A.1 First Research/Programming Assignment

By: Robin Singh

**Abstract:**

In this experiment, I will be analyzing the parameters of a dense neural network, and how they work together in creating a successful neural network. I will be working with a simple neural network with a single hidden layer consisting of different numbers of nodes to analyze how they work and what makes a certain number necessary. This experiment will be conducted using the MNIST dataset, a dataset consisting of handwritten digits numbered 0 through 9, an use the dense neural network in trying to successfully predict which digit was written based on values assigned to pixels depending on the attributes of that pixel. We begin by analyzing with a hidden layer consisting of one node, then moving on to two nodes, and finally creating a best performing model with 128 nodes. We compare the activation values with the actual classes of the digits to analyze overlapping values. We also conduct PCA analysis to reduce the dimensionality of the data and evaluate performance after the fact. Finally, we use a Random Forest classifier to find the most important 70 pixels of the data and evaluate performance using only the most important pixel values.

**Introduction:**

This research is conducted to analyze various parameters of a neural network and determine what they mean and how they help make up a successful neural network. For this case we are working with classifying handwritten digits from the MNIST dataset. We need to determine how to preprocess the data, split the data into train, validation, and test sets, and build a neural network capable to accepting the data in the processed format and successfully classify them into one of 10 digits in the end. The accuracy will be judged by how well it is predicting the set of pixel values into digits after being trained by the training set, validated by the validation set, and finally tested on the test set that had not yet been seen by the model, which would be used in place of customer or client handwritten digits in this case. This way, in a real-world scenario, had this occurred, we would now what it takes for a neural network to successfully predict the digits, such as how the data should be shaped, how many hidden layers the dense neural network should have, how the results should be categorized, and what we need to be weary of when feeding the data in (such as similar numbers, human error, etc.).

**Literature Review:**

Many people have worked on classifying the MNIST dataset, as it is one of the first examples of image classifications students of the data science field interact with. One such example is by Tyler Elliot Bettilyon, who posted his walkthrough on creating a neural network on Medium. His neural network was created with an input layer leading to a hidden layer of 32 nodes. This led to an accuracy of around 90%. He then goes on to create a more complex neural network to show how far one may go with Keras and TensorFlow. A link to his experiment will be below, under sources.

**Methods:**

I started off by splitting the MNIST data into train and test data, with the training set consisting of 60,000 images and the test set consisting of 10,000 images. I plotted the data to see what the first few data entries looked like. Then I began preprocessing the MNIST data, reshaping the individual data of each image from 28x28 matrices to single arrays of 784 values. This allowed to be more easily fit into a dense neural network when the time came.

Graphical user interface, text, application

Description automatically generated

For the first experiment, I created a simple dense neural network with an input layer consisting of 784 nodes (the same size of the arrays), a hidden node consisting of 1 node, and the output layer consisting of 10 nodes. To see how such an experiment worked out, I plotted the activation values against the individual classes via boxplot. I had also created a confusion matrix with the probabilities associated with each of the data entries per image to their probable classes.

Diagram

Description automatically generated

For the second experiment, I created another similar dense neural network, but this time with 2 nodes in the hidden layer. I then output the activation values of the two nodes on a scatterplot to analyze overlapping values and evaluate some of the performance. I had also output a confusion matrix for these results to compare against the previous neural network.

Diagram

Description automatically generated

For the third experiment, I worked on founding the best dense neural network setup for this data. I did this by increasing the number of nodes in the hidden layer.

For the fourth experiment, I ran a PCA on the image data to reduce the dimensionality of the 784 value arrays down to 154 values. I then evaluated the performance of the three previously created neural network setups on this PCA dataset and compared the performance.

Graphical user interface, text, application

Description automatically generated

Finally, for experiment 5, I used a Random Forest classifier to find the 70 most important pixels in most of the digit entries. Then, I ran the previous three neural network setups on this dataset and compared the results to the original. Here is a plot showing the most important pixels within the images:

A picture containing graphical user interface

Description automatically generated

Here is a plot showing the most important 70 pixels on top of a digit example:

Chart

Description automatically generated

**Results:**

From the first experiment, hers is the boxplot of the actual classes against the activation values:

Chart, box and whisker chart

Description automatically generated

There is a lot of overlap in the activation values towards the 10 labels we are originally classifying the data on. This is to be expected as we gave the neural network only one node to work with, so a lot of the activation values may overlap, confusing the neural network and leading to a low accuracy when it came to predicting the digit value in the end (around 34%).

Text

Description automatically generated with medium confidence

The training and validation losses were still quite high at this point by the time the accuracy had stopped improving. This indicated a poorly run model all together. It needs more nodes in the hidden layer to be able to differentiate between factors more heavily in the images.

Chart, line chart

Description automatically generated

The model was very confused when it came to classifying digits, almost splitting the percentage chance of each digit evenly when predicting:

Table

Description automatically generated

For the second experiment, we added an additional node into the hidden layer increasing the number of nodes from one to two. This increased our accuracy from around 34% to 69%, more than doubling the accuracy.

Text

Description automatically generated with low confidence

The loss is also significantly lower at around 0.99. We output a scatterplot to analyze the activation values to the actual classes:

Chart, scatter chart

Description automatically generated

There is still a lot of overlap present in the data, but a lot more of the major classes are distinguishable, unlike what was previously seen by the boxplot in experiment 1.

The prediction confidence when classifying classes also improved significantly compared to experiment 1 when analyzing the confusion matrix.

Graphical user interface, text, application, table, Excel

Description automatically generated

For experiment three, I focused on increasing the number of nodes in the hidden layer. I tried 128, and then 256. The former offered around a 97% accuracy, and the latter did not increase the accuracy as much and its increase was negligible as it was most likely due to chance.

128 nodes:

Graphical user interface

Description automatically generated with medium confidence

256 nodes:

A picture containing text

Description automatically generated

Hence, for the best performing model, we will keep the 128 nodes, as it is more efficient in terms of time and resources.

This model also performed confidently when splitting probabilities between digit classes:

Table

Description automatically generated

For experiment 4, we analyzed the same neural network models but after conducting a PCA on the image dataset, reducing its dimensionality from 784 to 158. We plan that these 158 values will retain most of the variance associated with the image data. Running that data through the NN with 1 node in the hidden layer, we get an accuracy of around 36%, greater than the accuracy of the NN running on the full image data.

Text

Description automatically generated with low confidence

This could be explainable by chance or running the epochs for a longer duration due to the accuracy not quite stabilizing ass fast, but perhaps the PCA allows for the activation values to spread out more, allowing the NN to distinguish between some of the features more easily when the data is simplified through PCA. When run through the NN with 2 nodes in the hidden layer we got an accuracy of around 66%.

Graphical user interface

Description automatically generated with low confidence

This is lower than the accuracy of the full image data run though the same NN. This must mean that the increase and decrease are due to chance. Still, this may mean that the PCA does a great job retaining most of the variance in the data as similar results came out of the full image data and the PCA data. Finally, on the best model, the accuracy was around 97%, which is about the same as the full image data. This indicates that PCA is a great way to reduce the dimensionality of the data to use your resources more efficiently on smaller sets of data, that is that would do not simplify the data into too few principal components. 154 seems to have been a good estimate to reduce the dimensionality of the data to.

Graphical user interface, text

Description automatically generated with medium confidence

For the fifth and final experiment, we used Random-Forest Classification to find the 70 most important pixel values of the images and run a NN on those 70 values only. For our first NN with 1 node in the hidden layer, we got an accuracy of around 30%.

A picture containing text

Description automatically generated

This is lower than the NNs before, indicating that 70 pixels may not be enough to remain on par with retaining as much variance as the full data or PCA data had.

The second NN with 2 nodes in the hidden layer has an accuracy of around 48%, which is significantly lower than the accuracy of the same NN setup on the previous datasets.

A picture containing text

Description automatically generated

The accuracy of the best model with this dataset also fell short maxing out at around 93%.

Text

Description automatically generated with low confidence

This further reinforces the fact that not enough variance is held within the 70 most important pixels of an image. These important pixels may also vary image to image, so the 70 most common may not account for as many images as one would like. PCA had performed better than Random Forest, but also help more data.

**Conclusion:**

With these experiments it is determined how important it is to have the correct number of nodes in the activation layer which hold the activation values for the digits to be properly classified into the correct classes. Having more nodes means a wider array of activation function present and varying feature sets the digits can be classified into. This could determine the output of the digits more accurately. However, there is a law of diminishing returns as well. Having more nodes may not necessarily always improve performance. It may result in overfitting onto the training data and lead to poor results on test data. Hence this number should be evaluated carefully.

Moving on to PCA, as seen in the results this was a great way to reduce the dimensionality of the data without losing much of the information that the data provides. This could allow for a more efficient use of resources and time. PCA reduced the dataset to about 20% of the original size and performed as well when run through the best NN model.

Random Forest, however, did not perform as well. However, for how much of its dimensionality was reduced, it still retained a lot of the information from the original model. The best model returned an accuracy of 93% which is good considering how much of the original data was being used (about 9%). When one has a low number of resources, this may be a good alternative. This may be improved it more features were retained, such as perhaps 100 of the most important pixels, or 150.

This experiment showed the importance of nodes within a hidden layer, and how enough are necessary to determine differences in the features within an image.

**Sources:**

Bettilyon, Tyler Elliot. “How to Classify MNIST Digits with Different Neural Network

Architectures.” Medium. Teb's Lab, May 9, 2019. <https://medium.com/tebs-lab/how-to-classify-mnist-digits-with-different-neural-network-architectures-39c75a0f03e3>.