Computer Vision:

Experimentation on Neural Networks with MNIST Handwritten Digits Images

Wicky Woo[1]

Northwestern University School of Professional Studies

633 Clark St, Evanston, IL, 60208

April 15, 2022

[1] Contact email: wicky818@gmail.com

Abstract

The famous computer vision problem with the MNIST (Modified National Institute of Standards and Technology) dataset is a classic dataset for new machine learning enthusiasts to explore how neural networks work. The dataset contains a total of 70,000 black and white images of handwritten digits and each image is presented as a 28 by 28-pixel matrix. Each pixel is represented by an integer ranging from 0 to 255 with 0 meaning white, and 255 meaning completely black.

Image 1. Tonal range from 0-255 shows the intensity of the pixels

This research experiments with different structures of single hidden layer neural networks and shows the performance with a different number of nodes inside the hidden layer. The results show the limitation of the single hidden layer with one node. At first, as the number of nodes increases, the model performance increases dramatically. As the number of nodes increases, the model performance plateaus.

These experiments also show the benefits of utilizing dimensionality reduction methods such as Principal Component Analysis (PCA), which decreases the size of the dataset but increases the accuracy of the model. The experiments also include a Random Forest Classifier to find feature importance to reduce the number of data needed for model training.

Introduction

The focus of this study is to understand the composition of a simple single hidden layer neural network, primarily on how different numbers of nodes would affect the performance of

the neural network. Understanding this effect would allow further exploration of deeper neural networks that have multi-hidden layers.

This exercise utilized a Jupyter Lab instance and launched the Keras library for TensorFlow 2.0 with Python. Keras is a popular deep learning API for the development of machine learning models such as computer vision and natural language processing. Keras also allows the utilization of GPUs when training neural network models. During the experiments, the Python codes are executed to develop the models.

Literature Review

Several computer vision algorithm techniques apply to the analysis of the MNIST data set, and a selection of relevant resources was explored. The important findings are summarized below.

A similar study was conducted to estimate the number of hidden nodes of the single hidden layer neural networks (Cai, 2019). G. Cai and colleagues proposed a method to estimate the optimal number of hidden nodes in a single hidden layer neural network using singular value decomposition of training data with two main operations. First, the training data is normalized. Second, a singular value decomposition is performed on the normalized data to obtain the number of main eigenvalues. Image 2 shares the result from the study using the MNIST dataset.

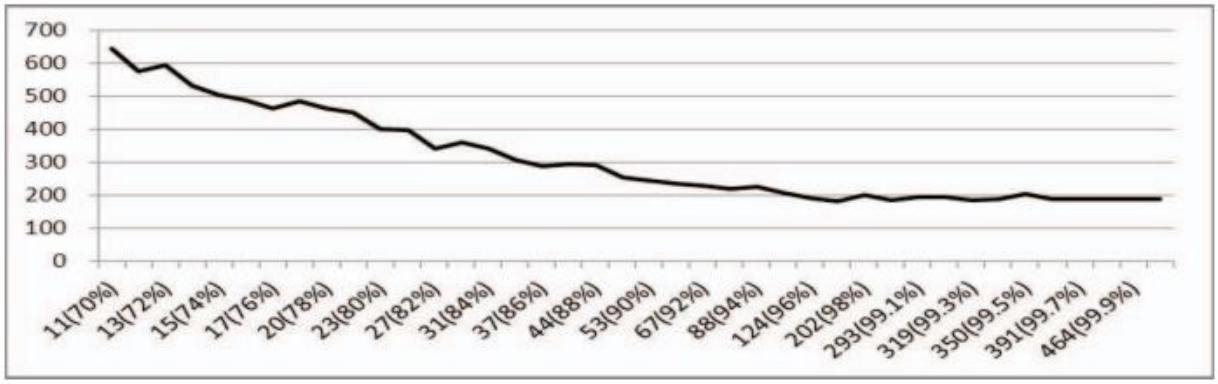

Image 2. The number of errors for different numbers of hidden nodes of the MNIST data set (Cai, 2019)

Methods

The Keras library allows easy data imports for the MNIST dataset which contains 70,000 total images. The dataset is split into a training dataset with 60,000 images and a testing dataset. The dataset also contains the corresponding training dataset labels and testing dataset labels that specify the true value digit (0 to 9) of the datasets. Before diving into the experimentation, some data reshaping and normalization were performed on both the test dataset and the training dataset. Both datasets were reshaped from a two-dimensional 28 by 28-matrix into a one-dimensional 784 vector. Furthermore, the corresponding label datasets were converted from a sparse format to a categorical format using one-hot encoding. A validation set that includes 5,000 images was split from the training dataset to prevent overfitting for all of the neural network training experiments.

For the first experiment, a neural network was constructed with an input layer with 784 nodes for the one-dimensional vector, a single hidden layer with one node with the ReLU activation function, and a SoftMax output layer of 10 nodes. For the second experiment, the single hidden layer had two nodes to allow comparison to the first experiment. The third

experiment consisted of the utilization of the Optuna library. The Optuna library is a hyperparameter tuning library that allows easy experimentation on different hyperparameters to optimize model performance metrics. The goal of the third experiment was to find the optimal number of hidden nodes for the single hidden layer of the neural network. Thirty different experiments were performed to test the number of nodes by randomly selecting a node count ranging from 4 to 170. The Optuna function helped select the number of nodes with the best accuracy from the test dataset.

In the fourth experiment, a dimensionality reduction technique called Principal Component Analysis (PCA) was conducted to reduce the number of features. PCA allows users to pick the number of principal components derived from the original dataset, which could explain most of the variance in the training dataset. In this case, 784 features were reduced to 154 principal components to reduce the time of training which allowed for better generalization of the model. In the fifth experiment, Random Forest Classification was conducted to find the 70 most important features. The goal of the experiment was to see how the model performed with only a tenth of the training dataset.

For the sixth and final experiment, transfer learning was explored to predict the handwritten digits. There are several trained neural networks available from Keras. ResNet50 is a common neural network for image recognition. The accuracy of the experiment was compared to the other experiments.

Results

Experiment 1

With only one node within the single hidden layer in the neural network, the model performed poorly as expected at 36% for the test set accuracy. Graphing the activation values from the hidden layer using a box plot, showed that there is a lot of overlap between the different digit classes. Overlapping activation values for the different digit classes can be interpreted as the model having trouble distinguishing between the different handwritten digit images. Image 3 shows the activation value distribution for each class.

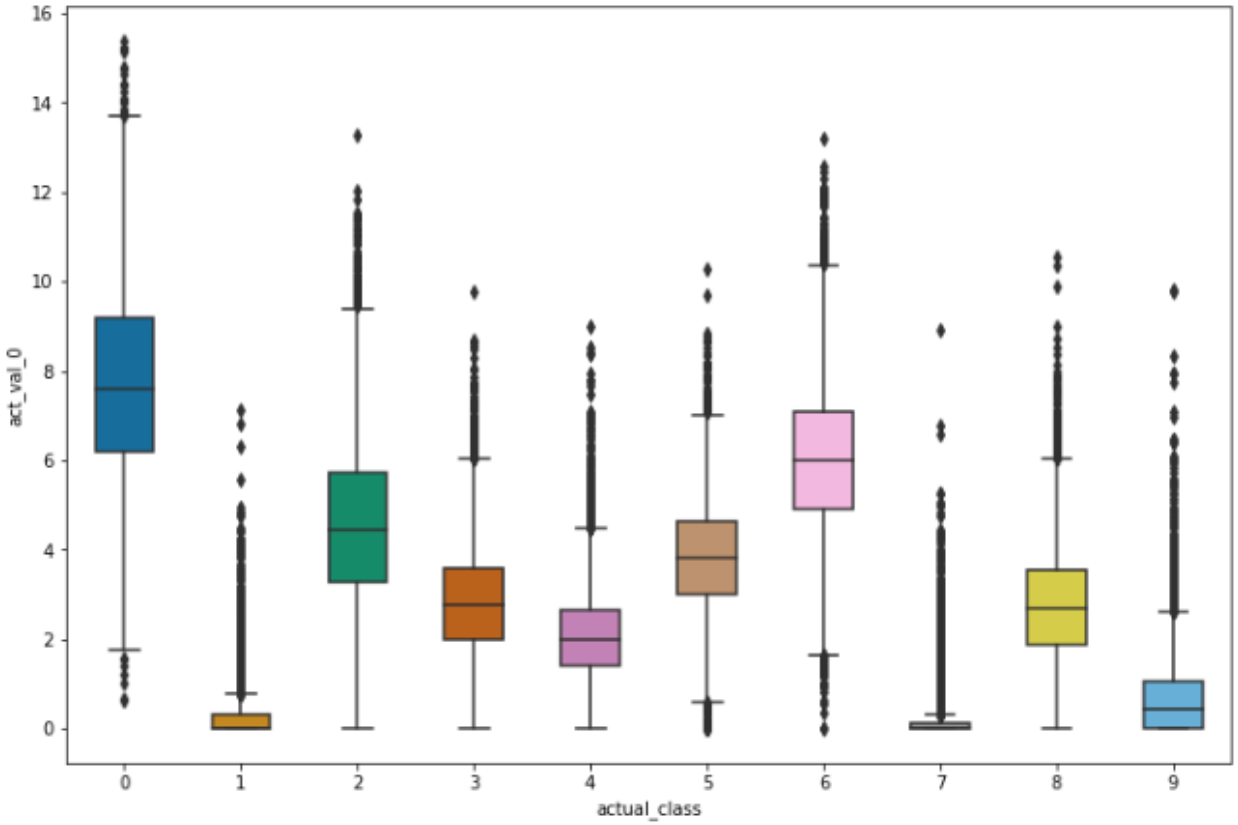

Image 3. Activation values of the different predicted classes

Experiment 1 concludes that one node in the single hidden layer is not enough to make an accurate and robust neural network to classify handwritten images.

Experiment 2

With two nodes within the single hidden layer, the model performed better than with only one node within the single hidden layer in Experiment 1. The result of the new model drastically increases the accuracy of the test set from 36% to 66%. From graphing a scatter plot of the activation values, there is clearly overlap between the different classes, but the clusters can be seen forming from Image 4.

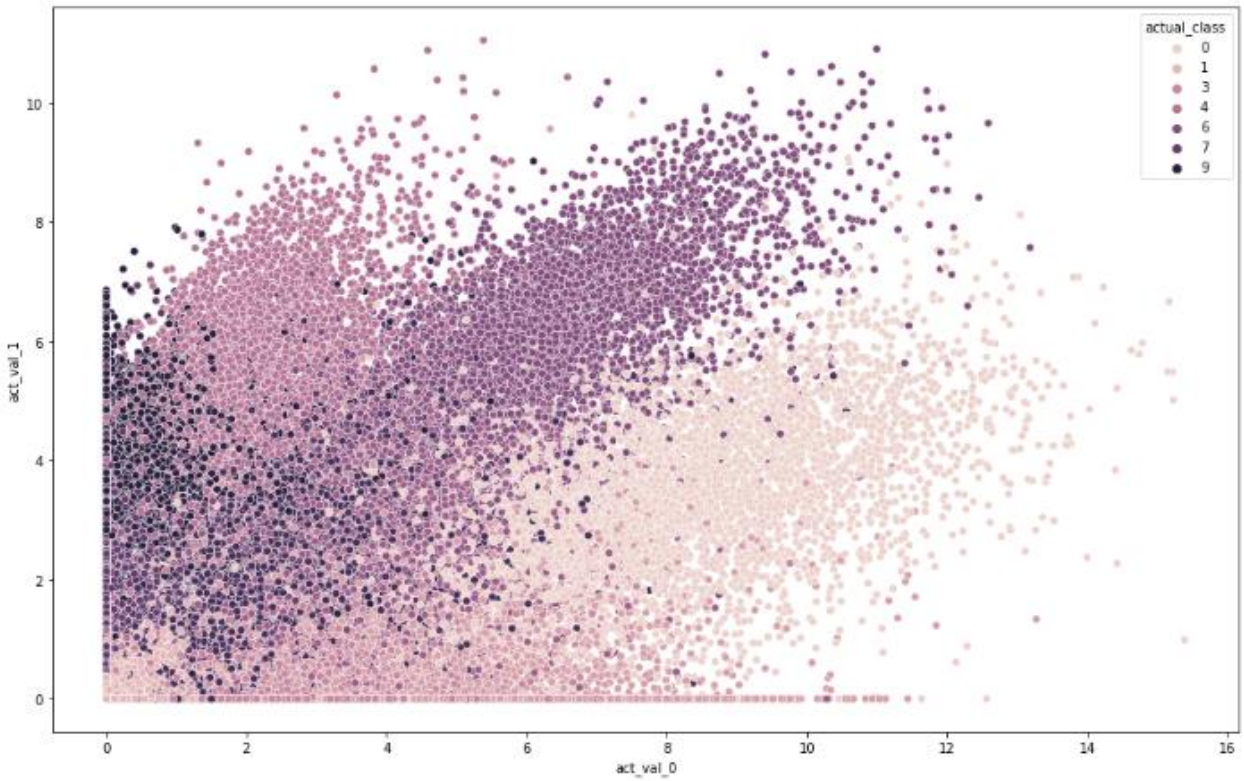

Image 4. Scatter plot for activation values

Experiment 2 concludes that two nodes in the single hidden layer performed better than a single node, but still not enough to make an accurate and robust neural network to classify handwritten images.

Experiment 3

The goal of Experiment 3 was to find the best model by experimenting with different numbers of nodes in the single hidden layer. The Optuna optimization function randomly selected 30 integers ranging from 4 to 170 and performed the model training to find the optimal number of nodes for the highest test accuracy. Image 5 (Appendix) shows the 30 different numbers of nodes and the test accuracy with the corresponding number of nodes. The highest test accuracy from this experiment is 96.8% with 93 hidden nodes. Furthermore, Image 6 shows the result as a line graph on the number of nodes and test set accuracy.

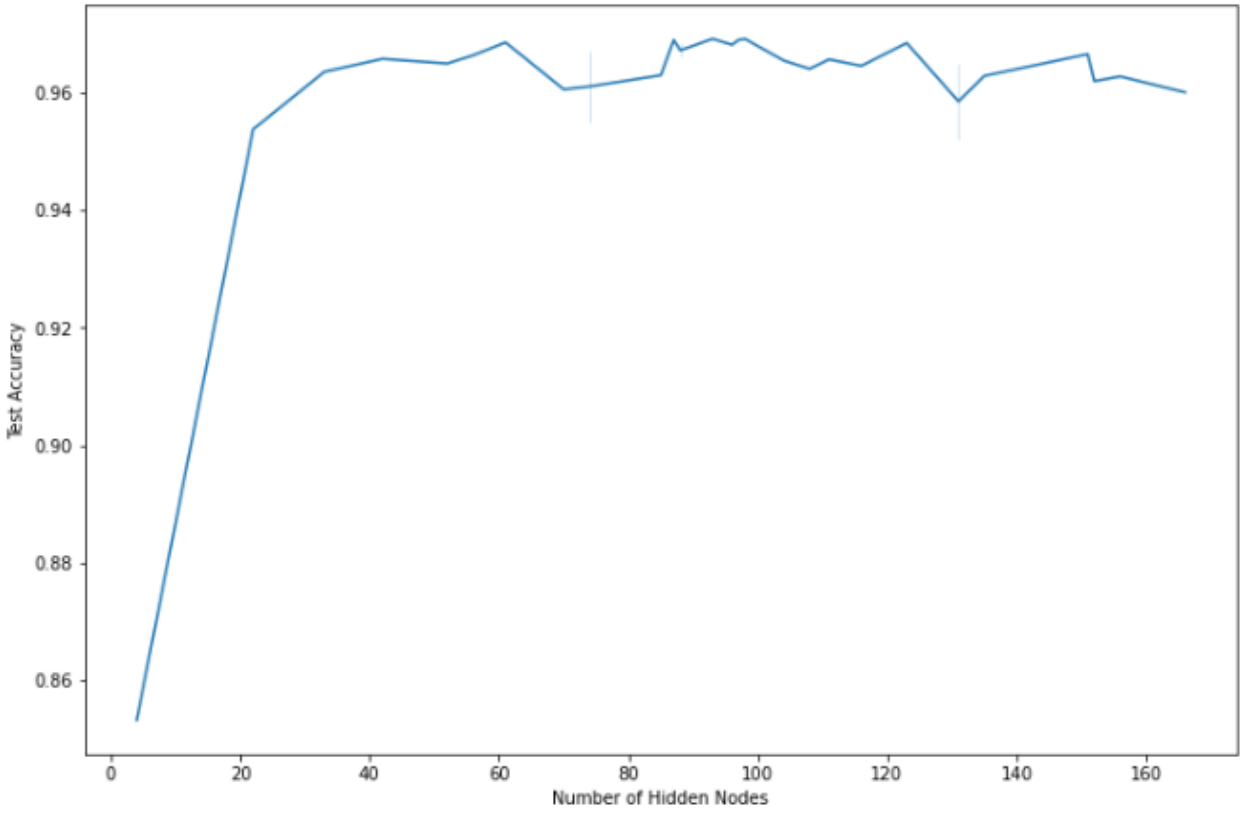

Image 6. Line graph based on the number of nodes (x-axis) and the test accuracy (y-axis)

The model was able to correctly identify all of the first 25 images, see Image 7.

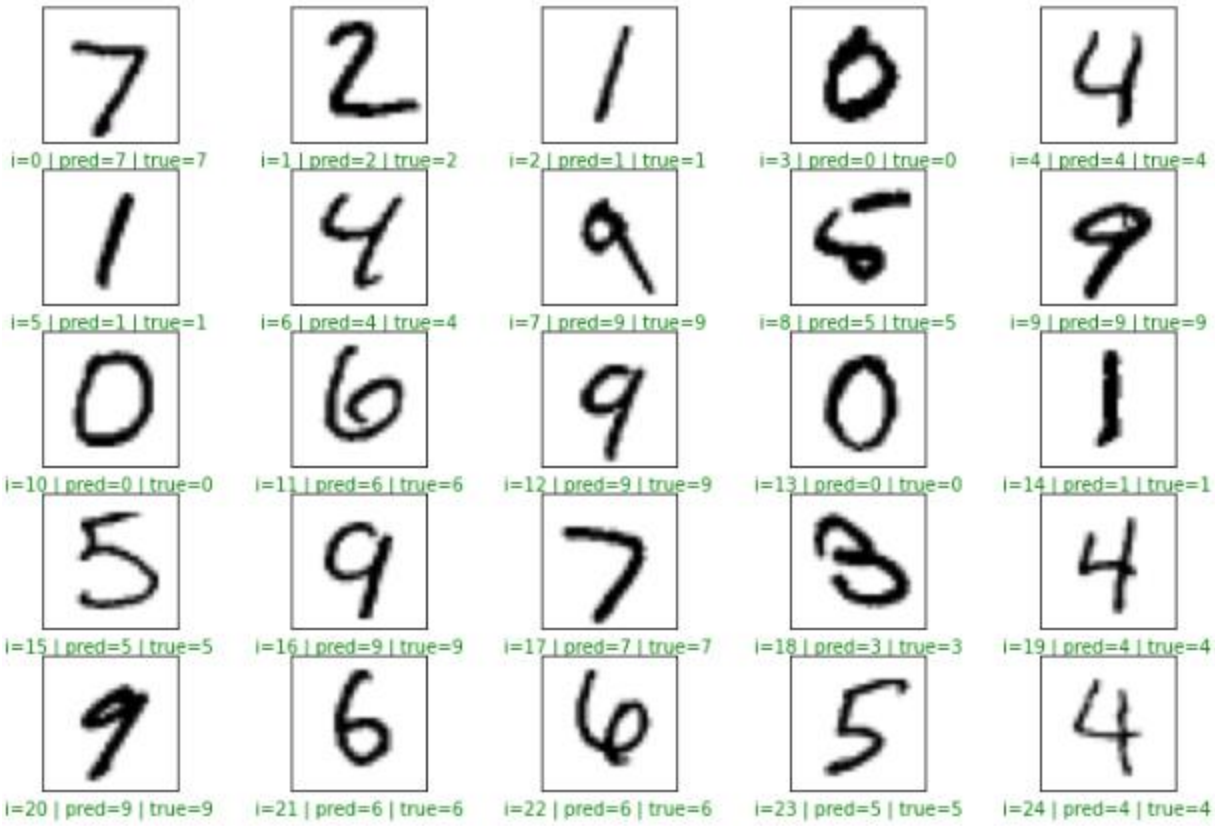

Image 7. Images with the predicted label and the true label

Experiment 4

For Experiment 4, Principal Component Analysis was performed to reduce the number of features from 784 to 154 principal components. The result of the neural network model that was trained on the 154 principal components was better than the "best model" from Experiment 3 with a test set accuracy of 98.3% even though the model structure is the same. Compared to the "best model" from Experiment 3, the test accuracy improved by 1.5%.

Experiment 5

Experiment 5 explores the idea of feature importance. Random Forest Classifier allows the user to identify the most important features within the dataset. In this experiment, the most important 70 pixels out of the 784 pixels from the images were identified to be used to train and test the

model. Image 8 is an example of the 70 pixels selected as the most important from the 784 pixels.

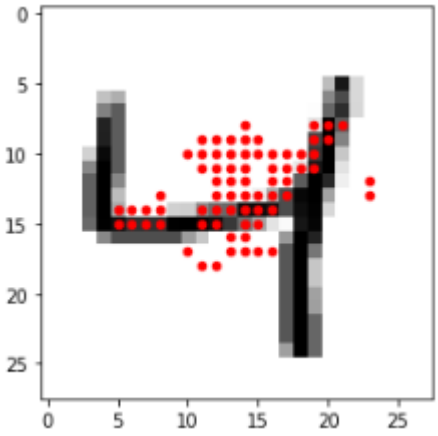

Image 8. 70 selected pixels by feature importance from the Random Forest Classifier

With the most important 70 selected pixels, the test accuracy is 92%. Compared to the 96.8% test set accuracy from Experiment 3, and the 98.3% test set accuracy from Experiment 4, this model is not as accurate, however, it utilizes the least amount of space and memory because of the size of the dataset. Depending on the use case, this can be the ideal approach for efficient handwritten digit recognition.

Experiment 6

For the last experiment, ResNet50, a convolutional neural network with 50 layers, was used to compare the results. Image 9 shows the performance of the ResNet50 model to the MNIST dataset.

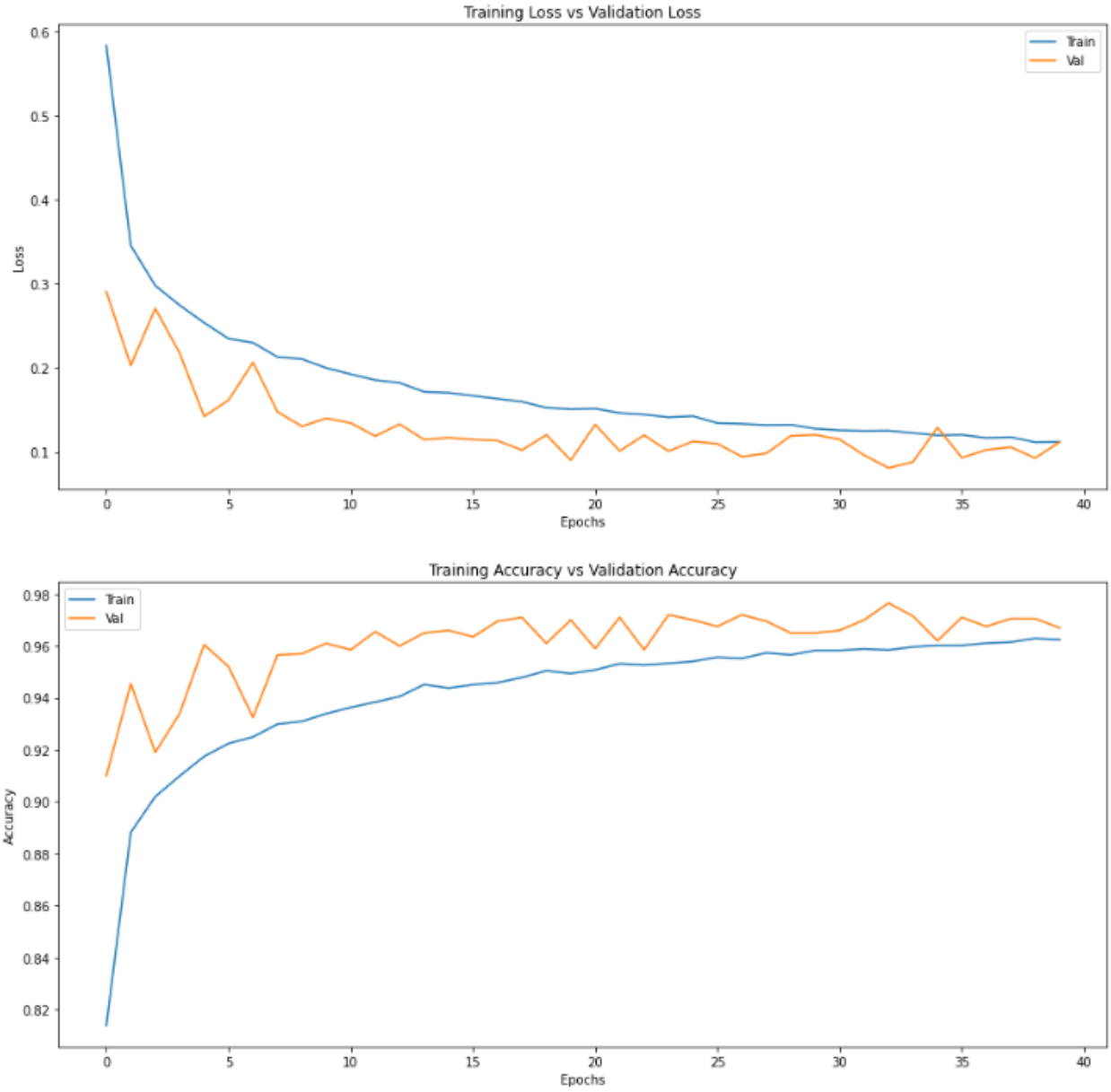

Image 9. Training and Validation Loss and Accuracy

The test accuracy of this ResNet50 implementation is 95.6%. Compared to the other experiments, the performance of this pre-trained model does not outperform the original “best model” with 93 hidden nodes, or the PCA applied model from Experiment 4.

Conclusion

In summary, the neural network can be optimized with hyperparameter tuning methods to find the optimal number of nodes. Furthermore, with Principal Component Analysis, the model actually performs better even with fewer data to the same model architecture. Random Forest Classifier can be used to find the important pixels from the images and is a good way to explore dimensionality reduction, but it sacrifices model accuracy compared to methods like PCA.

Based on these experiments, PCA with hyperparameter tuning to find the optimal number of nodes within a single hidden layer of the neural network would be the recommended approach for handwritten digit image recognition.

Acknowledgments

The author would like to thank Daniel Jensen for providing a code starter.

References

G. Cai, Z. Fang and Y. Chen, "Estimating the Number of Hidden Nodes of the Single-Hidden-Layer Feedforward Neural Networks," 2019 15th International Conference on Computational Intelligence and Security (CIS), 2019, pp. 172-176, Doi: 10.1109/CIS.2019.00044.

He, Kaiming & Zhang, Xiangyu & Ren, Shaoqing & Sun, Jian. (2016). “Deep Residual Learning for Image Recognition”. 770–778. 10.1109/CVPR.2016.90.

Appendix

[104]:

	number	value	datetime_start	datetime_complete	duration	params_units	state
0	0	0.9674	2022-04-16 16:03:52.674576	2022-04-16 16:04:19.776830	0 days 00:00:27.102254	153	COMPLETE
1	1	0.9689	2022-04-16 16:04:19.777855	2022-04-16 16:04:51.477712	0 days 00:00:31.699857	154	COMPLETE
2	2	0.9691	2022-04-16 16:04:51.478708	2022-04-16 16:05:08.681039	0 days 00:00:17.202331	168	COMPLETE
3	3	0.9640	2022-04-16 16:05:08.682036	2022-04-16 16:05:24.341220	0 days 00:00:15.659184	219	COMPLETE
4	4	0.9687	2022-04-16 16:05:24.342248	2022-04-16 16:05:50.845725	0 days 00:00:26.503477	79	COMPLETE
5	5	0.9634	2022-04-16 16:05:50.846750	2022-04-16 16:06:31.765126	0 days 00:00:40.918376	116	COMPLETE
6	6	0.9638	2022-04-16 16:06:31.766121	2022-04-16 16:06:45.239406	0 days 00:00:13.473285	228	COMPLETE
7	7	0.9635	2022-04-16 16:06:45.240431	2022-04-16 16:07:07.215888	0 days 00:00:21.975457	118	COMPLETE
8	8	0.9691	2022-04-16 16:07:07.216885	2022-04-16 16:07:36.578634	0 days 00:00:29.361749	80	COMPLETE
9	9	0.9704	2022-04-16 16:07:36.579619	2022-04-16 16:08:05.448831	0 days 00:00:28.869212	199	COMPLETE
10	10	0.9663	2022-04-16 16:08:05.449836	2022-04-16 16:08:25.198000	0 days 00:00:19.748164	196	COMPLETE
11	11	0.9561	2022-04-16 16:08:25.198000	2022-04-16 16:08:40.694102	0 days 00:00:15.496102	185	COMPLETE
12	12	0.9654	2022-04-16 16:08:40.695099	2022-04-16 16:09:04.870277	0 days 00:00:24.175178	195	COMPLETE
13	13	0.9585	2022-04-16 16:09:04.871261	2022-04-16 16:09:19.873008	0 days 00:00:15.001747	175	COMPLETE
14	14	0.9635	2022-04-16 16:09:19.875015	2022-04-16 16:09:34.843002	0 days 00:00:14.967987	249	COMPLETE
15	15	0.9658	2022-04-16 16:09:34.844000	2022-04-16 16:09:59.512313	0 days 00:00:24.668313	130	COMPLETE
16	16	0.9673	2022-04-16 16:09:59.513310	2022-04-16 16:10:23.089416	0 days 00:00:23.576106	215	COMPLETE
17	17	0.9606	2022-04-16 16:10:23.090425	2022-04-16 16:10:36.336779	0 days 00:00:13.246354	72	COMPLETE
18	18	0.9669	2022-04-16 16:10:36.337783	2022-04-16 16:10:55.082891	0 days 00:00:18.745108	92	COMPLETE
19	19	0.9598	2022-04-16 16:10:55.083887	2022-04-16 16:11:11.763974	0 days 00:00:16.680087	171	COMPLETE
20	20	0.9646	2022-04-16 16:11:11.765000	2022-04-16 16:11:35.737917	0 days 00:00:23.972917	136	COMPLETE
21	21	0.9645	2022-04-16 16:11:35.738914	2022-04-16 16:11:55.757257	0 days 00:00:20.018343	99	COMPLETE
22	22	0.9571	2022-04-16 16:11:55.758256	2022-04-16 16:12:21.380045	0 days 00:00:25.621789	207	COMPLETE
23	23	0.9657	2022-04-16 16:12:21.381061	2022-04-16 16:12:41.018476	0 days 00:00:19.637415	169	COMPLETE
24	24	0.9657	2022-04-16 16:12:41.019474	2022-04-16 16:13:00.328329	0 days 00:00:19.308855	144	COMPLETE
25	25	0.9614	2022-04-16 16:13:00.329326	2022-04-16 16:13:19.411151	0 days 00:00:19.081825	245	COMPLETE
26	26	0.9654	2022-04-16 16:13:19.412176	2022-04-16 16:13:32.855102	0 days 00:00:13.442926	190	COMPLETE
27	27	0.9587	2022-04-16 16:13:32.856098	2022-04-16 16:13:50.301610	0 days 00:00:17.445512	231	COMPLETE
28	28	0.9665	2022-04-16 16:13:50.303605	2022-04-16 16:14:10.100767	0 days 00:00:19.797162	165	COMPLETE
29	29	0.9657	2022-04-16 16:14:10.101787	2022-04-16 16:14:30.279561	0 days 00:00:20.177774	154	COMPLETE

Figure 5. Confusion matrix for predictions of wine type classification.