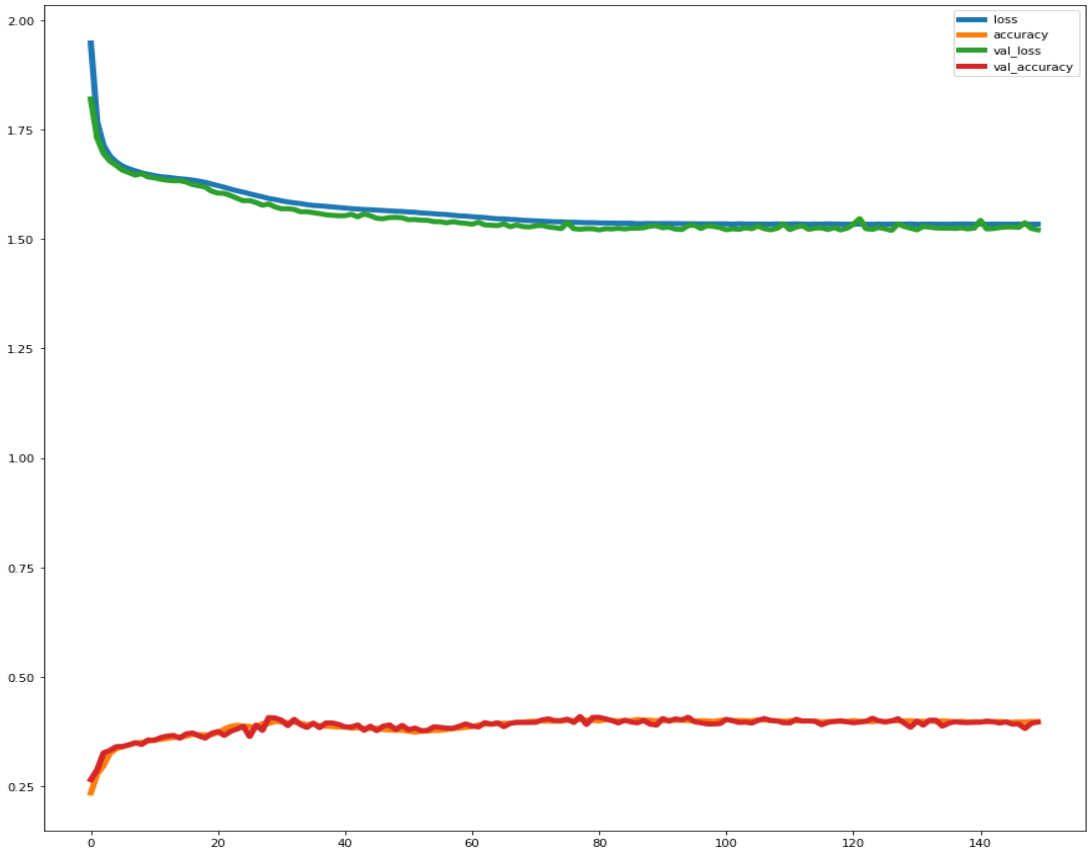


**Figure 3.** A historical graph of the metrics in model 1 over the 150 epochs.

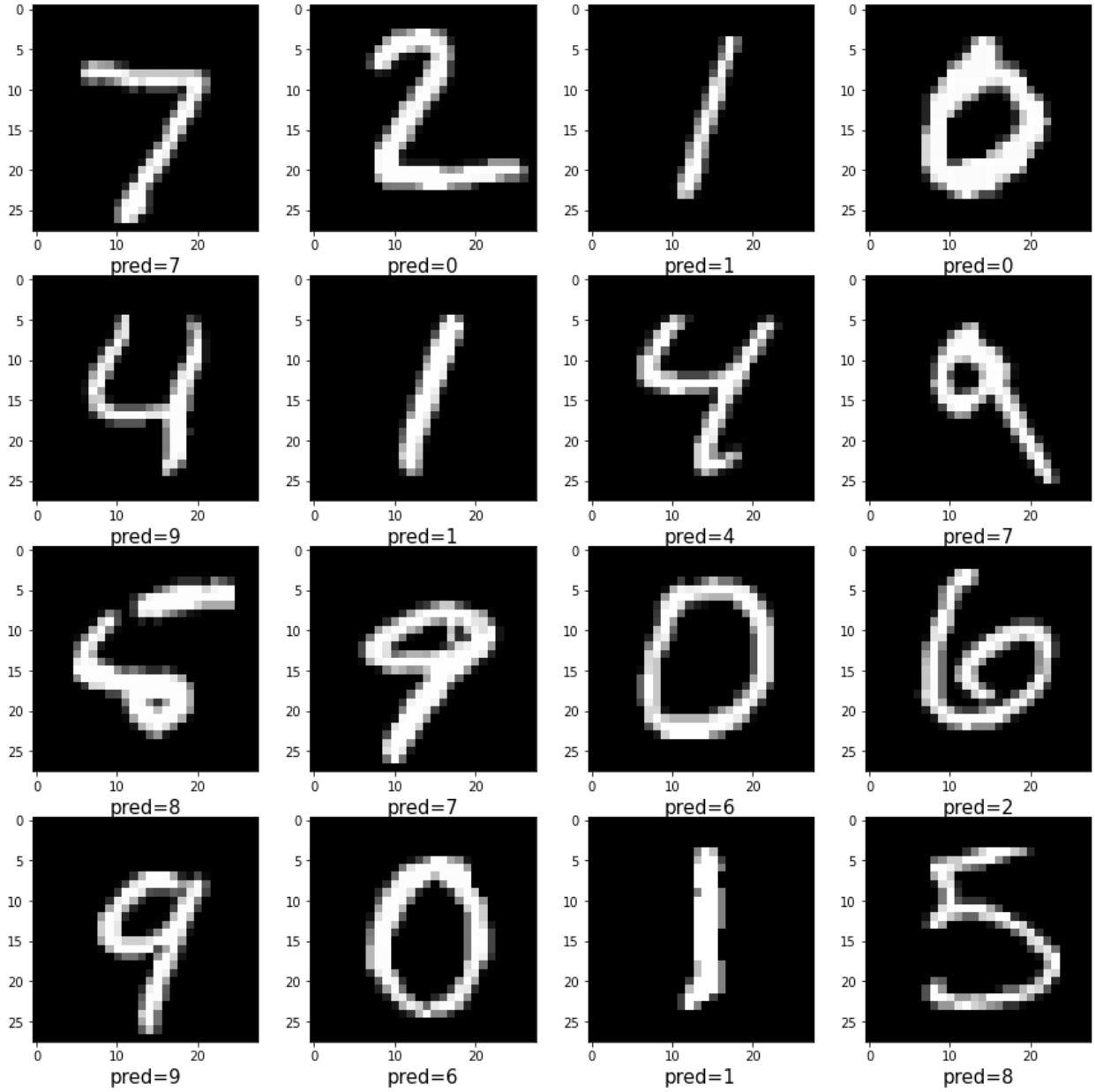


**Figure 4.** A random selection of 16 MNIST images and their predictions from model 1.

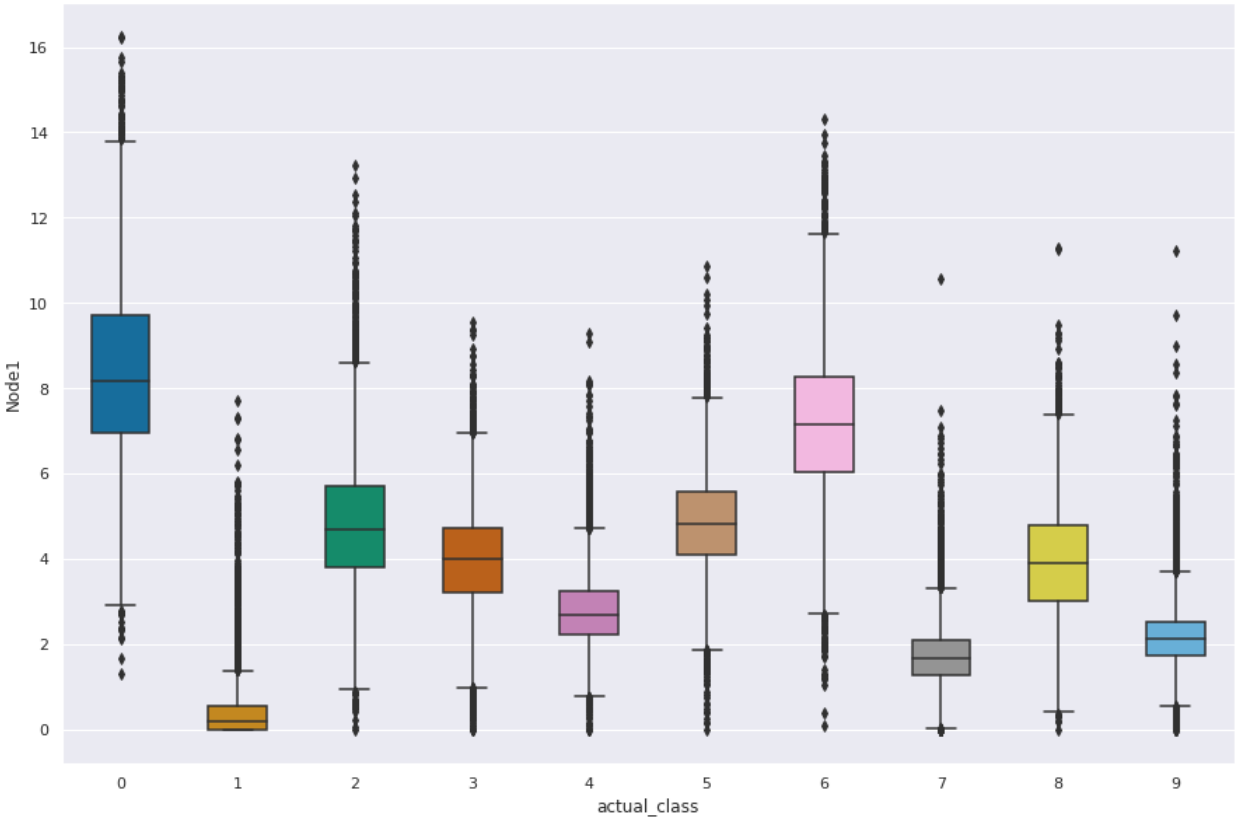


**Figure 5.** A boxplot showing the range of activation values for the output of the single node in the hidden network compared to the actual values of the MNIST dataset.

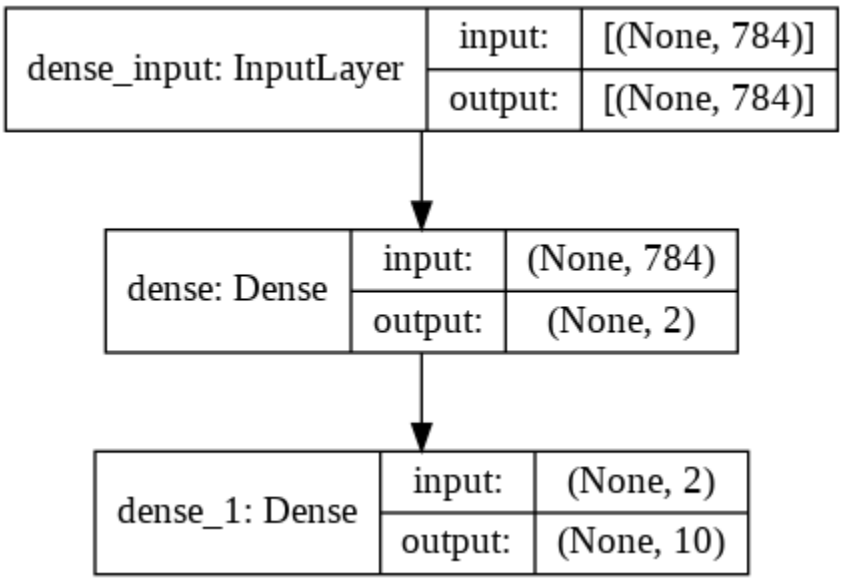


**Figure 6.** A visual architecture of the second model.

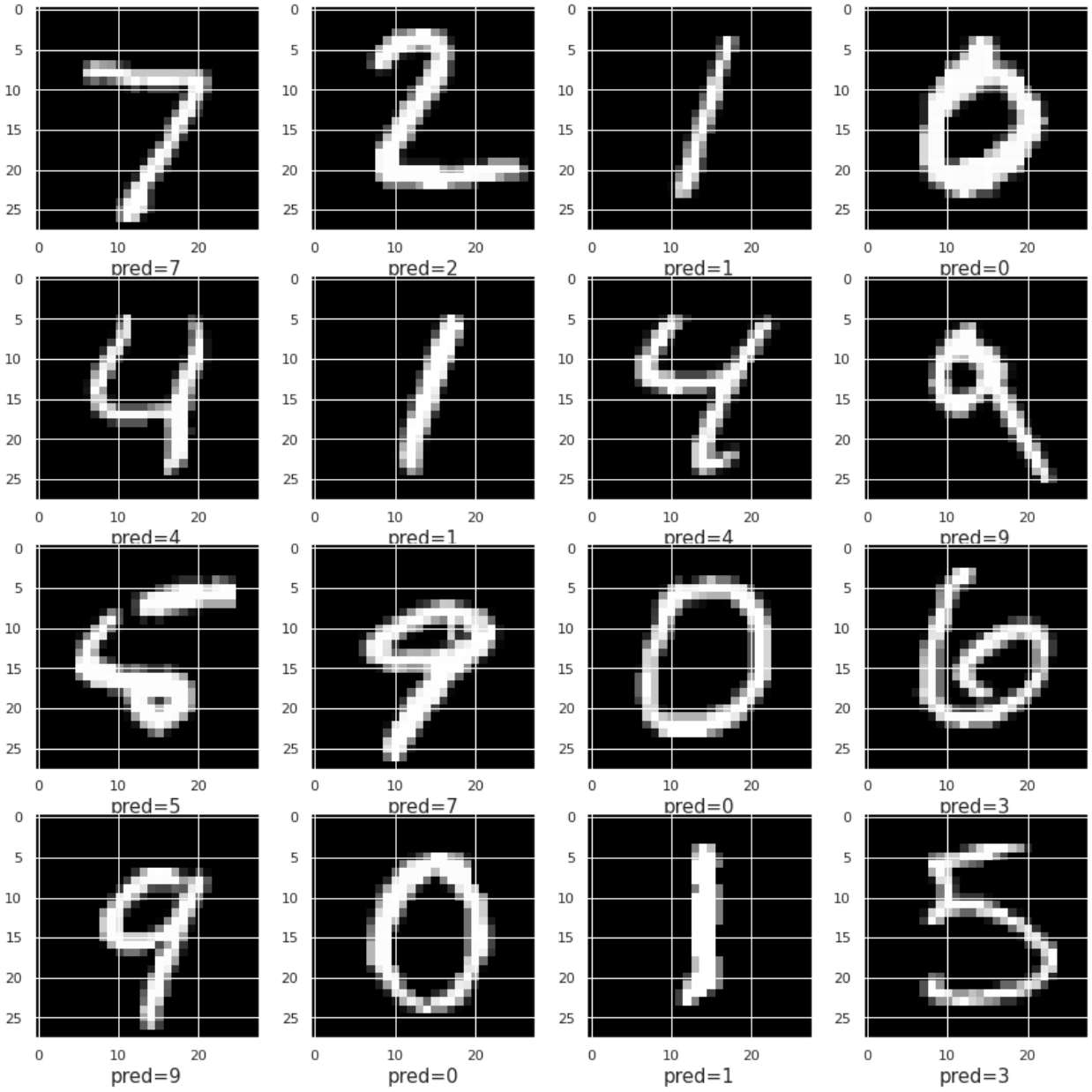


**Figure 7.** A random selection of 16 MNIST images and their predictions from model 2.

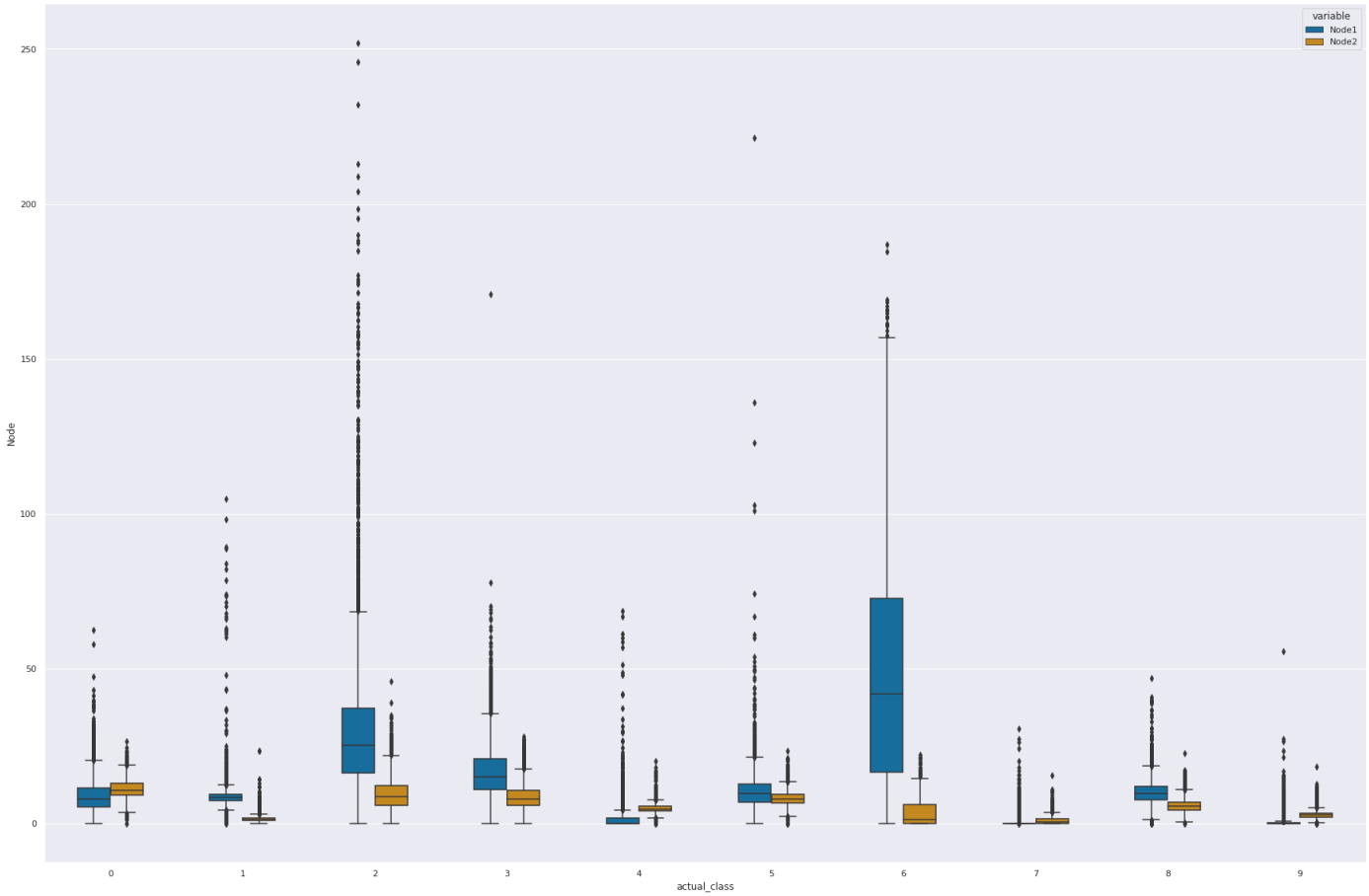


**Figure 8.** A demonstration of the range overlap of the values for nodes 1 and 2.


**Figure 9.** A scatter plot of the values of nodes 1 and 2 of the second model with the hue equaling the real values of the input classes.

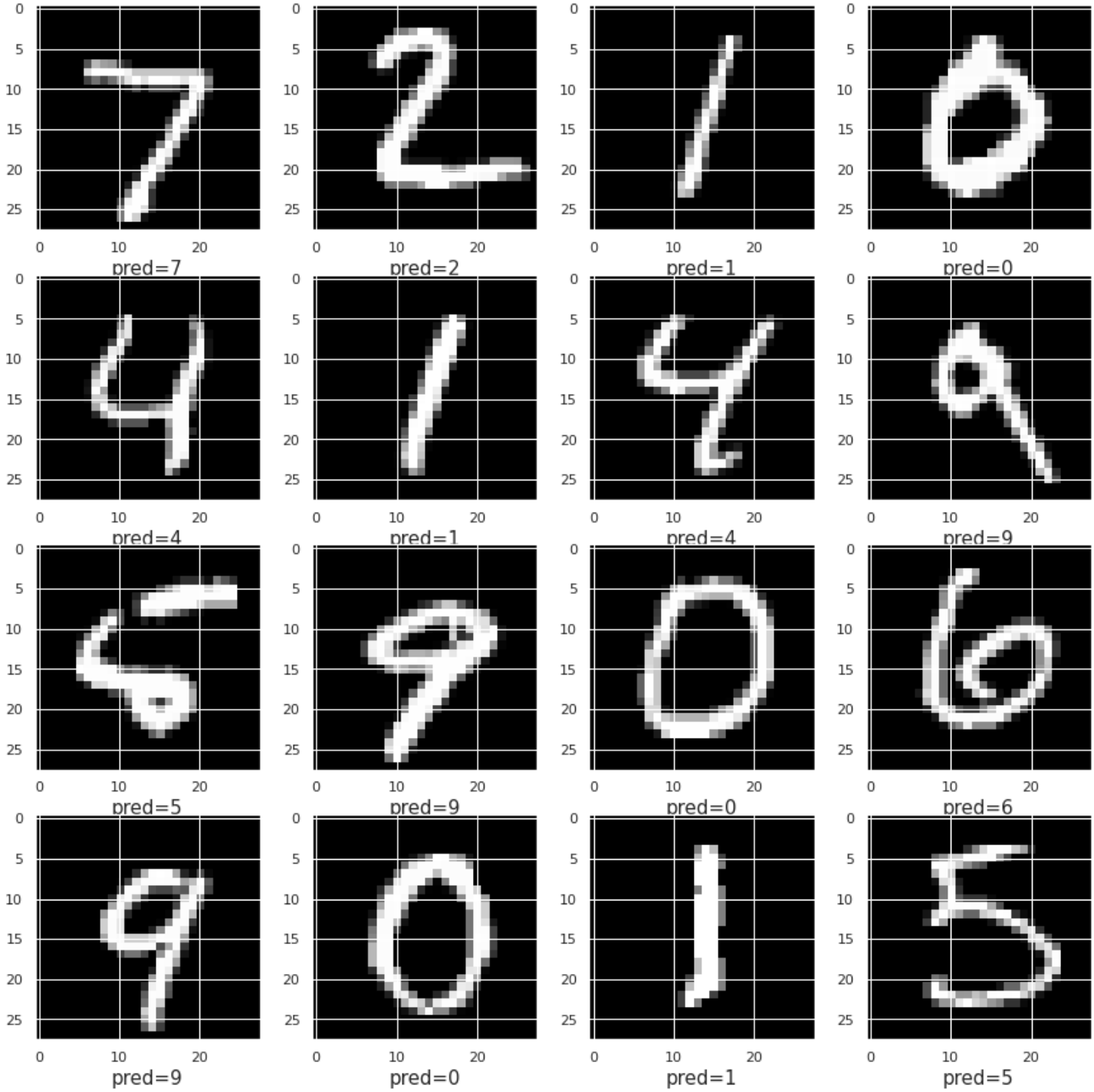


**Figure 10.** A random selection of 16 MNIST images and their predictions from model 1.

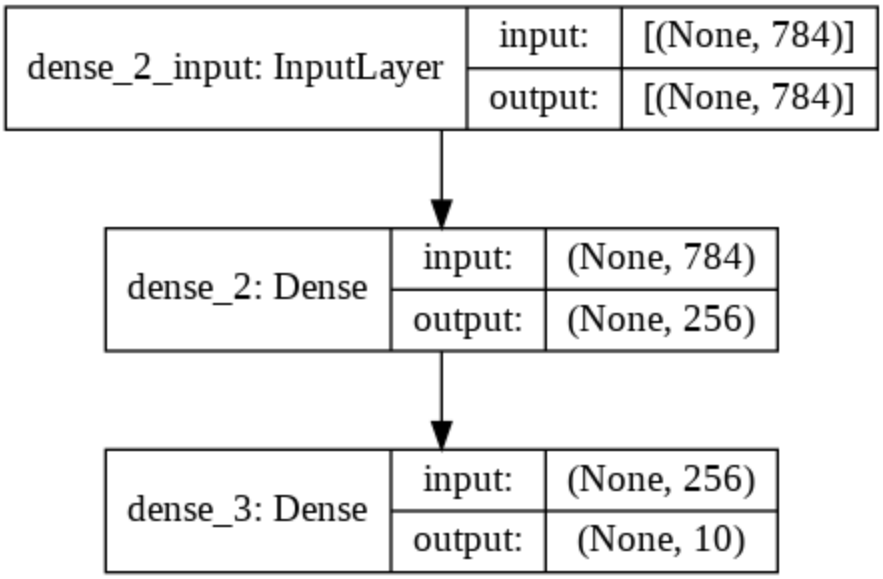


**Figure 11.** A visual architecture of the third and ‘best’ model.

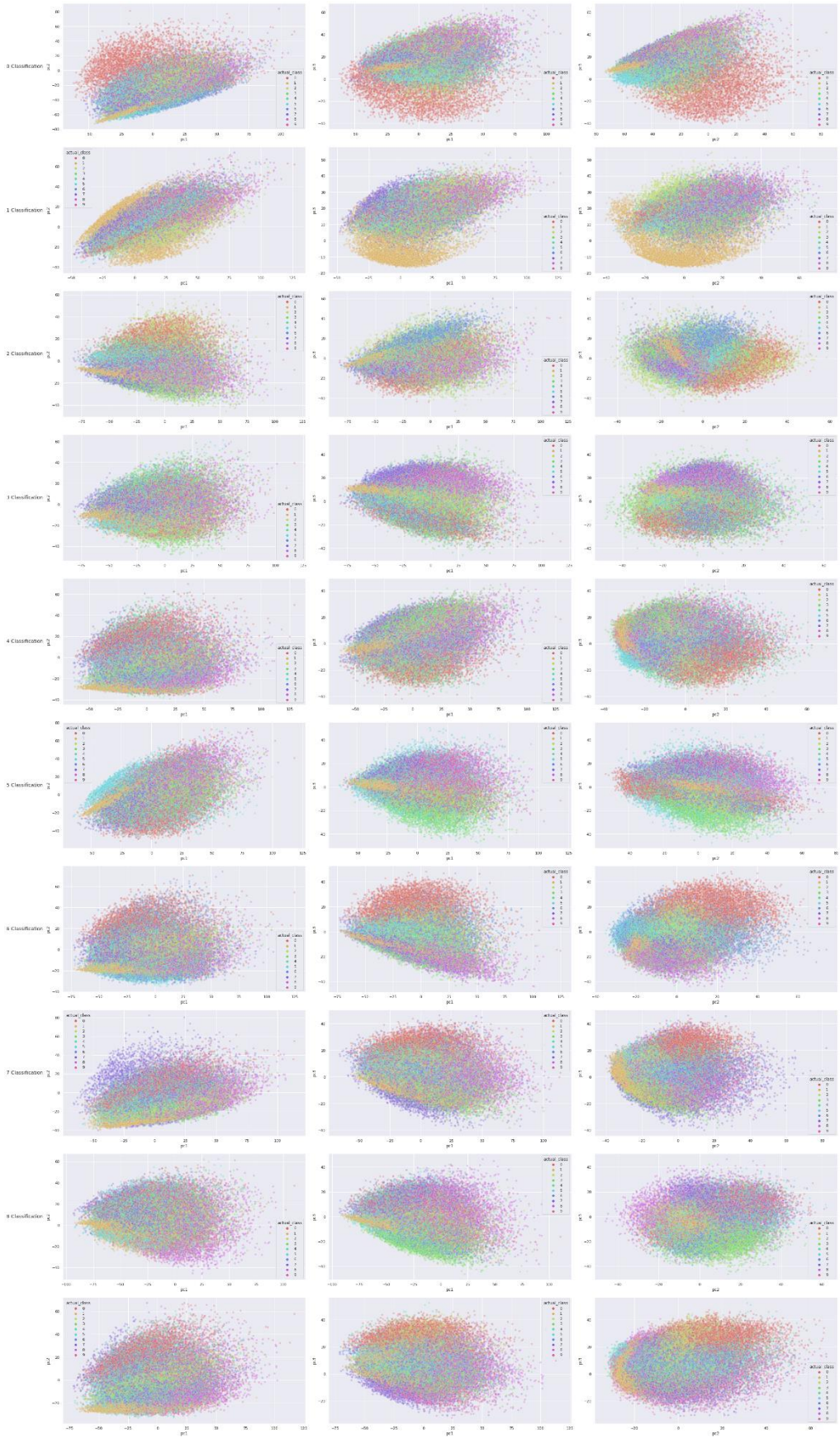

**Figure 12.** A PCA analysis of the 3 principal nodes most responsible for identifying the individual classes.

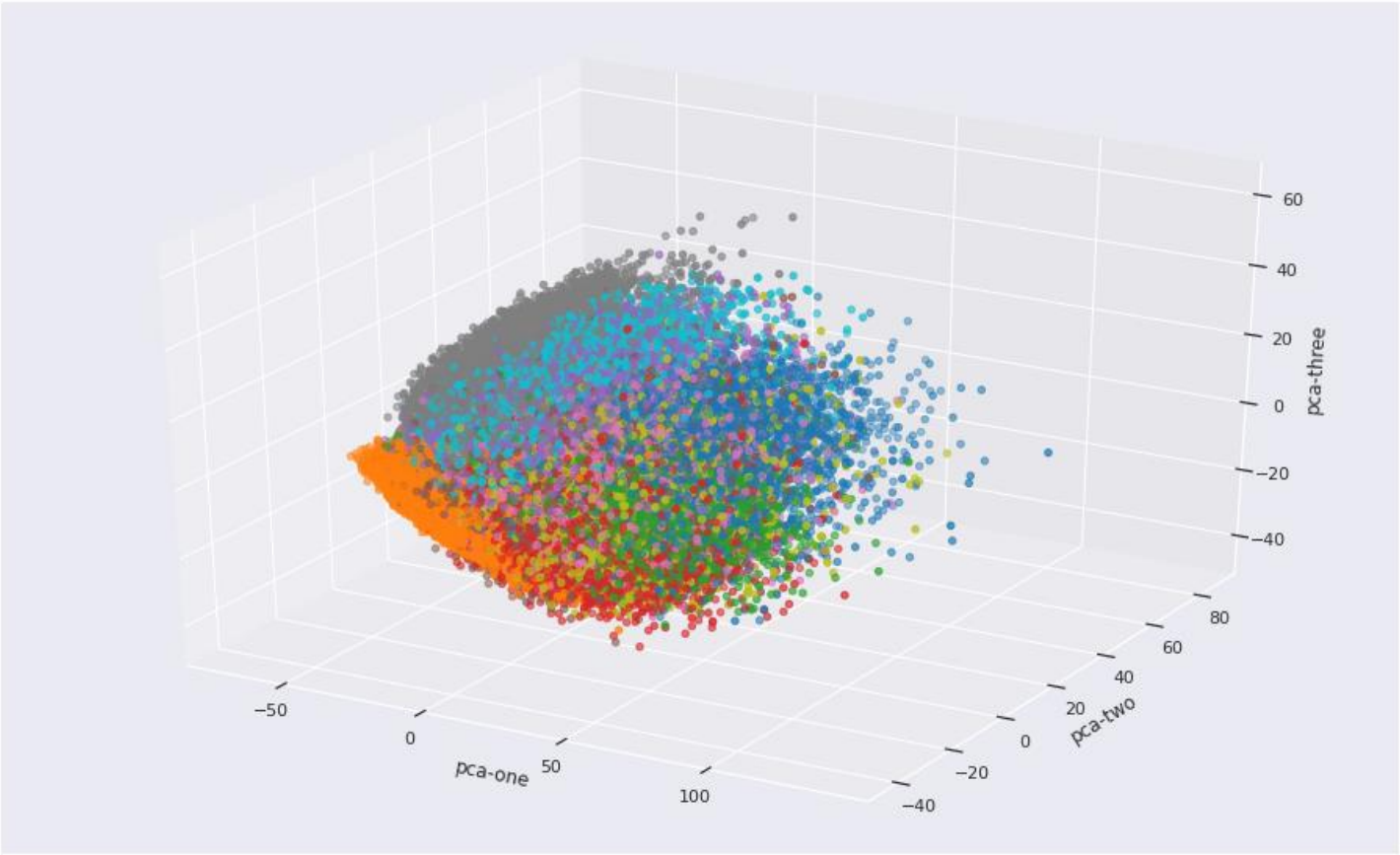


**Figure 13.** A PCA analysis of the 3 principal nodes most responsible for identifying all classes.

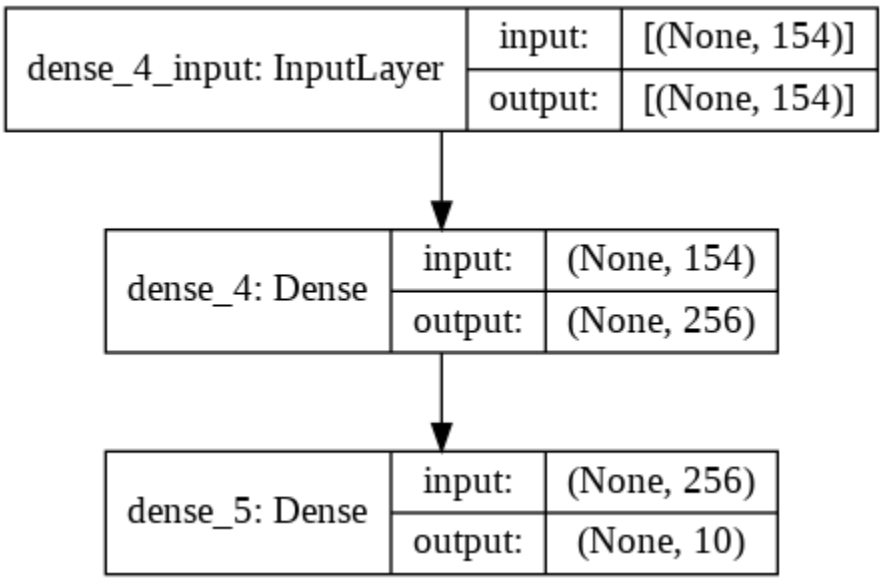


**Figure 14.** A visual architecture of the fourth model, that was based on model 3 with the exception of a reduced input layer to accommodate the training data which has undergone dimensionality reduction through a PCA analysis.

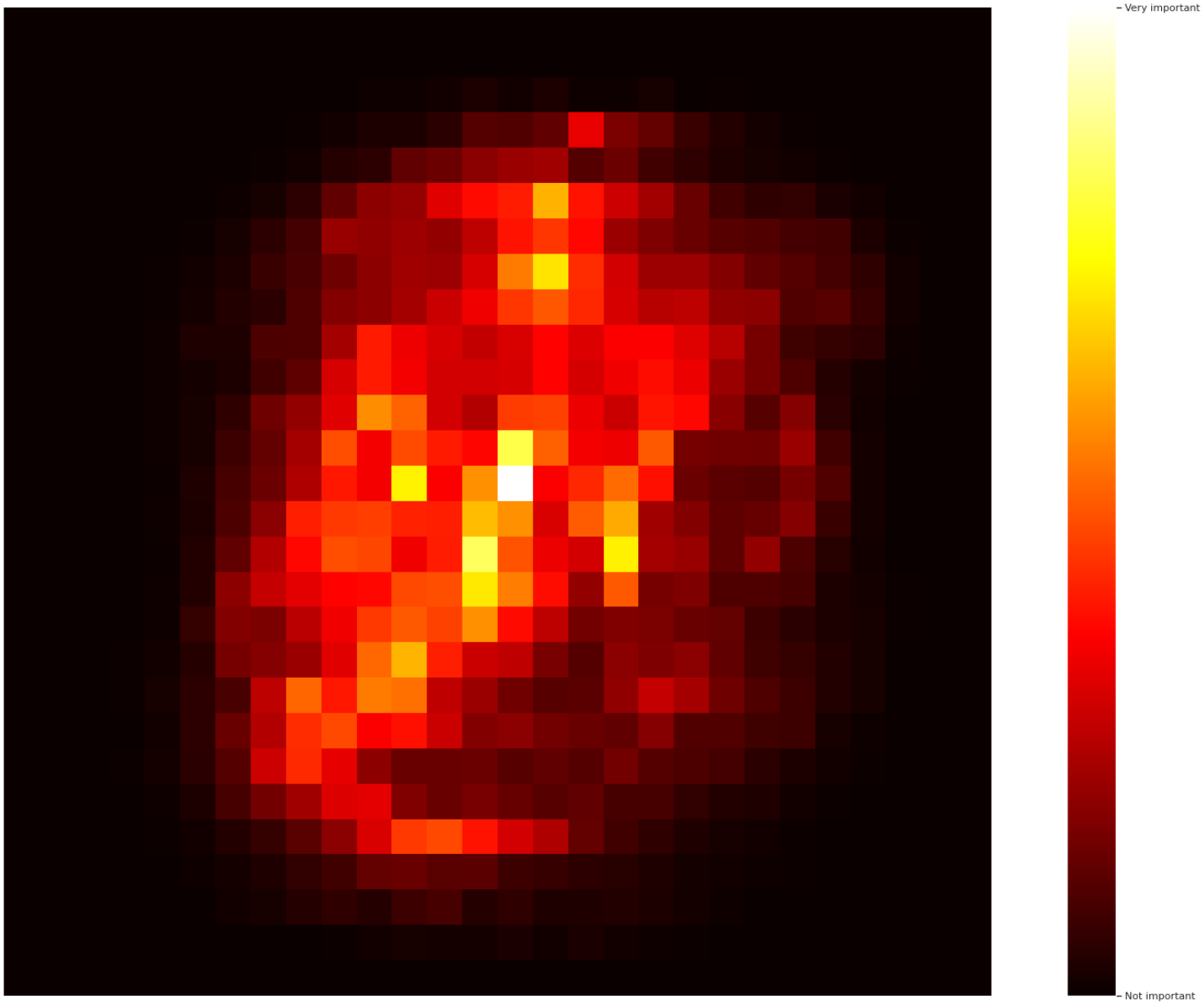


**Figure 15.** A demonstration of the importance of different pixels on the 28x28 MNIST grid and their importance in digit classification.